OMG - Enterprise Knowledge Graph Forum

EKGF Ontologies

EKGF Editor's Draft 11 November 2026

This version:
https://ekgf.github.io/ekg-ontologies/pdf/ekgf-spec-ekg-ontologies.pdf

Latest published version:
https://ekgf.github.io/ekg-ontologies/pdf/ekgf-spec-ekg-ontologies.pdf

Latest editor's draft:
https://github.com/EKGF/ekg-ontologies

Document license

Contents

Abstract . . . 20
Status of This Document . . . 20
1 Family overview . . . 21
1.1 EKGF Ontologies . . . 21
2 Annotation Ontology . . . 24
2.1 Overview . . . 24
2.2 Properties . . . 24
2.2.1 Hexadecimal RGB color . . . 24
2.2.2 Markdown description . . . 24
2.2.3 short title . . . 24
2.2.4 slug . . . 25
2.2.5 Sort key . . . 25
2.2.6 subtitle . . . 26
2.2.7 summary . . . 26
2.3 Shapes . . . 26
2.3.1 Short Title Shape . . . 26
3 Artifact Dependency Ontology . . . 27
3.1 Overview . . . 27
3.2 Classes . . . 27
3.2.1 Dependency . . . 27
3.2.2 Dependency evidence . . . 27
3.2.3 Dependency kind . . . 28
3.2.4 Dependency resolution . . . 28
3.2.5 Explicit dependency . . . 29
3.2.6 Implicit dependency . . . 29
3.2.7 Ontology load request . . . 30
3.2.8 Resolution status . . . 30
3.2.9 Semantic artifact . . . 31
3.3 Properties . . . 31
3.3.1 dependency evidence . . . 31
3.3.2 dependency kind . . . 32
3.3.3 dependency target . . . 32
3.3.4 dependent artifact . . . 32
3.3.5 depends on . . . 33
3.3.6 evidence graph . . . 33
3.3.7 evidence object . . . 33
3.3.8 evidence predicate . . . 34
3.3.9 evidence subject . . . 34
3.3.10 has dependency . . . 34
3.3.11 load hint . . . 34
3.3.12 load request authority . . . 35
3.3.13 load request dependency . . . 35
3.3.14 load request family . . . 35
3.3.15 load request target . . . 36
3.3.16 referenced namespace . . . 36
3.3.17 resolution . . . 36
3.3.18 resolution branch . . . 36
3.3.19 resolution commit . . . 37
3.3.20 resolution message . . . 37

3.3.21 resolution status . . . 37
3.3.22 resolution workspace . . . 38
3.3.23 resolved artifact . . . 38
3.4 Named Individuals . . . 38
3.4.1 Ambiguous . . . 38
3.4.2 Declared import dependency . . . 39
3.4.3 External . . . 39
3.4.4 Failed . . . 39
3.4.5 Incomplete . . . 39
3.4.6 Loaded . . . 40
3.4.7 Namespace reference dependency . . . 40
3.4.8 Profile dependency . . . 40
3.4.9 Shape graph dependency . . . 41
3.4.10 Term reference dependency . . . 41
3.4.11 Unloaded . . . 41
3.4.12 Unsupported . . . 42
3.4.13 Validation dependency . . . 42
4 Artifact Governance Ontology . . . 43
4.1 Overview . . . 43
4.2 Classes . . . 43
4.2.1 Authority . . . 43
4.2.2 Authority publishing role . . . 44
4.2.3 Family . . . 44
4.2.4 Module . . . 45
4.2.5 Organization category . . . 46
4.2.6 Reference . . . 46
4.2.7 Standards community role . . . 47
4.3 Properties . . . 47
4.3.1 Authority acronym . . . 47
4.3.2 Authority host . . . 48
4.3.3 Auto-load . . . 48
4.3.4 Base IRI . . . 48
4.3.5 Community organization . . . 49
4.3.6 Community role . . . 49
4.3.7 Family acronym . . . 49
4.3.8 File path prefix . . . 50
4.3.9 Git branch . . . 50
4.3.10 Git path . . . 50
4.3.11 Git repository URL . . . 51
4.3.12 Governed by . . . 51
4.3.13 Graph IRI prefix . . . 51
4.3.14 Historical name . . . 52
4.3.15 In family . . . 52
4.3.16 In module . . . 52
4.3.17 Legal owner . . . 53
4.3.18 Maximum family load quads . . . 53
4.3.19 Maximum source quads . . . 53
4.3.20 Namespace IRI prefix . . . 54
4.3.21 Organization category . . . 54
4.3.22 Predecessor authority . . . 54
4.3.23 Publishing role . . . 55

4.3.24 RDF file extension . . . 55
4.3.25 Refers to . . . 55
4.3.26 Source graph repository URL . . . 56
4.3.27 Successor authority . . . 56
4.4 Named Individuals . . . 56
4.4.1 co-owner . . . 56
4.4.2 maintainer . . . 57
4.4.3 owner . . . 57
4.4.4 publisher . . . 57
4.4.5 Community . . . 58
4.4.6 Consortium . . . 58
4.4.7 Organization categories for artifact governance . . . 58
4.4.8 Project . . . 59
4.4.9 Standards body . . . 59
4.4.10 standards editor . . . 59
4.4.11 standards publisher . . . 60
4.4.12 standards reviewer . . . 60
4.4.13 standards staff . . . 60
4.4.14 working-group chair . . . 61
4.5 Shapes . . . 61
4.5.1 Module Membership Shape . . . 61
4.5.2 Module Shape . . . 62
5 Artifact Publication Ontology . . . 63
5.1 Overview . . . 63
5.2 Classes . . . 63
5.2.1 Publication artifact . . . 63
5.2.2 Publication profile . . . 64
5.2.3 Publication run . . . 65
5.2.4 Publication target . . . 66
5.2.5 Release approval . . . 66
5.2.6 Standards release . . . 67
5.3 Properties . . . 68
5.3.1 Approved at . . . 68
5.3.2 Approved by . . . 69
5.3.3 Approves release . . . 69
5.3.4 Artifact checksum . . . 69
5.3.5 Artifact media type . . . 69
5.3.6 Artifact URL . . . 70
5.3.7 Generator version . . . 70
5.3.8 Input commit CID . . . 70
5.3.9 Publication artifact . . . 71
5.3.10 Publication branch . . . 71
5.3.11 Publication completed at . . . 71
5.3.12 Publication profile . . . 71
5.3.13 Publication release . . . 72
5.3.14 Publication run . . . 72
5.3.15 Publication started at . . . 72
5.3.16 Publication target . . . 72
5.3.17 Publishes family . . . 73
5.3.18 Publishes reference . . . 73
5.3.19 Release family . . . 73

5.3.20 Release reference . . . 74
5.3.21 Release version . . . 74
5.3.22 Released at . . . 74
5.3.23 Target kind . . . 75
5.3.24 Target URL . . . 75
5.4 Shapes . . . 75
5.4.1 Publication Profile Shape . . . 75
5.4.2 Release Approval Shape . . . 76
5.4.3 Standards Release Shape . . . 77
6 Authorization Ontology . . . 79
6.1 Overview . . . 79
6.2 Classes . . . 79
6.2.1 Authorization Group . . . 79
6.2.2 Authorization Rule . . . 79
6.3 Properties . . . 80
6.3.1 Has Group . . . 80
6.3.2 Has Persona . . . 80
7 Capability Ontology . . . 81
7.1 Overview . . . 81
7.2 Classes . . . 81
7.2.1 Capability . . . 81
7.2.2 Capability Domain . . . 81
7.2.3 Sub-capability . . . 82
7.3 Named Individuals . . . 82
7.3.1 Data sharing and marketplace . . . 82
7.3.2 Designing and managing data products . . . 82
8 Communication Ontology . . . 84
8.1 Overview . . . 84
8.2 Classes . . . 84
8.2.1 Audience . . . 84
8.2.2 Audience membership . . . 84
8.2.3 Campaign audience snapshot . . . 85
8.2.4 Campaign recipient . . . 85
8.2.5 Communication attempt . . . 86
8.2.6 Communication channel . . . 86
8.2.7 Communication endpoint . . . 87
8.2.8 Communication program . . . 87
8.2.9 Communication purpose . . . 88
8.2.10 Delivery event . . . 88
8.2.11 Delivery status . . . 89
8.2.12 Email endpoint . . . 89
8.2.13 Email recipient . . . 90
8.2.14 Group recipient . . . 90
8.2.15 Human recipient . . . 91
8.2.16 In-app endpoint . . . 91
8.2.17 In-app recipient . . . 92
8.2.18 Legal basis . . . 92
8.2.19 List membership . . . 93
8.2.20 Mailbox recipient . . . 93
8.2.21 Marketing program . . . 94
8.2.22 Membership . . . 94

8.2.23 Membership derivation evidence . . . 95
8.2.24 Membership event . . . 95
8.2.25 Membership evidence . . . 96
8.2.26 Membership status . . . 96
8.2.27 Organization recipient . . . 97
8.2.28 Postal endpoint . . . 97
8.2.29 Postal recipient . . . 98
8.2.30 Push endpoint . . . 98
8.2.31 Push recipient . . . 99
8.2.32 Recipient . . . 99
8.2.33 Recipient collection . . . 100
8.2.34 Recipient consent . . . 100
8.2.35 Recipient consent withdrawal . . . 101
8.2.36 Recipient legal category . . . 101
8.2.37 Recipient legal classification . . . 102
8.2.38 Recipient list . . . 102
8.2.39 Recipient suppression . . . 103
8.2.40 Recipient tag . . . 103
8.2.41 Service recipient . . . 104
8.2.42 SMS endpoint . . . 104
8.2.43 SMS recipient . . . 105
8.2.44 Tag assignment . . . 105
8.2.45 Voice endpoint . . . 106
8.2.46 Voice recipient . . . 106
8.2.47 Webhook endpoint . . . 107
8.2.48 Webhook recipient . . . 107
8.3 Properties . . . 108
8.3.1 allowed channel . . . 108
8.3.2 attempt endpoint . . . 108
8.3.3 attempt recipient . . . 109
8.3.4 campaign audience . . . 109
8.3.5 captured at . . . 109
8.3.6 collection purpose . . . 109
8.3.7 communication program . . . 110
8.3.8 consent membership . . . 110
8.3.9 delivery attempt . . . 110
8.3.10 delivery status . . . 111
8.3.11 derived from membership . . . 111
8.3.12 effective from . . . 111
8.3.13 effective through . . . 111
8.3.14 endpoint address . . . 112
8.3.15 endpoint channel . . . 112
8.3.16 evidence collection . . . 112
8.3.17 evidence source . . . 113
8.3.18 has delivery event . . . 113
8.3.19 has endpoint . . . 113
8.3.20 has membership . . . 114
8.3.21 jurisdiction code . . . 114
8.3.22 legal basis . . . 114
8.3.23 legal category . . . 114
8.3.24 legal classification recipient . . . 115

8.3.25 legal jurisdiction . . . 115
8.3.26 membership collection . . . 115
8.3.27 membership endpoint . . . 116
8.3.28 membership event . . . 116
8.3.29 membership evidence . . . 116
8.3.30 membership recipient . . . 116
8.3.31 membership status . . . 117
8.3.32 occurred at . . . 117
8.3.33 policy version . . . 117
8.3.34 program channel . . . 118
8.3.35 program purpose . . . 118
8.3.36 provenance . . . 118
8.3.37 provider message id . . . 119
8.3.38 provider recipient id . . . 119
8.3.39 recorded at . . . 119
8.3.40 represented resource . . . 119
8.3.41 retention period . . . 120
8.3.42 retention policy . . . 120
8.3.43 suppression membership . . . 120
8.3.44 withdrawn consent . . . 121
8.4 Named Individuals . . . 121
8.4.1 Active membership . . . 121
8.4.2 Bounced . . . 121
8.4.3 Complained . . . 122
8.4.4 Consent legal basis . . . 122
8.4.5 Consent withdrawn membership . . . 122
8.4.6 Contract legal basis . . . 123
8.4.7 Delivered . . . 123
8.4.8 Email channel . . . 123
8.4.9 Employee category . . . 123
8.4.10 Employee operational legal basis . . . 124
8.4.11 Failed . . . 124
8.4.12 Group mailbox category . . . 124
8.4.13 In-app notification channel . . . 125
8.4.14 Inactive membership . . . 125
8.4.15 Legal notice purpose . . . 125
8.4.16 Legal obligation legal basis . . . 126
8.4.17 Legitimate interest legal basis . . . 126
8.4.18 Marketing purpose . . . 126
8.4.19 Natural person category . . . 126
8.4.20 Operational purpose . . . 127
8.4.21 Organization category . . . 127
8.4.22 Pending . . . 127
8.4.23 Postal mail channel . . . 128
8.4.24 Public authority category . . . 128
8.4.25 Push notification channel . . . 128
8.4.26 Security purpose . . . 128
8.4.27 Sent . . . 129
8.4.28 Skipped . . . 129
8.4.29 SMS channel . . . 129
8.4.30 Support purpose . . . 130

8.4.31 Suppressed . . . 130
8.4.32 Suppressed membership . . . 130
8.4.33 Technical service category . . . 130
8.4.34 Technical service legal basis . . . 131
8.4.35 Transactional purpose . . . 131
8.4.36 Unsubscribed . . . 131
8.4.37 Voice call channel . . . 132
8.4.38 Webhook channel . . . 132
8.5 Shapes . . . 132
8.5.1 Communication Attempt Shape . . . 132
8.5.2 Communication Endpoint Shape . . . 133
8.5.3 Communication Program Shape . . . 133
8.5.4 Delivery Event Shape . . . 133
8.5.5 Email Recipient Shape . . . 134
8.5.6 Membership Shape . . . 134
8.5.7 Recipient Collection Shape . . . 135
8.5.8 Recipient Consent Shape . . . 135
8.5.9 Recipient Legal Classification Shape . . . 136
8.5.10 Recipient Shape . . . 136
9 Concept Ontology . . . 137
9.1 Overview . . . 137
9.1.1 Why this much detail: data lineage . . . 137
9.1.2 Meaning, manifestations and sightings . . . 137
9.1.3 Start with business language . . . 138
9.1.4 From capture to ontology mapping . . . 138
9.1.5 Reuse across the use case tree . . . 138
9.1.6 Data elements in stories . . . 138
9.1.7 Relationship to ISO/IEC 11179 . . . 138
9.1.8 Vocabulary class labels . . . 138
9.2 Classes . . . 139
9.2.1 Abbreviation . . . 139
9.2.2 Business Term . . . 139
9.2.3 Concept Manifestation . . . 140
9.2.4 Concept Set . . . 140
9.2.5 Cypher Identifier Manifestation . . . 141
9.2.6 Data Element Concept . . . 141
9.2.7 Description . . . 143
9.2.8 Full Plural Business Term . . . 144
9.2.9 Full Singular Business Term . . . 144
9.2.10 JSON Field Manifestation . . . 145
9.2.11 Java Identifier Manifestation . . . 145
9.2.12 JavaScript Identifier Manifestation . . . 146
9.2.13 Literal Datatype Manifestation . . . 147
9.2.14 Manifestation Occurrence . . . 147
9.2.15 OWL Axiom Manifestation . . . 148
9.2.16 OWL Class Manifestation . . . 149
9.2.17 OWL Named Individual Manifestation . . . 149
9.2.18 OWL Property Manifestation . . . 150
9.2.19 Python Identifier Manifestation . . . 150
9.2.20 RDF Predicate Manifestation . . . 151
9.2.21 RDFS Class Manifestation . . . 151

9.2.22 Raw Source Field Manifestation . . . 152
9.2.23 SHACL Shape Manifestation . . . 152
9.2.24 SHACL Shape Set Manifestation . . . 153
9.2.25 SKOS Concept Manifestation . . . 153
9.2.26 SKOS Concept Scheme Manifestation . . . 154
9.2.27 SPARQL Result Column Manifestation . . . 154
9.2.28 SPARQL Variable Manifestation . . . 155
9.2.29 SQL Column Manifestation . . . 156
9.2.30 SQL Table Manifestation . . . 156
9.2.31 Source Code Identifier Manifestation . . . 157
9.2.32 Statement Position . . . 157
9.2.33 Tagline . . . 158
9.2.34 Technical Manifestation . . . 158
9.2.35 Template Language . . . 159
9.2.36 Template Parameter Manifestation . . . 160
9.2.37 URI Parameter Manifestation . . . 160
9.3 Properties . . . 161
9.3.1 concept . . . 161
9.3.2 found in . . . 161
9.3.3 has concept . . . 162
9.3.4 has manifestation . . . 162
9.3.5 in concept set . . . 162
9.3.6 manifestation resource . . . 163
9.3.7 manifestation value . . . 163
9.3.8 occurrence of . . . 163
9.3.9 occurs within . . . 164
9.3.10 preferred business term . . . 164
9.3.11 preferred manifestation . . . 164
9.3.12 preferred query parameter manifestation . . . 165
9.3.13 preferred result column manifestation . . . 165
9.3.14 source column . . . 165
9.3.15 source commit . . . 166
9.3.16 source line . . . 166
9.3.17 source path . . . 166
9.3.18 source repository . . . 167
9.3.19 source system . . . 167
9.3.20 statement position . . . 167
9.3.21 taxonomy kind . . . 168
9.3.22 template language . . . 168
9.4 Named Individuals . . . 169
9.4.1 Graph Pattern . . . 169
9.4.2 Graph Template . . . 169
9.4.3 Parameter . . . 169
9.4.4 Projection . . . 170
9.5 Shapes . . . 170
9.5.1 Legacy Template Manifestation Shape . . . 170
9.5.2 Occurrence Containment Shape . . . 171
9.5.3 Occurrence Sighting Shape . . . 171
9.5.4 Statement Position Shape . . . 172
9.5.5 Template Language Shape . . . 173
9.5.6 Template Parameter Manifestation Shape . . . 173

10 DataOps Ontology . . . 175
10.1 Overview . . . 175
10.2 Classes . . . 175
10.2.1 Failed RDF Load Request . . . 175
10.2.2 Finished RDF Load Request . . . 175
10.2.3 RDF Load Request . . . 176
10.2.4 Loading RDF Load Request . . . 176
10.2.5 Logical RDF Dataset . . . 177
10.2.6 Pipeline . . . 177
10.2.7 Queued RDF Load Request . . . 178
10.2.8 Single-graph RDF Dataset . . . 178
10.3 Properties . . . 179
10.3.1 Created By Pipeline . . . 179
10.3.2 Load Request Target Graph . . . 179
10.3.3 In Pipeline . . . 180
10.3.4 Loaded By Load Request . . . 180
10.3.5 Load Request Source IRI . . . 180
11 File System Ontology . . . 181
11.1 Overview . . . 181
11.2 Classes . . . 181
11.2.1 archive entry kind . . . 181
11.2.2 archive format . . . 181
11.2.3 archive member . . . 182
11.2.4 archive version . . . 182
11.2.5 attachment . . . 183
11.2.6 attachment mode . . . 184
11.2.7 attachment role . . . 184
11.2.8 capability profile . . . 185
11.2.9 directory . . . 186
11.2.10 directory entry . . . 186
11.2.11 external object . . . 187
11.2.12 file series . . . 187
11.2.13 file system view . . . 188
11.2.14 file version . . . 188
11.2.15 name family rule . . . 189
11.2.16 path rule . . . 190
11.2.17 service shortcut . . . 190
11.2.18 source kind . . . 190
11.2.19 suffix pattern . . . 191
11.2.20 symbolic link . . . 191
11.2.21 tombstone . . . 192
11.3 Properties . . . 192
11.3.1 allowed suffix pattern . . . 192
11.3.2 also allow final/draft suffix . . . 192
11.3.3 archive format . . . 193
11.3.4 archive version . . . 193
11.3.5 attached to . . . 193
11.3.6 attachment mode . . . 193
11.3.7 attachment role . . . 194
11.3.8 base name . . . 194
11.3.9 byte size . . . 194

11.3.10 capability profile . . . 195
11.3.11 case sensitive names . . . 195
11.3.12 content . . . 195
11.3.13 current version . . . 195
11.3.14 deleted at . . . 196
11.3.15 entry count . . . 196
11.3.16 entry kind . . . 196
11.3.17 extension . . . 197
11.3.18 external id . . . 197
11.3.19 external object . . . 197
11.3.20 external revision id . . . 197
11.3.21 folder . . . 198
11.3.22 has attachment . . . 198
11.3.23 has version . . . 198
11.3.24 in view . . . 199
11.3.25 legal name pattern . . . 199
11.3.26 link target . . . 199
11.3.27 member count . . . 199
11.3.28 member mode . . . 200
11.3.29 member mtime . . . 200
11.3.30 member ordinal . . . 200
11.3.31 member path . . . 201
11.3.32 member target . . . 201
11.3.33 name . . . 201
11.3.34 normalized name . . . 202
11.3.35 normalized source file name . . . 202
11.3.36 observed at . . . 202
11.3.37 office package kind . . . 202
11.3.38 opaque reason . . . 203
11.3.39 parent . . . 203
11.3.40 provider . . . 203
11.3.41 root . . . 204
11.3.42 source file name . . . 204
11.3.43 source kind . . . 204
11.3.44 supports atomic move . . . 205
11.3.45 supports links . . . 205
11.3.46 supports multiple roots . . . 205
11.3.47 supports range reads . . . 205
11.3.48 supports server-side copy . . . 206
11.3.49 supports shortcuts . . . 206
11.3.50 supports stable revision ids . . . 206
11.3.51 supports trash . . . 207
11.3.52 supports versioning . . . 207
11.3.53 symbolic target path . . . 207
11.3.54 target . . . 208
11.3.55 target external object . . . 208
11.3.56 target name family rule . . . 208
11.3.57 target path rule . . . 208
11.3.58 target series . . . 209
11.3.59 target version . . . 209
11.3.60 tombstone of . . . 209

11.3.61 unicode normalization . . . 210
11.3.62 view path . . . 210
11.4 Other Terms . . . 210
11.4.1 archive directory entry . . . 210
11.4.2 exact version . . . 210
11.4.3 gzip . . . 211
11.4.4 hardlink entry . . . 211
11.4.5 native upload . . . 211
11.4.6 nested archive . . . 211
11.4.7 other entry . . . 212
11.4.8 provider sync . . . 212
11.4.9 regular file . . . 212
11.4.10 symlink entry . . . 212
11.4.11 tar . . . 213
11.4.12 tar+gzip . . . 213
11.4.13 track external id . . . 213
11.4.14 track name family . . . 213
11.4.15 track path . . . 214
11.4.16 track series . . . 214
11.4.17 windows copy suffix . . . 214
11.4.18 zip . . . 214
12 Lifecycle Ontology . . . 216
12.1 Overview . . . 216
12.2 Classes . . . 216
12.2.1 Lifecycle . . . 216
12.2.2 Phase . . . 217
12.2.3 State . . . 217
12.2.4 State transition . . . 218
12.2.5 Transition . . . 219
12.3 Properties . . . 220
12.3.1 applies to . . . 220
12.3.2 current state . . . 220
12.3.3 follows lifecycle . . . 221
12.3.4 from state . . . 221
12.3.5 in lifecycle . . . 221
12.3.6 in state . . . 222
12.3.7 is terminal . . . 222
12.3.8 order . . . 222
12.3.9 phase . . . 223
12.3.10 prior state transition . . . 223
12.3.11 to state . . . 223
12.3.12 trigger . . . 224
12.4 Shapes . . . 224
12.4.1 Current State Shape . . . 224
12.4.2 History State Shape . . . 226
12.4.3 Legacy State Shape . . . 226
12.4.4 State Transition Shape . . . 226
12.4.5 Transition Shape . . . 228
13 Maturity Model Ontology . . . 230
13.1 Overview . . . 230
13.2 Classes . . . 230

13.2.1 Capability . . . 230
13.2.2 Capability Area . . . 230
13.2.3 Level . . . 231
13.2.4 Model . . . 231
13.2.5 Pillar . . . 232
13.3 Properties . . . 232
13.3.1 Capability Number . . . 232
13.3.2 hasCapability . . . 233
13.3.3 hasLevel . . . 233
13.3.4 hasPillar . . . 233
13.3.5 inArea . . . 234
13.3.6 inModel . . . 234
13.3.7 inPillar . . . 234
13.3.8 IRI local name . . . 235
13.3.9 IRI local type name . . . 235
13.3.10 levelInModel . . . 235
13.3.11 pillarInModel . . . 235
13.4 Named Individuals . . . 236
13.4.1 Business Pillar . . . 236
13.4.2 Data Pillar . . . 236
13.4.3 Test EKG/MM . . . 236
13.4.4 Level 1 . . . 237
13.4.5 Level 2 . . . 237
13.4.6 Level 3 . . . 238
13.4.7 Level 4 . . . 238
13.4.8 Level 5 . . . 238
13.4.9 Organization Pillar . . . 239
13.4.10 Technology Pillar . . . 239
14 Outcome Ontology . . . 240
14.1 Overview . . . 240
14.2 Classes . . . 240
14.2.1 Outcome . . . 240
14.2.2 Outcome Relationship . . . 241
14.2.3 Outcome Stereotype . . . 242
14.3 Properties . . . 242
14.3.1 contribution description . . . 242
14.3.2 Has Outcome . . . 243
14.3.3 priority . . . 243
14.3.4 stereotype . . . 243
14.3.5 target outcome . . . 244
14.3.6 updated at . . . 244
14.3.7 use case . . . 244
14.4 Named Individuals . . . 245
14.4.1 Benefit . . . 245
14.4.2 Goal . . . 245
14.4.3 Objective . . . 245
14.4.4 Theme . . . 246
14.4.5 Accountability . . . 246
14.4.6 Composable Business . . . 246
14.4.7 Contextual Truth . . . 247
14.4.8 Digital Assets . . . 247

14.4.9 Environmental Sustainability . . . 248
14.4.10 Fairness . . . 248
14.4.11 Manage Profitability . . . 248
14.4.12 Manage Risk & Compliance . . . 249
14.4.13 Openness . . . 249
14.4.14 Power of Abstraction . . . 250
14.4.15 Resilience . . . 250
14.4.16 Reusability . . . 250
14.4.17 Sustainability . . . 251
14.4.18 Transparency . . . 251
14.4.19 Unknown Outcome . . . 252
14.5 Shapes . . . 252
14.5.1 Outcome Relationship Shape . . . 252
15 Persona Ontology . . . 254
15.1 Overview . . . 254
15.2 Classes . . . 254
15.2.1 Accountability . . . 254
15.2.2 Persona . . . 254
15.3 Properties . . . 255
15.3.1 has accountability . . . 255
15.3.2 Is Sub Persona Of . . . 255
15.3.3 Persona key . . . 256
16 Raw Data Ontology . . . 257
16.1 Overview . . . 257
16.2 Classes . . . 257
16.2.1 Column . . . 257
16.2.2 CSV file . . . 257
16.2.3 File . . . 258
16.2.4 Null value . . . 259
16.2.5 Parquet file . . . 259
16.2.6 Row . . . 260
16.2.7 Source . . . 261
16.2.8 TSV file . . . 262
16.2.9 Value . . . 262
16.2.10 View . . . 263
16.2.11 XLSX file . . . 264
16.3 Properties . . . 265
16.3.1 25th percentile . . . 265
16.3.2 Median . . . 265
16.3.3 75th percentile . . . 265
16.3.4 Column number . . . 266
16.3.5 Conforming value count . . . 266
16.3.6 Value count . . . 266
16.3.7 File name . . . 267
16.3.8 For column . . . 267
16.3.9 For Predicate . . . 267
16.3.10 Formula . . . 268
16.3.11 Formula (R1C1) . . . 268
16.3.12 Most frequent value count . . . 268
16.3.13 Git blob . . . 269
16.3.14 Git commit . . . 269

16.3.15 Git modified . . . 269
16.3.16 Git path . . . 270
16.3.17 Git repository . . . 270
16.3.18 Has column . . . 270
16.3.19 Has View . . . 271
16.3.20 Inferred datatype . . . 271
16.3.21 Is column in view . . . 272
16.3.22 Is Row In View . . . 272
16.3.23 Is View In . . . 272
16.3.24 Legacy ID . . . 273
16.3.25 Legacy Id In Iri . . . 273
16.3.26 Literal value . . . 273
16.3.27 Local name . . . 274
16.3.28 Maximum . . . 274
16.3.29 Mean . . . 274
16.3.30 Minimum . . . 275
16.3.31 Null count . . . 275
16.3.32 Predicate . . . 275
16.3.33 Source text . . . 276
16.3.34 Standard deviation . . . 276
16.3.35 Most frequent value . . . 276
16.3.36 Distinct value count . . . 277
16.3.37 Value of . . . 277
16.3.38 View number . . . 278
16.3.39 View Row Number . . . 278
16.4 Shapes . . . 278
16.4.1 Column Location Shape . . . 278
16.4.2 Column Profile Shape . . . 279
16.4.3 Formula R1C1 Shape . . . 281
16.4.4 Formula Shape . . . 282
16.4.5 Git Origin Shape . . . 282
16.4.6 Row Location Shape . . . 283
16.4.7 Value Content Shape . . . 283
16.4.8 Value Location Shape . . . 284
16.4.9 View Number Shape . . . 285
17 Specification by Example Ontology . . . 286
17.1 Overview . . . 286
17.2 Classes . . . 286
17.2.1 Concept Value . . . 286
17.2.2 Graph . . . 286
17.2.3 Scenario . . . 287
17.2.4 State . . . 288
17.2.5 Story Request . . . 289
17.2.6 capture . . . 289
17.3 Properties . . . 289
17.3.1 Concept Value . . . 289
17.3.2 Dataset . . . 289
17.3.3 follows . . . 290
17.3.4 Given . . . 290
17.3.5 has Dataset . . . 290
17.3.6 has Graph . . . 291

17.3.7 Has Scenario . . . 291
17.3.8 ignore . . . 291
17.3.9 is depended on in state . . . 292
17.3.10 Is Given State Of . . . 292
17.3.11 is Graph in State . . . 292
17.3.12 Is Used In Graph . . . 293
17.3.13 Named Graph . . . 293
17.3.14 Story Parameters For Scenario . . . 293
17.3.15 Then . . . 294
17.3.16 Validates . . . 294
17.3.17 Value . . . 294
17.3.18 When . . . 294
17.4 Named Individuals . . . 295
17.4.1 any . . . 295
18 Story Ontology . . . 296
18.1 Overview . . . 296
18.1.1 Implementations and ownership . . . 296
18.1.2 Ownership and reuse . . . 297
18.1.3 Concept-usage model . . . 297
18.1.4 Input-concept behavior . . . 297
18.1.5 Migration from earlier generations . . . 298
18.2 Classes . . . 298
18.2.1 Agent Job Implementation . . . 298
18.2.2 API Proxy Implementation . . . 299
18.2.3 Named Statement Implementation . . . 299
18.2.4 Story . . . 300
18.2.5 Story Concept Usage . . . 300
18.2.6 Story Dependent . . . 301
18.2.7 Story Implementation . . . 302
18.2.8 Story Input . . . 302
18.2.9 Story Output . . . 303
18.2.10 Story Stereotype . . . 304
18.2.11 Workflow Implementation . . . 305
18.3 Properties . . . 305
18.3.1 contributes to . . . 305
18.3.2 expected usage role . . . 306
18.3.3 has concept usage . . . 306
18.3.4 has dependent concept . . . 307
18.3.5 has input concept . . . 307
18.3.6 has output concept . . . 307
18.3.7 has stereotype . . . 308
18.3.8 implementation of . . . 308
18.3.9 Implementation Type . . . 309
18.3.10 implemented by . . . 309
18.3.11 is deletion . . . 309
18.3.12 is mandatory . . . 310
18.3.13 Is Plural . . . 310
18.3.14 is server provided . . . 310
18.3.15 Is Singular . . . 311
18.3.16 Is Update . . . 311
18.3.17 not found on empty . . . 311

18.3.18 Played By . . . 312
18.3.19 provided by . . . 312
18.4 Named Individuals . . . 312
18.4.1 Query . . . 312
18.4.2 Transformation Rule . . . 313
18.4.3 Update . . . 313
18.4.4 Validation Rule . . . 314
18.5 Shapes . . . 314
18.5.1 Legacy Concept Usage Shape . . . 314
18.5.2 Legacy Implementation Shape . . . 315
18.5.3 Statement Position Usage Role Shape . . . 315
18.5.4 Story Classification Shape . . . 315
18.5.5 Story Concept Usage Shape . . . 316
18.5.6 Story Implementation Shape . . . 317
18.5.7 Story Input Shape . . . 317
18.5.8 Story Stereotype Shape . . . 317
18.5.9 Undeclared Concept Usage Shape . . . 318
19 SPARQL Story Implementation Ontology . . . 321
19.1 Overview . . . 321
19.2 Classes . . . 321
19.2.1 ASK . . . 321
19.2.2 CONSTRUCT . . . 322
19.2.3 DELETE . . . 322
19.2.4 DESCRIBE . . . 323
19.2.5 DROP . . . 323
19.2.6 RDFox SPARQL flavor . . . 324
19.2.7 SELECT . . . 324
19.2.8 SPARQL 1.0 . . . 325
19.2.9 SPARQL 1.1 . . . 325
19.2.10 SPARQL database product . . . 326
19.2.11 SPARQL database vendor . . . 326
19.2.12 SPARQL Flavor . . . 327
19.2.13 Stardog SPARQL flavor . . . 327
19.2.14 Statement Template . . . 328
19.2.15 SPARQL story implementation . . . 329
19.2.16 SPARQL query story implementation . . . 330
19.2.17 Story SPARQL Template Implementation . . . 330
19.2.18 SPARQL update story implementation . . . 331
19.2.19 Template Renderer . . . 332
19.2.20 UPDATE . . . 332
19.3 Properties . . . 333
19.3.1 File name . . . 333
19.3.2 Supports SPARQL Flavor; Supports SPARQL Flavour . . . 333
19.3.3 has template . . . 334
19.3.4 SPARQL statement . . . 334
19.3.5 supports template language . . . 334
19.3.6 template file name . . . 335
19.3.7 template text . . . 335
19.3.8 uses renderer . . . 335
19.4 Named Individuals . . . 336
19.4.1 Allegro Graph . . . 336

19.4.2 Amazon Web Services . . . 336
19.4.3 Anzo Graph . . . 336
19.4.4 Blazegraph . . . 337
19.4.5 Cambridge Semantics . . . 337
19.4.6 GraphDB . . . 337
19.4.7 Neptune . . . 338
19.4.8 Ontotext . . . 338
19.4.9 Oxford Semantic Technologies . . . 338
19.4.10 RD Fox . . . 338
19.4.11 Stardog . . . 339
19.4.12 Stardog Union . . . 339
19.4.13 Virtuoso . . . 339
19.5 Shapes . . . 340
19.5.1 Statement Template Shape . . . 340
19.5.2 Story SPARQL Template Implementation Shape . . . 340
19.5.3 Template Renderer Shape . . . 341
20 Story Service Ontology . . . 342
20.1 Overview . . . 342
20.1.1 Input-concept defaults . . . 342
20.1.2 Business concepts and technical bindings . . . 342
20.2 Classes . . . 343
20.2.1 Parameter . . . 343
20.2.2 Result . . . 343
20.2.3 SHACL Statement . . . 344
20.2.4 SPARQL Parameter . . . 344
20.2.5 SPARQL Results . . . 345
20.2.6 SPARQL Statement . . . 345
20.2.7 Statement . . . 346
20.2.8 Story Service . . . 347
20.3 Properties . . . 347
20.3.1 Has Parameter . . . 347
20.3.2 Is Service Parameter . . . 348
20.3.3 On . . . 348
20.3.4 Parameter Direction . . . 348
20.3.5 Parameter For . . . 349
20.3.6 Parameter Key . . . 349
20.3.7 Parameter Op . . . 350
20.3.8 Parameter Type . . . 350
20.3.9 should be supplied by . . . 350
20.3.10 Result For . . . 351
20.3.11 Result Key . . . 351
20.3.12 Result Type . . . 351
20.3.13 Should Be Generated By . . . 352
20.3.14 Sparql Query Name . . . 352
20.3.15 Sparql Statement File Name . . . 352
21 Use Case Ontology . . . 354
21.1 Overview . . . 354
21.1.1 Outcome contributions . . . 354
21.2 Classes . . . 354
21.2.1 Use Case . . . 354
21.2.2 Use Case Stereotype . . . 355

21.3 Properties . . . 355
21.3.1 has color; has colour . . . 355
21.3.2 has outcome . . . 356
21.3.3 has stereotype . . . 356
21.3.4 hex color . . . 356
21.3.5 Involves persona . . . 357
21.3.6 Is involved in . . . 357
21.3.7 is part of . . . 357
21.3.8 is used in . . . 358
21.3.9 Is used in use case . . . 358
21.3.10 outcome link . . . 359
21.3.11 owns concept-set . . . 359
21.3.12 screen layout index . . . 359
21.3.13 Used in . . . 360
21.3.14 Uses concept . . . 360
21.4 Named Individuals . . . 360
21.4.1 Application . . . 360
21.4.2 Business Capability . . . 361
21.4.3 Business Capability Area . . . 361
21.4.4 Data Domain . . . 361
21.4.5 Data Landscape . . . 362
21.4.6 Data Subdomain . . . 362
21.4.7 Data Transformation . . . 362
21.4.8 Enterprise . . . 363
21.4.9 Use Case . . . 363
A Documentation coverage . . . 365
B Document provenance . . . 366
Recent commits . . . 366
Diagram interpretation . . . 366

Abstract

An ontology family for enterprise knowledge graphs: concepts, use cases, stories, artifacts, governance, publication and lifecycles.

Status of This Document

This editor's draft is generated from the ontology sources and their registry. It is a documentation preview, not an approved standards release. Supplied source annotations are reproduced. Documentation gaps are reported explicitly; ontology imports are not fetched and this publication does not establish semantic correctness.

1 Family overview

1.1 EKGF Ontologies

Explore the 20 ontologies in this family. Select an ontology to read its vocabulary reference.

- **Annotation Ontology**

  Descriptive resource metadata: Markdown descriptions, subtitles, summaries, short titles, URL slugs, colors and sort keys. Text annotations apply to classes, properties and individuals; slug and sortKey are datatype properties for resource addressing and ordering.

- **Artifact Dependency Ontology**

  Generic dependency vocabulary for Semantic Modelling artifacts such as OWL ontologies, SHACL shapes graphs, SKOS taxonomies, and concept sets.

- **Artifact Governance Ontology**

  Who governs a semantic artifact and which family it belongs to. One vocabulary for OWL ontologies, SHACL shapes graphs, SKOS taxonomies and concept sets. Replaces the Ontology Ontology (https://ekgf.org/ontology/ontology#).

- **Artifact Publication Ontology**

  How semantic artifacts are published, either as a family or as explicitly selected artifact references: profiles configure generation, targets identify destinations, runs generate outputs, releases identify versions and approvals record authority decisions about exact releases. Family grouping is optional. The state of a run and of a release is described with the Lifecycle Ontology. SHACL shapes validate the scope and approval records.

- **Authorization Ontology**

  The EKGF Authorization Ontology.

  ```
  Modeling "Role Based Access Control" which is not the same
  as EKG-level entitlement calculation but could be part thereof.
  ```

- **Capability Ontology**

  A vocabulary for capabilities and capability domains, with example capabilities for sharing data and managing data products. The current vocabulary does not define the relationships assigning capabilities to domains.

- **Communication Ontology**

  Channel-neutral ontology for recipients, endpoints, audiences, memberships, consent and legal-basis evidence, campaign snapshots, communication attempts, and delivery outcomes.

- **Concept Ontology**

  Data element concepts, called data elements for short, describe logical data elements with all their various forms of physical manifestations, as business and technical terms, and their sightings. The scope includes individual data elements and compound data constructs such as whole classes and shapes. ManifestationOccurrence records each sighting: the artifacts in which a manifestation was found and, within a statement, the positions in which it appears.

- **DataOps Ontology**

  A vocabulary for DataOps pipelines, RDF load requests, their processing states, and the single-graph and multi-graph datasets produced during loading and assembly.

- **File System Ontology**

  A provider-neutral filesystem vocabulary: rooted views, directories, named directory entries, stable file series, immutable file versions whose content is a MIME-typed literal, symbolic links and service-style shortcuts, external provider objects, tombstones, declared capability profiles, and attachment assertions with explicit tracking modes. Names and parents belong to directory entries, never to files, so hard links, multiple parents, aliases, and provider shortcuts are representable without copying the target. Provider-specific API fields and credentials do not enter this ontology.

- **Lifecycle Ontology**

  Everything has a lifecycle: use cases, stories, concepts, personas, outcomes, data products and datasets. This ontology says what a lifecycle is, which lifecycle a thing follows and which state the thing is in. It depends on none of the ontologies that describe those things.

  A lifecycle has a name, states and permitted transitions, optionally grouped into phases. Its states and transitions form a directed graph; it does not require a separate RDF named graph. Multiple lifecycles can share an RDF graph, with inLifecycle expressing membership independently of graph placement. State order controls presentation, not which moves are permitted. There can be any number of lifecycles. This ontology defines none: the well-known lifecycles are a dataset (dataset/well-known-lifecycles.ttl), and anyone can define more. The registry uses a separate namespace for each lifecycle to distinguish shared state names such as Draft and Deprecated. This is an IRI naming convention, not a graph-storage requirement; custom lifecycles can use another IRI naming convention.

  Which lifecycles a thing follows is decided in this order:

  1. The complete set of lifecycles the thing names itself, with followsLifecycle.
  2. Otherwise, the union of lifecycles that apply to its classes (including superclasses), with appliesTo.

  A thing has one inState history head per effective lifecycle. Each head is a StateTransition occurrence whose currentState belongs to that lifecycle. priorStateTransition links earlier occurrences; the lifecycle is derived from each state, and optional provenance is recorded on the occurrence.

- **Maturity Model Ontology**

  A vocabulary for organizing maturity models into pillars, capability areas, capabilities, and levels, with an example EKGF model containing four pillars and five numbered levels.

- **Outcome Ontology**

  Business results, goals, benefits and themes, their optional classifications and priorities, and independently identified use-case contribution relationships. Outcome lifecycles are described with the Lifecycle Ontology.

- **Persona Ontology**

  A vocabulary for personas participating in EKG stories, their accountabilities, and sub-persona relationships. Persona identities are distinct from individual people.

- **Raw Data Ontology**

  The Raw Data Ontology is used by a "capture step" in an EKG DataOps pipeline to capture any data from any data source as RDF triples without mapping it to higher level ontologies. It is assumed that subsequent steps in a given EKG DataOps pipeline will transform the raw data to match with target ontologies.

- **Specification by Example Ontology**

A vocabulary for executable story scenarios, including initial graph and dataset states, story requests, expected results, chained scenarios, and captured run-time values.

- **Story Ontology**

  Stories, their extensible primary-purpose stereotypes, story-owned input, output, and dependent concept usages, execution requirements, and implementation ownership.

- **SPARQL Story Implementation Ontology**

  The SPARQL Story Implementation Ontology defines story implementations using executable SPARQL queries or updates, or explicitly declared template documents and renderers that generate SPARQL.

  ```
  The Story ontology itself is: https://ekgf.org/ontology/story#
  ```

- **Story Service Ontology**

  The Story Service Ontology supports the Story Service which is a program that can translate Use Case Stories to executable APIs.

- **Use Case Ontology**

  The EKGF Use Case Ontology, based on the EKG/Method as documented on https://method.ekgf.org. It describes the use case tree and the stereotype of each use case. The lifecycle of a use case is described with the Lifecycle Ontology.

2 Annotation Ontology

2.1 Overview

Annotation Ontology

Identifier	Value
Ontology IRI / Namespace	`https://ekgf.org/ontology/annotation#`
Prefix	`ekg-annotation`

Descriptive resource metadata: Markdown descriptions, subtitles, summaries, short titles, URL slugs, colors and sort keys. Text annotations apply to classes, properties and individuals; slug and sortKey are datatype properties for resource addressing and ordering.

2.2 Properties

2.2.1 Hexadecimal RGB color

An RGB color code in hex

`ekg-annotation:hexColor`	
`https://ekgf.org/ontology/annotation#hexColor`	
RDF predicate	**Value**
`rdf:type`	`owl:AnnotationProperty`
`rdfs:label`	Hexadecimal RGB color

2.2.2 Markdown description

Use this annotation on an ontology, class, property, or individual for explanations that benefit from Markdown: scope, examples, modeling decisions, ownership rules, and application behavior. Use an English language tag for the English explanation. Keep a short definition in `rdfs:comment` where useful.

Markdown can include headings, lists, tables, fenced code examples, and links. Consumers render this description when present and use `rdfs:comment` when it is absent.

An explanation is an annotation. Describing a subclass relationship, a property range, a SHACL constraint, or an application default does not implement it. Identify agreed changes awaiting implementation explicitly, so readers can distinguish the recorded decision from the current axioms.

`ekg-annotation:markdownDescription`	
`https://ekgf.org/ontology/annotation#markdownDescription`	
RDF predicate	**Value**
`rdf:type`	`owl:AnnotationProperty`
`rdfs:label`	Markdown description

2.2.3 short title

A short title is a real, authored name, not the result of trimming a title. For example, "Commons Identifiers Ontology" can have the short title "Identifiers Ontology" for use under the Commons family heading. A viewer chooses the short title where the context makes the full title redundant, and the full label elsewhere.

Each value carries a language tag, with at most one short title per language, so translations work like any other label. Resources without a short title use their normal label. A tool that knows a naming convention can generate short titles, and `ShortTitleShape` validates them like any other data.

ekg-annotation:shortTitle	
https://ekgf.org/ontology/annotation#shortTitle	
RDF predicate	**Value**
rdf:type	owl:AnnotationProperty
rdfs:label	short title

2.2.4 slug

A slug is a readable URL name, such as `enterprise` or `working-draft`. It does not replace the resource's stable IRI or registered UUID identity. Different contexts can reuse a slug: states in different lifecycles can both be named `deprecated`. This property declares no global uniqueness rule, cardinality restriction or automatic slug generation.

The property is an `owl:DatatypeProperty` with string values. Its placement in this ontology does not turn it into an OWL annotation property. The existing Markdown, subtitle and summary predicates retain their annotation semantics.

ekg-annotation:slug	
https://ekgf.org/ontology/annotation#slug	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:label	slug
rdfs:range	xsd:string

2.2.5 Sort key

A sort key places a resource among its siblings wherever a user interface or report lists them: use cases in a tree, concepts in a vocabulary, tabs on a page. Sorting compares the key where a resource has one and its label where it does not, so keys compete with the labels themselves.

A key is a hint, not a rank. Siblings often come from datasets and sources that know nothing of each other, so they cannot share a numbering. Each source picks an arbitrary string that lands its resource roughly in the right place: "aaa" near the start, "zzz" near the end. Two resources with the same key sit side by side and sort by their labels instead of colliding.

```
@prefix annotation: <https://ekgf.org/ontology/annotation#> .
@prefix ex: <https://example.org/> .

ex:overview annotation:sortKey "aaa" .
ex:shapes annotation:sortKey "zzz1" .
ex:provenance annotation:sortKey "zzz2" .
```

The property has no domain, so any resource can carry one. Its placement in this ontology does not make it an OWL annotation property; like `slug`, it is a datatype property with string values.

ekg-annotation:sortKey	
https://ekgf.org/ontology/annotation#sortKey	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:label	Sort key
rdfs:range	xsd:string

2.2.6 subtitle

One line shown under the label of the resource.

ekg-annotation:subtitle	
https://ekgf.org/ontology/annotation#subtitle	
RDF predicate	**Value**
rdf:type	owl:AnnotationProperty
rdfs:label	subtitle

2.2.7 summary

A short summary of the resource, in plain text.

ekg-annotation:summary	
https://ekgf.org/ontology/annotation#summary	
RDF predicate	**Value**
rdf:type	owl:AnnotationProperty
rdfs:label	summary

2.3 Shapes

2.3.1 Short Title Shape

Requires each short title to be a nonempty language-tagged literal, with at most one per language.

ekg-annotation:ShortTitleShape	
https://ekgf.org/ontology/annotation#ShortTitleShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Short Title Shape
sh:property	(sh:datatype: rdf:langString; sh:message: A short title is a nonempty language-tagged string, with at most one per language.; sh:minLength: 1; sh:path: ekg-annotation:shortTitle; sh:uniqueLang: true)
sh:targetSubjectsOf	ekg-annotation:shortTitle

3 Artifact Dependency Ontology

3.1 Overview

Artifact Dependency Ontology

Identifier	Value
Ontology IRI / Namespace	`https://ekgf.org/ontology/artifact-dependency#`
Prefix	`ekg-artdep`

Separates dependency facts, evidence, and workspace or branch resolution status for Semantic Modelling dependency management.

3.2 Classes

3.2.1 Dependency

A reified dependency edge from one semantic artifact to another resource or artifact.

`ekg-artdep:Dependency`	
`https://ekgf.org/ontology/artifact-dependency#Dependency`	
RDF predicate	**Value**
`rdf:type`	`owl:Class`
`rdfs:label`	Dependency

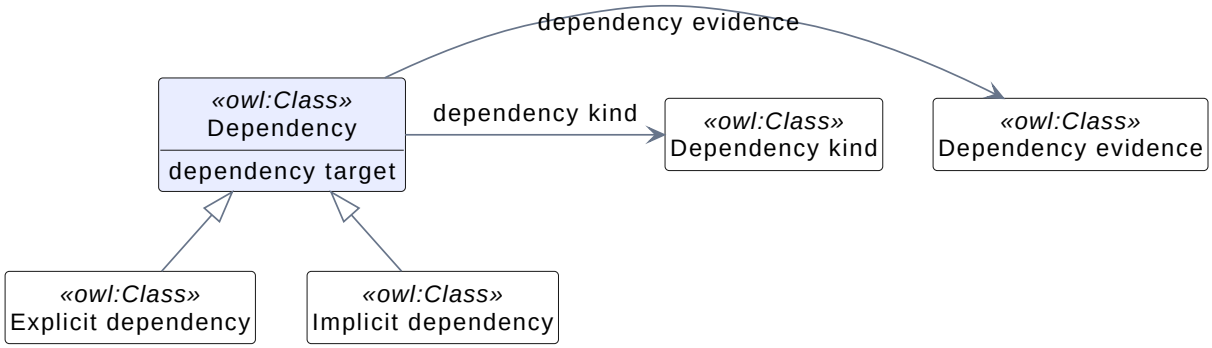


Diagram note

3.2.2 Dependency evidence

The concrete triple, graph, namespace observation, or other fact that supports a dependency edge.

`ekg-artdep:DependencyEvidence`	
`https://ekgf.org/ontology/artifact-dependency#DependencyEvidence`	
RDF predicate	**Value**
`rdf:type`	`owl:Class`
`rdfs:label`	Dependency evidence

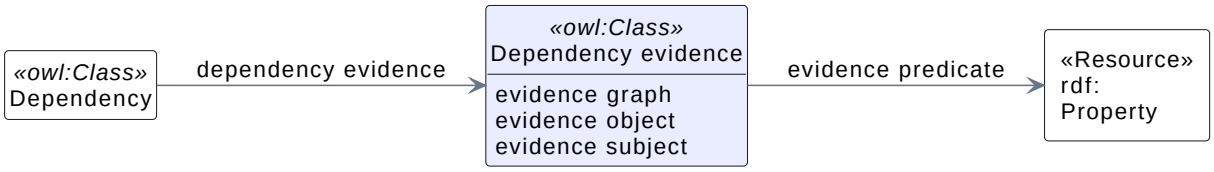


Diagram note

3.2.3 Dependency kind

Controlled values describing why the dependency exists.

ekg-artdep:DependencyKind	
https://ekgf.org/ontology/artifact-dependency#DependencyKind	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Dependency kind

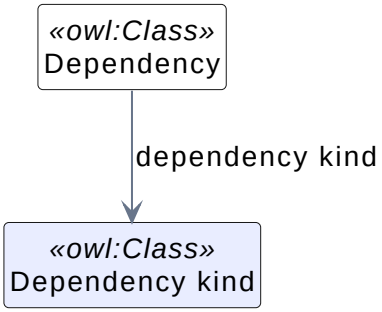


Diagram note

3.2.4 Dependency resolution

Workspace and branch scoped resolution state for a dependency target.

ekg-artdep:DependencyResolution	
https://ekgf.org/ontology/artifact-dependency#DependencyResolution	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Dependency resolution

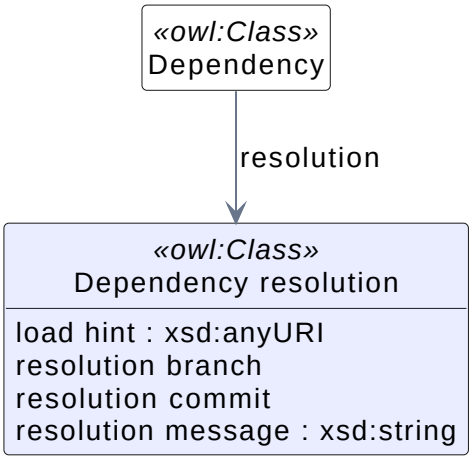


Diagram note

3.2.5 Explicit dependency

A dependency directly declared by a source triple such as owl:imports, sh:shapesGraph, or artdep:dependsOn.

ekg-artdep:ExplicitDependency	
https://ekgf.org/ontology/artifact-dependency#ExplicitDependency	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Explicit dependency
rdfs:subClassOf	ekg-artdep:Dependency

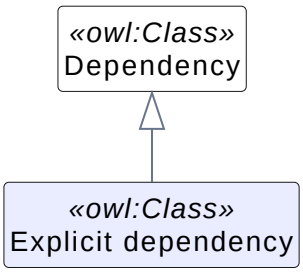


Diagram note

3.2.6 Implicit dependency

A dependency inferred from referenced terms, namespaces, validation constraints, or profile requirements rather than an explicit import declaration.

ekg-artdep:ImplicitDependency	
https://ekgf.org/ontology/artifact-dependency#ImplicitDependency	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Implicit dependency
rdfs:subClassOf	ekg-artdep:Dependency

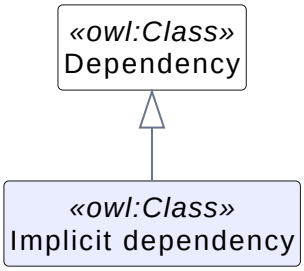


Diagram note

3.2.7 Ontology load request

A request for a worker node to load an ontology family that would satisfy one or more unloaded dependency targets. How far the request has come is its lifecycle state; by default a request follows the execution lifecycle.

ekg-artdep:OntologyLoadRequest	
https://ekgf.org/ontology/artifact-dependency#OntologyLoadRequest	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Ontology load request

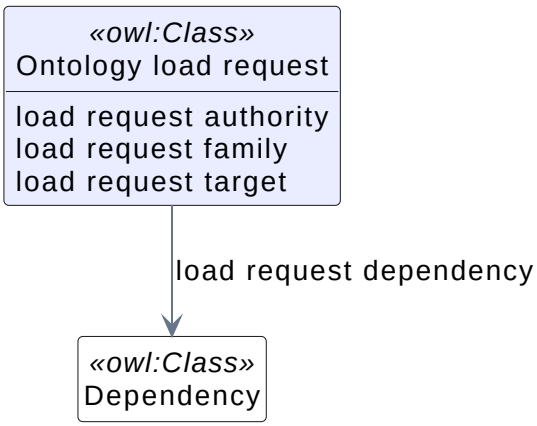


Diagram note

3.2.8 Resolution status

Controlled values describing whether and how a dependency target resolved in a workspace context.

ekg-artdep:ResolutionStatus	
https://ekgf.org/ontology/artifact-dependency#ResolutionStatus	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Resolution status

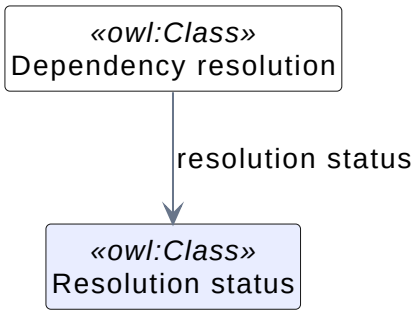


Diagram note

3.2.9 Semantic artifact

A Semantic Modelling artifact that can depend on another artifact, including OWL ontologies, SHACL shapes graphs, SKOS taxonomies, and concept sets.

ekg-artdep:SemanticArtifact	
https://ekgf.org/ontology/artifact-dependency#SemanticArtifact	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Semantic artifact

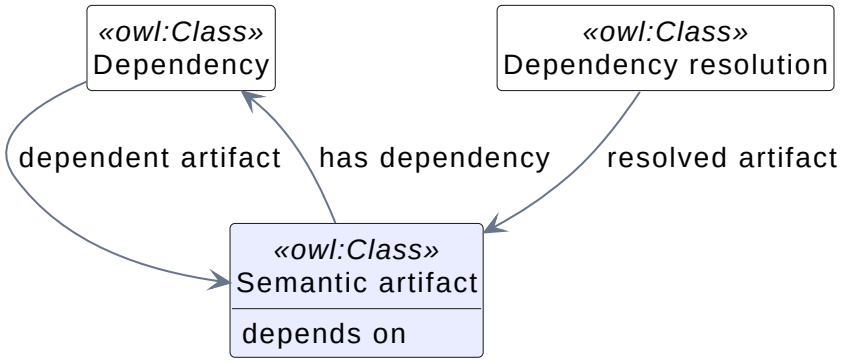


Diagram note

3.3 Properties

3.3.1 dependency evidence

Links a dependency edge to one supporting evidence record.

ekg-artdep:dependencyEvidence	
https://ekgf.org/ontology/artifact-dependency#dependencyEvidence	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-artdep:Dependency
rdfs:label	dependency evidence
rdfs:range	ekg-artdep:DependencyEvidence

3.3.2 dependency kind

Classifies why the dependency exists.

ekg-artdep:dependencyKind	
https://ekgf.org/ontology/artifact-dependency#dependencyKind	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-artdep:Dependency
rdfs:label	dependency kind
rdfs:range	ekg-artdep:DependencyKind

3.3.3 dependency target

The target resource named by the dependency edge, loaded or not.

ekg-artdep:dependencyTarget	
https://ekgf.org/ontology/artifact-dependency#dependencyTarget	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-artdep:Dependency
rdfs:label	dependency target
rdfs:range	rdfs:Resource

3.3.4 dependent artifact

The artifact that owns the dependency edge.

ekg-artdep:dependentArtifact	
https://ekgf.org/ontology/artifact-dependency#dependentArtifact	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-artdep:Dependency

ekg-artdep:dependentArtifact	
https://ekgf.org/ontology/artifact-dependency#dependentArtifact	
RDF predicate	**Value**
rdfs:label	dependent artifact
rdfs:range	ekg-artdep:SemanticArtifact

3.3.5 depends on

Direct dependency projection from an artifact to the dependency target. Reified edges use artdep:Dependency when evidence or resolution state is needed.

ekg-artdep:dependsOn	
https://ekgf.org/ontology/artifact-dependency#dependsOn	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-artdep:SemanticArtifact
rdfs:label	depends on
rdfs:range	rdfs:Resource

3.3.6 evidence graph

Named graph containing the source triple or observation that supports the dependency.

ekg-artdep:evidenceGraph	
https://ekgf.org/ontology/artifact-dependency#evidenceGraph	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-artdep:DependencyEvidence
rdfs:label	evidence graph
rdfs:range	rdfs:Resource

3.3.7 evidence object

IRI object of the source triple or observation that supports the dependency.

ekg-artdep:evidenceObject	
https://ekgf.org/ontology/artifact-dependency#evidenceObject	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-artdep:DependencyEvidence
rdfs:label	evidence object
rdfs:range	rdfs:Resource

3.3.8 evidence predicate

Predicate of the source triple or observation that supports the dependency. It is an annotation property because its values are properties, which are not individuals.

ekg-artdep:evidencePredicate	
https://ekgf.org/ontology/artifact-dependency#evidencePredicate	
RDF predicate	**Value**
rdf:type	owl:AnnotationProperty
rdfs:domain	ekg-artdep:DependencyEvidence
rdfs:label	evidence predicate
rdfs:range	rdf:Property

3.3.9 evidence subject

Subject of the source triple or observation that supports the dependency.

ekg-artdep:evidenceSubject	
https://ekgf.org/ontology/artifact-dependency#evidenceSubject	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-artdep:DependencyEvidence
rdfs:label	evidence subject
rdfs:range	rdfs:Resource

3.3.10 has dependency

Links a semantic artifact to a reified dependency edge.

ekg-artdep:hasDependency	
https://ekgf.org/ontology/artifact-dependency#hasDependency	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-artdep:SemanticArtifact
rdfs:label	has dependency
rdfs:range	ekg-artdep:Dependency

3.3.11 load hint

Optional URI hint for loading or locating an unresolved dependency target.

ekg-artdep:loadHint	
https://ekgf.org/ontology/artifact-dependency#loadHint	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-artdep:DependencyResolution
rdfs:label	load hint
rdfs:range	xsd:anyURI

3.3.12 load request authority

Authority governing the artifact family named by the load request.

ekg-artdep:loadRequestAuthority	
https://ekgf.org/ontology/artifact-dependency#loadRequestAuthority	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-artdep:OntologyLoadRequest
rdfs:label	load request authority
rdfs:range	rdfs:Resource

3.3.13 load request dependency

Links a load request to a dependency edge that the requested family load is expected to satisfy.

ekg-artdep:loadRequestDependency	
https://ekgf.org/ontology/artifact-dependency#loadRequestDependency	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-artdep:OntologyLoadRequest
rdfs:label	load request dependency
rdfs:range	ekg-artdep:Dependency

3.3.14 load request family

Artifact family that should be loaded to satisfy the unresolved dependency targets.

ekg-artdep:loadRequestFamily	
https://ekgf.org/ontology/artifact-dependency#loadRequestFamily	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-artdep:OntologyLoadRequest
rdfs:label	load request family

ekg-artdep:loadRequestFamily	
https://ekgf.org/ontology/artifact-dependency#loadRequestFamily	
RDF predicate	**Value**
rdfs:range	rdfs:Resource

3.3.15 load request target

Dependency target IRI that caused this load request.

ekg-artdep:loadRequestTarget	
https://ekgf.org/ontology/artifact-dependency#loadRequestTarget	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-artdep:OntologyLoadRequest
rdfs:label	load request target
rdfs:range	rdfs:Resource

3.3.16 referenced namespace

Namespace IRI observed in an implicit dependency, usually derived from a referenced term IRI.

ekg-artdep:referencedNamespace	
https://ekgf.org/ontology/artifact-dependency#referencedNamespace	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-artdep:DependencyEvidence
rdfs:label	referenced namespace
rdfs:range	rdfs:Resource

3.3.17 resolution

Links a dependency edge to its workspace or branch scoped resolution state.

ekg-artdep:resolution	
https://ekgf.org/ontology/artifact-dependency#resolution	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-artdep:Dependency
rdfs:label	resolution
rdfs:range	ekg-artdep:DependencyResolution

3.3.18 resolution branch

Branch context in which the dependency resolution was computed.

ekg-artdep:resolutionBranch	
https://ekgf.org/ontology/artifact-dependency#resolutionBranch	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-artdep:DependencyResolution
rdfs:label	resolution branch
rdfs:range	rdfs:Resource

3.3.19 resolution commit

Commit context in which the dependency resolution was computed, when a resolution is pinned to a specific commit.

ekg-artdep:resolutionCommit	
https://ekgf.org/ontology/artifact-dependency#resolutionCommit	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-artdep:DependencyResolution
rdfs:label	resolution commit
rdfs:range	rdfs:Resource

3.3.20 resolution message

Human-readable diagnostic explaining a non-loaded or ambiguous resolution.

ekg-artdep:resolutionMessage	
https://ekgf.org/ontology/artifact-dependency#resolutionMessage	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-artdep:DependencyResolution
rdfs:label	resolution message
rdfs:range	xsd:string

3.3.21 resolution status

The loaded, unloaded, incomplete, ambiguous, external, or unsupported state of a dependency resolution.

ekg-artdep:resolutionStatus	
https://ekgf.org/ontology/artifact-dependency#resolutionStatus	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty

ekg-artdep:resolutionStatus	
https://ekgf.org/ontology/artifact-dependency#resolutionStatus	
RDF predicate	**Value**
rdfs:domain	ekg-artdep:DependencyResolution
rdfs:label	resolution status
rdfs:range	ekg-artdep:ResolutionStatus

3.3.22 resolution workspace

Workspace context in which the dependency resolution was computed.

ekg-artdep:resolutionWorkspace	
https://ekgf.org/ontology/artifact-dependency#resolutionWorkspace	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-artdep:DependencyResolution
rdfs:label	resolution workspace
rdfs:range	rdfs:Resource

3.3.23 resolved artifact

The loaded artifact that satisfies the dependency target, when resolution status is Loaded.

ekg-artdep:resolvedArtifact	
https://ekgf.org/ontology/artifact-dependency#resolvedArtifact	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-artdep:DependencyResolution
rdfs:label	resolved artifact
rdfs:range	ekg-artdep:SemanticArtifact

3.4 Named Individuals

3.4.1 Ambiguous

Instance of: ekg-artdep:ResolutionStatus

The dependency target matches more than one candidate artifact in the current workspace context.

ekg-artdep:Ambiguous	
https://ekgf.org/ontology/artifact-dependency#Ambiguous	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-artdep:ResolutionStatus

ekg-artdep:Ambiguous	
https://ekgf.org/ontology/artifact-dependency#Ambiguous	
RDF predicate	**Value**
rdfs:label	Ambiguous

3.4.2 Declared import dependency

Instance of: ekg-artdep:DependencyKind

A dependency declared by an import-like predicate such as owl:imports.

ekg-artdep:DeclaredImportDependency	
https://ekgf.org/ontology/artifact-dependency#DeclaredImportDependency	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-artdep:DependencyKind
rdfs:label	Declared import dependency

3.4.3 External

Instance of: ekg-artdep:ResolutionStatus

The dependency target is intentionally outside the workspace and does not need to be loaded locally.

ekg-artdep:External	
https://ekgf.org/ontology/artifact-dependency#External	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-artdep:ResolutionStatus
rdfs:label	External

3.4.4 Failed

Instance of: ekg-artdep:ResolutionStatus

Dependency resolution was attempted and failed in the current workspace context.

ekg-artdep:Failed	
https://ekgf.org/ontology/artifact-dependency#Failed	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-artdep:ResolutionStatus
rdfs:label	Failed

3.4.5 Incomplete

Instance of: ekg-artdep:ResolutionStatus

The dependency target is not sufficiently described to load or resolve automatically.

ekg-artdep:Incomplete	
https://ekgf.org/ontology/artifact-dependency#Incomplete	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-artdep:ResolutionStatus
rdfs:label	Incomplete

3.4.6 Loaded

Instance of: ekg-artdep:ResolutionStatus

The dependency target resolves to an artifact loaded in the current workspace context.

ekg-artdep:Loaded	
https://ekgf.org/ontology/artifact-dependency#Loaded	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-artdep:ResolutionStatus
rdfs:label	Loaded

3.4.7 Namespace reference dependency

Instance of: ekg-artdep:DependencyKind

An implicit dependency derived from terms that share an external namespace.

ekg-artdep:NamespaceReferenceDependency	
https://ekgf.org/ontology/artifact-dependency#NamespaceReferenceDependency	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-artdep:DependencyKind
rdfs:label	Namespace reference dependency

3.4.8 Profile dependency

Instance of: ekg-artdep:DependencyKind

A dependency on a standards or implementation profile.

ekg-artdep:ProfileDependency	
https://ekgf.org/ontology/artifact-dependency#ProfileDependency	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual

ekg-artdep:ProfileDependency	
https://ekgf.org/ontology/artifact-dependency#ProfileDependency	
RDF predicate	**Value**
rdf:type	ekg-artdep:DependencyKind
rdfs:label	Profile dependency

3.4.9 Shape graph dependency

Instance of: ekg-artdep:DependencyKind

A dependency on a SHACL shapes graph used for validation or schema reuse.

ekg-artdep:ShapeGraphDependency	
https://ekgf.org/ontology/artifact-dependency#ShapeGraphDependency	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-artdep:DependencyKind
rdfs:label	Shape graph dependency

3.4.10 Term reference dependency

Instance of: ekg-artdep:DependencyKind

An implicit dependency derived from a reference to a specific external class, property, shape, concept, or datatype.

ekg-artdep:TermReferenceDependency	
https://ekgf.org/ontology/artifact-dependency#TermReferenceDependency	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-artdep:DependencyKind
rdfs:label	Term reference dependency

3.4.11 Unloaded

Instance of: ekg-artdep:ResolutionStatus

The dependency target is known but no matching artifact is loaded in the current workspace context.

ekg-artdep:Unloaded	
https://ekgf.org/ontology/artifact-dependency#Unloaded	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-artdep:ResolutionStatus
rdfs:label	Unloaded

3.4.12 Unsupported

Instance of: ekg-artdep:ResolutionStatus

The dependency target uses a form that cannot be resolved yet.

ekg-artdep:Unsupported	
https://ekgf.org/ontology/artifact-dependency#Unsupported	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-artdep:ResolutionStatus
rdfs:label	Unsupported

3.4.13 Validation dependency

Instance of: ekg-artdep:DependencyKind

A dependency required to validate an artifact or data graph correctly.

ekg-artdep:ValidationDependency	
https://ekgf.org/ontology/artifact-dependency#ValidationDependency	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-artdep:DependencyKind
rdfs:label	Validation dependency

4 Artifact Governance Ontology

4.1 Overview

Artifact Governance Ontology

Identifier	Value
Ontology IRI / Namespace	`https://ekgf.org/ontology/artifact-governance#`
Prefix	`ekg-artgov`

Who governs a semantic artifact and which family it belongs to. One vocabulary for OWL ontologies, SHACL shapes graphs, SKOS taxonomies and concept sets. Replaces the Ontology Ontology (https://ekgf.org/ontology/ontology#).

4.2 Classes

4.2.1 Authority

A role responsible for governing semantic artifacts or their publication namespaces. The party playing the role is linked with cmns-rlcmp:isPlayedBy and has its own identity. An authority and an organization playing it are distinct individuals, following the Commons separation of agents and roles. An authority is assigned a random UUIDv4 URN once in the authority registry. Its identity persists when names, hosts, URLs or the party playing the role change. Hosts and URLs are metadata and may have multiple values; they do not determine identity.

`ekg-artgov:Authority`	
`https://ekgf.org/ontology/artifact-governance#Authority`	
RDF predicate	**Value**
`rdf:type`	`owl:Class`
`rdfs:label`	Authority
`rdfs:subClassOf`	`cmns-pts:PartyRole`

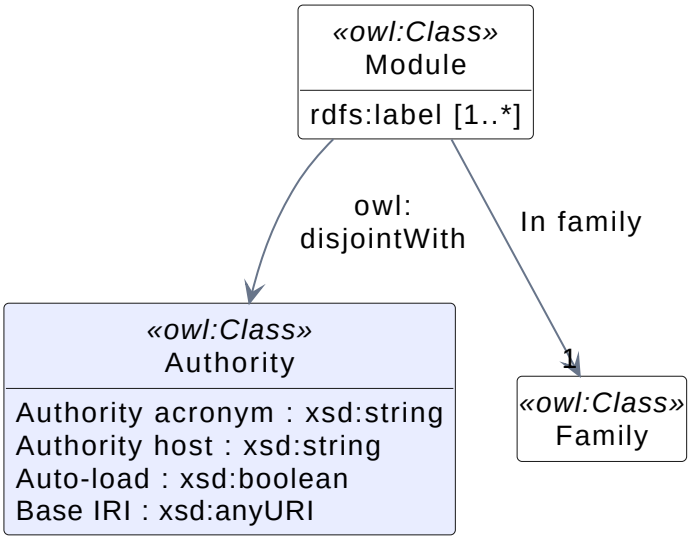


Diagram note

4.2.2 Authority publishing role

Controlled role values describing how an authority participates in publishing a family.

ekg-artgov:AuthorityPublishingRole	
https://ekgf.org/ontology/artifact-governance#AuthorityPublishingRole	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Authority publishing role

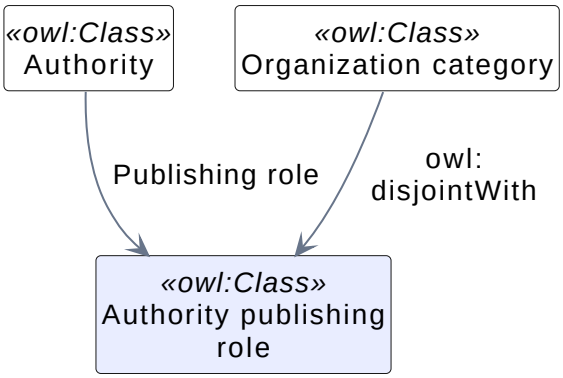


Diagram note

4.2.3 Family

An optional, enduring collection of related artifacts governed by one or more authorities, such as FIBO. A family spans artifact types: its ontologies, shapes graphs and taxonomies are members of the same family. Authorities can govern artifacts directly without organizing them into families. When used for publication, the family groups the artifacts across releases; each version and its approvals belong to a distinct standards release.

ekg-artgov:Family	
https://ekgf.org/ontology/artifact-governance#Family	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Family

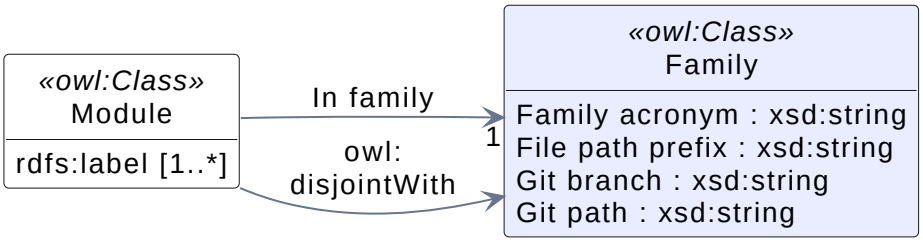


Diagram note

4.2.4 Module

Large families organise their artifacts into modules: FIBO has domains such as FBC, SEC and FND, and some of those contain further modules. A module belongs to exactly one family through `inFamily`. Members join it explicitly through `inModule`: an ontology, a registry reference, or a nested module.

Membership is stated, never derived. IRI path segments, file names and directories do not create modules, because module boundaries do not reliably follow them. A harvester that knows a family's conventions can generate the module statements, which validation then checks like any other data.

A family without modules simply declares none. Whether a viewer shows the module level is a presentation choice of the viewer or publication profile.

ekg-artgov:Module	
https://ekgf.org/ontology/artifact-governance#Module	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Module
owl:disjointWith	ekg-artgov:Authority
owl:disjointWith	ekg-artgov:Family
owl:disjointWith	ekg-artgov:Reference

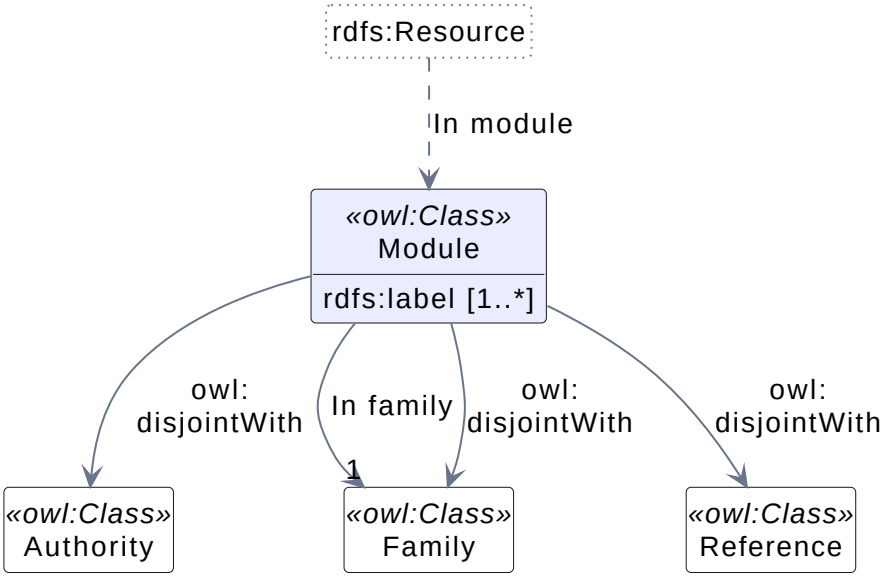


Diagram note

4.2.5 Organization category

A classifier describing the kind of organization involved in artifact governance, such as a standards body, consortium, project or community. An organization may have several categories. Categories are distinct from publishing responsibilities, community participation roles and legal or organizational structure.

ekg-artgov:OrganizationCategory	
https://ekgf.org/ontology/artifact-governance#OrganizationCategory	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Organization category
rdfs:subClassOf	cmns-cls:Classifier
owl:disjointWith	ekg-artgov:AuthorityPublishingRole
owl:disjointWith	ekg-artgov:StandardsCommunityRole

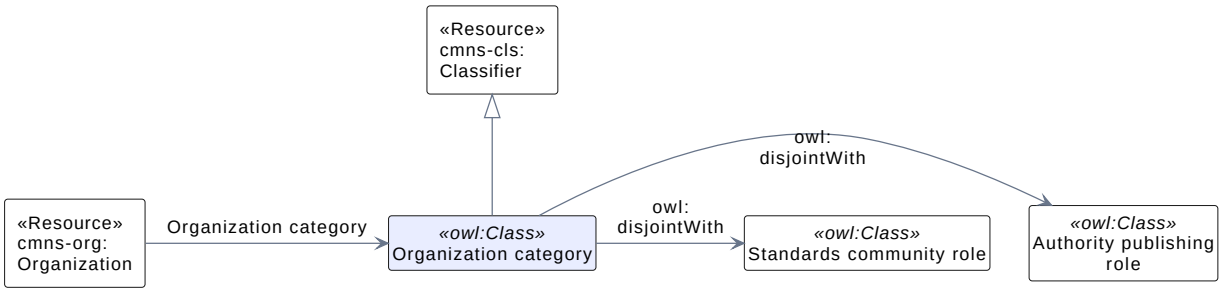


Diagram note

4.2.6 Reference

Stands for one artifact. It gives the artifact a stable identity and slug even when the artifact itself is not loaded, or is known under several IRIs that differ only in a trailing hash or slash.

ekg-artgov:Reference	
https://ekgf.org/ontology/artifact-governance#Reference	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Reference

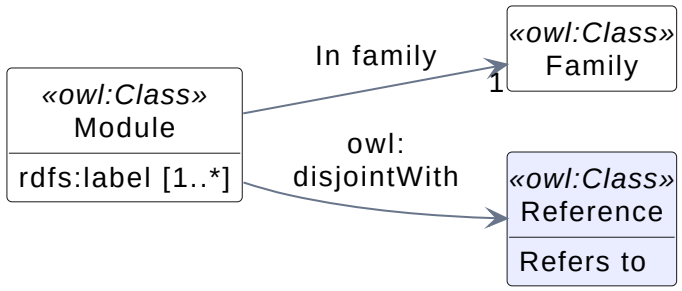


Diagram note

4.2.7 Standards community role

Controlled role values for people participating in the standards community behind an authority.

ekg-artgov:StandardsCommunityRole	
https://ekgf.org/ontology/artifact-governance#StandardsCommunityRole	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Standards community role

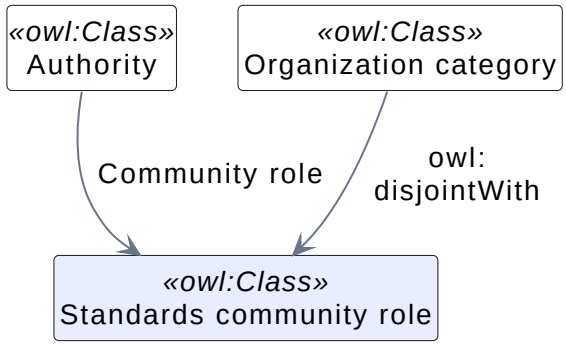


Diagram note

4.3 Properties

4.3.1 Authority acronym

Uppercase acronym identifying the authority, such as W3C, OMG, EDMA or EKGF.

ekg-artgov:authorityAcronym	
https://ekgf.org/ontology/artifact-governance#authorityAcronym	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-artgov:Authority
rdfs:label	Authority acronym
rdfs:range	xsd:string

4.3.2 Authority host

A host name used in the IRIs of the authority's artifacts, without protocol, such as www.w3.org or spec.edmcouncil.org. An authority may have multiple hosts, including historical hosts whose artifact IRIs remain in use. Adding, removing or changing a host does not change the authority's identity. A shared host does not imply shared governance; hosts are metadata, not identity keys.

ekg-artgov:authorityHost	
https://ekgf.org/ontology/artifact-governance#authorityHost	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-artgov:Authority
rdfs:label	Authority host
rdfs:range	xsd:string

4.3.3 Auto-load

When false, a loader does not fetch the families of this authority when their artifacts appear as unresolved owl:imports targets.

ekg-artgov:autoLoad	
https://ekgf.org/ontology/artifact-governance#autoLoad	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-artgov:Authority
rdfs:label	Auto-load
rdfs:range	xsd:boolean

4.3.4 Base IRI

A base IRI of artifacts published by an authority. An authority may carry multiple base IRIs, including different hosts, schemes and historical locations. These values do not determine the authority's identity.

ekg-artgov:baseIri	
https://ekgf.org/ontology/artifact-governance#baseIri	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-artgov:Authority
rdfs:label	Base IRI
rdfs:range	xsd:anyURI

4.3.5 Community organization

Links an authority to the organization that hosts its standards community. The party exercising the authority is stated separately with cmns-rlcmp:isPlayedBy.

ekg-artgov:communityOrganization	
https://ekgf.org/ontology/artifact-governance#communityOrganization	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-artgov:Authority
rdfs:label	Community organization
rdfs:range	cmns-org:Organization

4.3.6 Community role

Declares a role used by the standards community behind an authority.

ekg-artgov:communityRole	
https://ekgf.org/ontology/artifact-governance#communityRole	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-artgov:Authority
rdfs:label	Community role
rdfs:range	ekg-artgov:StandardsCommunityRole

4.3.7 Family acronym

Uppercase acronym identifying the family, such as FIBO or CDMC.

ekg-artgov:familyAcronym	
https://ekgf.org/ontology/artifact-governance#familyAcronym	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-artgov:Family

ekg-artgov:familyAcronym	
https://ekgf.org/ontology/artifact-governance#familyAcronym	
RDF predicate	**Value**
rdfs:label	Family acronym
rdfs:range	xsd:string

4.3.8 File path prefix

A portable, case-sensitive literal prefix matched at the beginning of any segment of the actual source file path. For example, fibo/ matches a directory and ekgf- matches a filename prefix. Values use forward slashes, are relative and nonempty, and contain no scheme, drive, empty segment or dot segment. A family may have several. Matching normalizes source path separators to forward slashes and removes current-directory and empty segments; callers must resolve parent-directory segments first. The actual path is supplied independently of the named-graph IRI. The longest matching file path prefix wins within this method; ties across families or disagreement with other methods are errors unless explicit inFamily assignments take precedence.

ekg-artgov:filePathPrefix	
https://ekgf.org/ontology/artifact-governance#filePathPrefix	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-artgov:Family
rdfs:label	File path prefix
rdfs:range	xsd:string

4.3.9 Git branch

The git branch or tag to clone.

ekg-artgov:gitBranch	
https://ekgf.org/ontology/artifact-governance#gitBranch	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-artgov:Family
rdfs:label	Git branch
rdfs:range	xsd:string

4.3.10 Git path

Directory inside the git repository to scan for RDF files. The repository root when absent.

ekg-artgov:gitPath	
https://ekgf.org/ontology/artifact-governance#gitPath	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-artgov:Family
rdfs:label	Git path
rdfs:range	xsd:string

4.3.11 Git repository URL

HTTPS URL of the git repository that holds the source files of the family. A loader clones it and loads the RDF files it finds.

ekg-artgov:gitRepositoryUrl	
https://ekgf.org/ontology/artifact-governance#gitRepositoryUrl	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-artgov:Family
rdfs:label	Git repository URL
rdfs:range	xsd:anyURI

4.3.12 Governed by

Relates an artifact, a reference or a family to an authority that governs it. It has no domain because it applies to every artifact type.

ekg-artgov:governedBy	
https://ekgf.org/ontology/artifact-governance#governedBy	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:label	Governed by
rdfs:range	ekg-artgov:Authority

4.3.13 Graph IRI prefix

A case-sensitive literal prefix of the complete named-graph IRI, such as s3://standards/fibo/. It is an absolute IRI prefix typed xsd:anyURI. A family may have several. Relative file path prefixes belong in filePathPrefix; a graph name need not reveal its source file path. The longest matching graph IRI prefix wins within this method; ties across families or disagreement with other methods are errors unless explicit inFamily assignments take precedence.

ekg-artgov:graphIriPrefix	
https://ekgf.org/ontology/artifact-governance#graphIriPrefix	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-artgov:Family
rdfs:label	Graph IRI prefix
rdfs:range	xsd:anyURI

4.3.14 Historical name

A former public name of an authority or organization.

ekg-artgov:historicalName	
https://ekgf.org/ontology/artifact-governance#historicalName	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:label	Historical name
rdfs:range	xsd:string

4.3.15 In family

Explicitly assigns an artifact or a reference to a family. It has no domain because it applies to every artifact type. Collect the asserted memberships on the artifact, its references and their known IRI aliases before attempting recognition. These assignments take precedence over all prefix rules, and several explicit memberships are permitted. Recognition applies only when no membership is asserted; its result is derived metadata and must not silently be stored as an explicit inFamily assertion. Without an assertion or a matching rule, the artifact remains without a family.

ekg-artgov:inFamily	
https://ekgf.org/ontology/artifact-governance#inFamily	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:label	In family
rdfs:range	ekg-artgov:Family

4.3.16 In module

Links a member to the one module directly containing it. Query `inModule+` for all enclosing modules; the property is not transitive. A member's family follows from its module, so `inFamily` on the member is optional. When a member does declare families, they must include its module's family.

```
@prefix artgov: <https://ekgf.org/ontology/artifact-governance#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix ex: <https://example.org/> .

ex:fibo a artgov:Family .
```

```
ex:fnd a artgov:Module ; rdfs:label "FND"@en ; artgov:inFamily ex:fibo .
ex:utilities a artgov:Module ; rdfs:label "Utilities"@en ;
    artgov:inFamily ex:fibo ; artgov:inModule ex:fnd .
ex:annotationVocabulary artgov:inModule ex:utilities .
```

`ModuleShape` and `ModuleMembershipShape` validate this structure.

ekg-artgov:inModule	
https://ekgf.org/ontology/artifact-governance#inModule	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:label	In module
rdfs:range	ekg-artgov:Module

4.3.17 Legal owner

Links an authority or an organization to the organization that legally owns or administers it. This relationship is separate from the party exercising an authority role, stated with cmns-rlcmp:isPlayedBy. This is governance metadata, not access inheritance.

ekg-artgov:legalOwner	
https://ekgf.org/ontology/artifact-governance#legalOwner	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:label	Legal owner
rdfs:range	cmns-org:Organization

4.3.18 Maximum family load quads

Hard ceiling on the total number of quads accepted while loading the family.

ekg-artgov:maxFamilyLoadQuads	
https://ekgf.org/ontology/artifact-governance#maxFamilyLoadQuads	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-artgov:Family
rdfs:label	Maximum family load quads
rdfs:range	xsd:unsignedLong

4.3.19 Maximum source quads

Hard ceiling on the number of quads accepted from one source file of the family.

ekg-artgov:maxSourceQuads	
https://ekgf.org/ontology/artifact-governance#maxSourceQuads	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-artgov:Family
rdfs:label	Maximum source quads
rdfs:range	xsd:unsignedLong

4.3.20 Namespace IRI prefix

A case-sensitive literal prefix of an artifact's IRI, such as https://spec.edmcouncil.org/fibo/ontology/. It is an absolute IRI prefix, typed xsd:anyURI, and a family may have several. The longest matching namespace prefix wins within this recognition method. Equally long best matches for different families are an error. Lengths are not compared with filePathPrefix or graphIriPrefix; if their winning families disagree, recognition fails. Explicit inFamily assignments take precedence over all recognition methods.

ekg-artgov:namespaceIriPrefix	
https://ekgf.org/ontology/artifact-governance#namespaceIriPrefix	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-artgov:Family
rdfs:label	Namespace IRI prefix
rdfs:range	xsd:anyURI

4.3.21 Organization category

Classifies an organization by one or more organization categories. Set on the organization playing an authority role or hosting its community. An authority's player can be reached with cmns-rlcmp:isPlayedBy. Categories are asserted independently of legal type, organizational hierarchy and the roles that an organization plays; they are not inherited from a parent organization.

ekg-artgov:organizationCategory	
https://ekgf.org/ontology/artifact-governance#organizationCategory	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	cmns-org:Organization
rdfs:label	Organization category
rdfs:range	ekg-artgov:OrganizationCategory
rdfs:subPropertyOf	cmns-cls:isClassifiedBy

4.3.22 Predecessor authority

Links an authority to a former authority identity that preceded it.

ekg-artgov:predecessorAuthority	
https://ekgf.org/ontology/artifact-governance#predecessorAuthority	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-artgov:Authority
rdfs:label	Predecessor authority
rdfs:range	ekg-artgov:Authority

4.3.23 Publishing role

Describes how an authority participates in publishing, owning, maintaining or endorsing artifacts.

ekg-artgov:publishingRole	
https://ekgf.org/ontology/artifact-governance#publishingRole	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-artgov:Authority
rdfs:label	Publishing role
rdfs:range	ekg-artgov:AuthorityPublishingRole

4.3.24 RDF file extension

Restricts file discovery to files with this extension, such as .rdf or .ttl. Every supported RDF extension when absent.

ekg-artgov:rdfFileExtension	
https://ekgf.org/ontology/artifact-governance#rdfFileExtension	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-artgov:Family
rdfs:label	RDF file extension
rdfs:range	xsd:string

4.3.25 Refers to

Relates a reference to the IRI of the artifact it stands for. A reference carries several values when the artifact is known under several IRIs.

ekg-artgov:refersTo	
https://ekgf.org/ontology/artifact-governance#refersTo	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty

ekg-artgov:refersTo	
https://ekgf.org/ontology/artifact-governance#refersTo	
RDF predicate	**Value**
rdfs:domain	ekg-artgov:Reference
rdfs:label	Refers to

4.3.26 Source graph repository URL

Canonical repository URL used to build stable, commit-pinned named graph IRIs for the source files when the loader clones from a mirror or a local checkout.

ekg-artgov:sourceGraphRepositoryUrl	
https://ekgf.org/ontology/artifact-governance#sourceGraphRepositoryUrl	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-artgov:Family
rdfs:label	Source graph repository URL
rdfs:range	xsd:anyURI

4.3.27 Successor authority

Links an authority to a later authority identity that supersedes it.

ekg-artgov:successorAuthority	
https://ekgf.org/ontology/artifact-governance#successorAuthority	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-artgov:Authority
rdfs:label	Successor authority
rdfs:range	ekg-artgov:Authority

4.4 Named Individuals

4.4.1 co-owner

Instance of: ekg-artgov:AuthorityPublishingRole

The authority co-owns the published artifacts with one or more other authorities.

ekg-artgov:AuthorityCoOwnerRole	
https://ekgf.org/ontology/artifact-governance#AuthorityCoOwnerRole	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-artgov:AuthorityPublishingRole

ekg-artgov:AuthorityCoOwnerRole	
https://ekgf.org/ontology/artifact-governance#AuthorityCoOwnerRole	
RDF predicate	**Value**
rdfs:label	co-owner
ekg-annotation:slug	co-owner

4.4.2 maintainer

Instance of: ekg-artgov:AuthorityPublishingRole

The authority maintains the artifacts between publication releases.

ekg-artgov:AuthorityMaintainerRole	
https://ekgf.org/ontology/artifact-governance#AuthorityMaintainerRole	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-artgov:AuthorityPublishingRole
rdfs:label	maintainer
ekg-annotation:slug	maintainer

4.4.3 owner

Instance of: ekg-artgov:AuthorityPublishingRole

The authority is the primary owning body of the published artifacts.

ekg-artgov:AuthorityOwnerRole	
https://ekgf.org/ontology/artifact-governance#AuthorityOwnerRole	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-artgov:AuthorityPublishingRole
rdfs:label	owner
ekg-annotation:slug	owner

4.4.4 publisher

Instance of: ekg-artgov:AuthorityPublishingRole

The authority publishes the artifacts or their specification site.

ekg-artgov:AuthorityPublisherRole	
https://ekgf.org/ontology/artifact-governance#AuthorityPublisherRole	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-artgov:AuthorityPublishingRole

ekg-artgov:AuthorityPublisherRole	
https://ekgf.org/ontology/artifact-governance#AuthorityPublisherRole	
RDF predicate	**Value**
rdfs:label	publisher
ekg-annotation:slug	publisher

4.4.5 Community

Instance of: ekg-artgov:OrganizationCategory

An organized community whose participants collaborate around a shared topic or practice.

ekg-artgov:CommunityCategory	
https://ekgf.org/ontology/artifact-governance#CommunityCategory	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-artgov:OrganizationCategory
rdfs:label	Community
ekg-annotation:slug	community
cmns-dsg:isDefinedIn	ekg-artgov:OrganizationCategoryScheme

4.4.6 Consortium

Instance of: ekg-artgov:OrganizationCategory

An organization through which independent organizations cooperate toward shared objectives.

ekg-artgov:ConsortiumCategory	
https://ekgf.org/ontology/artifact-governance#ConsortiumCategory	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-artgov:OrganizationCategory
rdfs:label	Consortium
ekg-annotation:slug	consortium
cmns-dsg:isDefinedIn	ekg-artgov:OrganizationCategoryScheme

4.4.7 Organization categories for artifact governance

Instance of: cmns-cls:ClassificationScheme

An extensible classification scheme for organizations involved in artifact governance. Categories may overlap: an organization may be both a standards body and a consortium. A missing category means that no classification has been asserted.

ekg-artgov:OrganizationCategoryScheme	
https://ekgf.org/ontology/artifact-governance#OrganizationCategoryScheme	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	cmns-cls:ClassificationScheme
rdfs:label	Organization categories for artifact governance
cmns-cls:isExclusive	false

4.4.8 Project

Instance of: ekg-artgov:OrganizationCategory

An organized group carrying out a defined initiative and producing its work products.

ekg-artgov:ProjectCategory	
https://ekgf.org/ontology/artifact-governance#ProjectCategory	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-artgov:OrganizationCategory
rdfs:label	Project
ekg-annotation:slug	project
cmns-dsg:isDefinedIn	ekg-artgov:OrganizationCategoryScheme

4.4.9 Standards body

Instance of: ekg-artgov:OrganizationCategory

An organization whose purpose includes developing, maintaining and publishing standards through an established governance process.

ekg-artgov:StandardsBodyCategory	
https://ekgf.org/ontology/artifact-governance#StandardsBodyCategory	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-artgov:OrganizationCategory
rdfs:label	Standards body
ekg-annotation:slug	standards-body
cmns-dsg:isDefinedIn	ekg-artgov:OrganizationCategoryScheme

4.4.10 standards editor

Instance of: ekg-artgov:StandardsCommunityRole

A standards-community participation role for editing the content of standards and their supporting artifacts.

ekg-artgov:StandardsEditorRole	
https://ekgf.org/ontology/artifact-governance#StandardsEditorRole	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-artgov:StandardsCommunityRole
rdfs:label	standards editor
ekg-annotation:slug	standards-editor

4.4.11 standards publisher

Instance of: ekg-artgov:StandardsCommunityRole

A standards-community participation role for publishing standards artifacts and releases.

ekg-artgov:StandardsPublisherRole	
https://ekgf.org/ontology/artifact-governance#StandardsPublisherRole	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-artgov:StandardsCommunityRole
rdfs:label	standards publisher
ekg-annotation:slug	standards-publisher

4.4.12 standards reviewer

Instance of: ekg-artgov:StandardsCommunityRole

A standards-community participation role for reviewing standards content and proposed changes.

ekg-artgov:StandardsReviewerRole	
https://ekgf.org/ontology/artifact-governance#StandardsReviewerRole	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-artgov:StandardsCommunityRole
rdfs:label	standards reviewer
ekg-annotation:slug	standards-reviewer

4.4.13 standards staff

Instance of: ekg-artgov:StandardsCommunityRole

A standards-community participation role for staff supporting the operation of the standards community.

ekg-artgov:StandardsStaffRole	
https://ekgf.org/ontology/artifact-governance#StandardsStaffRole	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-artgov:StandardsCommunityRole
rdfs:label	standards staff
ekg-annotation:slug	standards-staff

4.4.14 working-group chair

Instance of: ekg-artgov:StandardsCommunityRole

A standards-community participation role for chairing a working group and coordinating its work.

ekg-artgov:WorkingGroupChairRole	
https://ekgf.org/ontology/artifact-governance#WorkingGroupChairRole	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-artgov:StandardsCommunityRole
rdfs:label	working-group chair
ekg-annotation:slug	working-group-chair

4.5 Shapes

4.5.1 Module Membership Shape

Validates module membership: at most one direct module per member, and agreement with the module's family when the member declares families. A nested module is a member of its parent and always declares its family.

ekg-artgov:ModuleMembershipShape	
https://ekgf.org/ontology/artifact-governance#ModuleMembershipShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Module Membership Shape
sh:property	(sh:class: ekg-artgov:Module; sh:maxCount: 1; sh:message: A member has at most one direct module, identified by a Module IRI.; sh:nodeKind: sh:IRI; sh:path: ekg-artgov:inModule)
sh:sparql	(sh:message: A member that declares families must include its module's family {?family}.; sh:select: SPARQL SELECT query 1)
sh:targetSubjectsOf	ekg-artgov:inModule

4.5.1.1 SPARQL query 1

A member that declares families must include its module's family {?family}.

```
PREFIX artgov: <https://ekgf.org/ontology/artifact-governance#>
SELECT $this ?family WHERE {
    $this artgov:inModule/artgov:inFamily ?family ;
        artgov:inFamily ?declared .
    FILTER NOT EXISTS { $this artgov:inFamily ?family }
}
```

4.5.2 Module Shape

Validates modules: an IRI with a label and exactly one family, distinct from families, references and authorities, with acyclic nesting.

ekg-artgov:ModuleShape	
https://ekgf.org/ontology/artifact-governance#ModuleShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Module Shape
sh:nodeKind	sh:IRI
sh:not	(sh:or: ((sh:class: ekg-artgov:Family), (sh:class: ekg-artgov:Reference), (sh:class: ekg-artgov:Authority)))
sh:property	(sh:class: ekg-artgov:Family; sh:maxCount: 1; sh:message: A module belongs to exactly one family.; sh:minCount: 1; sh:nodeKind: sh:IRI; sh:path: ekg-artgov:inFamily)
sh:property	(sh:message: A module needs a nonempty label.; sh:minCount: 1; sh:minLength: 1; sh:nodeKind: sh:Literal; sh:path: rdfs:label)
sh:sparql	(sh:message: Module nesting must be acyclic; a module cannot contain itself directly or through other modules.; sh:select: SPARQL SELECT query 1)
sh:targetClass	ekg-artgov:Module
sh:targetObjectsOf	ekg-artgov:inModule

4.5.2.1 SPARQL query 1

Module nesting must be acyclic; a module cannot contain itself directly or through other modules.

```
PREFIX artgov: <https://ekgf.org/ontology/artifact-governance#>
SELECT $this WHERE { $this artgov:inModule+ $this . }
```

5 Artifact Publication Ontology

5.1 Overview

Artifact Publication Ontology

Identifier	Value
Ontology IRI / Namespace	`https://ekgf.org/ontology/artifact-publication#`
Prefix	`ekg-artpub`

How semantic artifacts are published, either as a family or as explicitly selected artifact references: profiles configure generation, targets identify destinations, runs generate outputs, releases identify versions and approvals record authority decisions about exact releases. Family grouping is optional. The state of a run and of a release is described with the Lifecycle Ontology. SHACL shapes validate the scope and approval records.

5.2 Classes

5.2.1 Publication artifact

A generated output of a publication run, such as a website bundle, PDF, RDF export, changelog, diff report or archive.

`ekg-artpub:PublicationArtifact`	
`https://ekgf.org/ontology/artifact-publication#PublicationArtifact`	
RDF predicate	**Value**
`rdf:type`	`owl:Class`
`rdfs:label`	Publication artifact

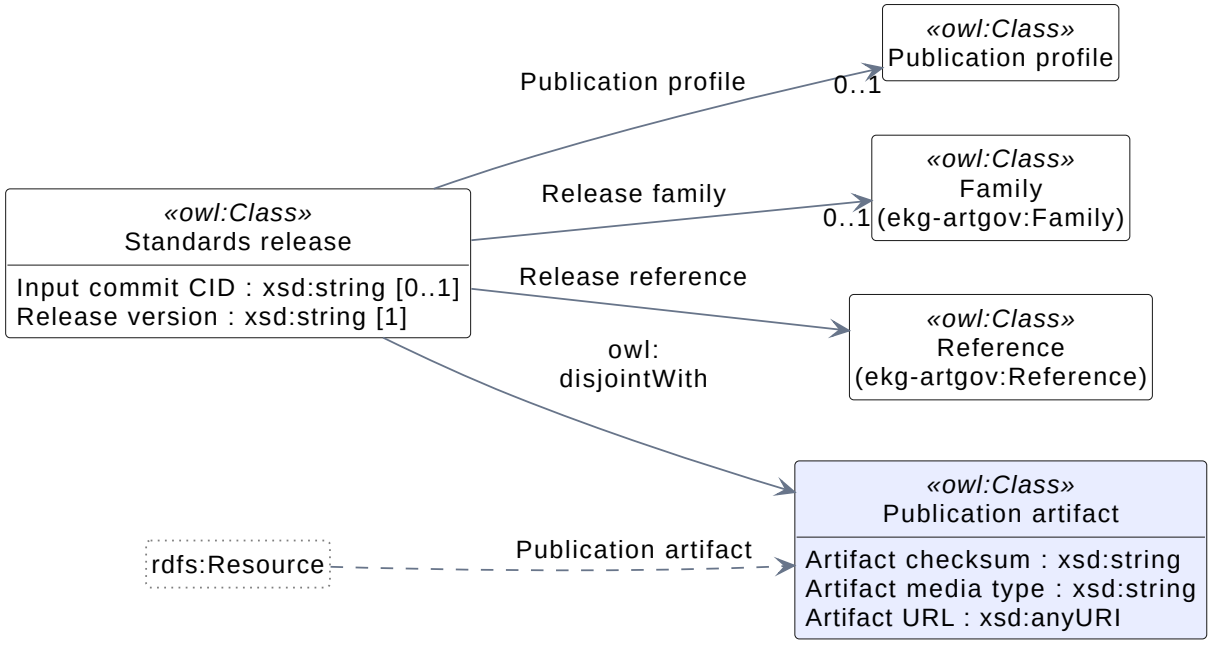


Diagram note

5.2.2 Publication profile

Branch-scoped generator configuration for publishing either one family or one or more explicit artifact references, including section ordering, render policy, validation gates and output targets. Retain the configuration referenced by a release; changed configuration requires a new profile identity.

ekg-artpub:PublicationProfile	
https://ekgf.org/ontology/artifact-publication#PublicationProfile	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Publication profile

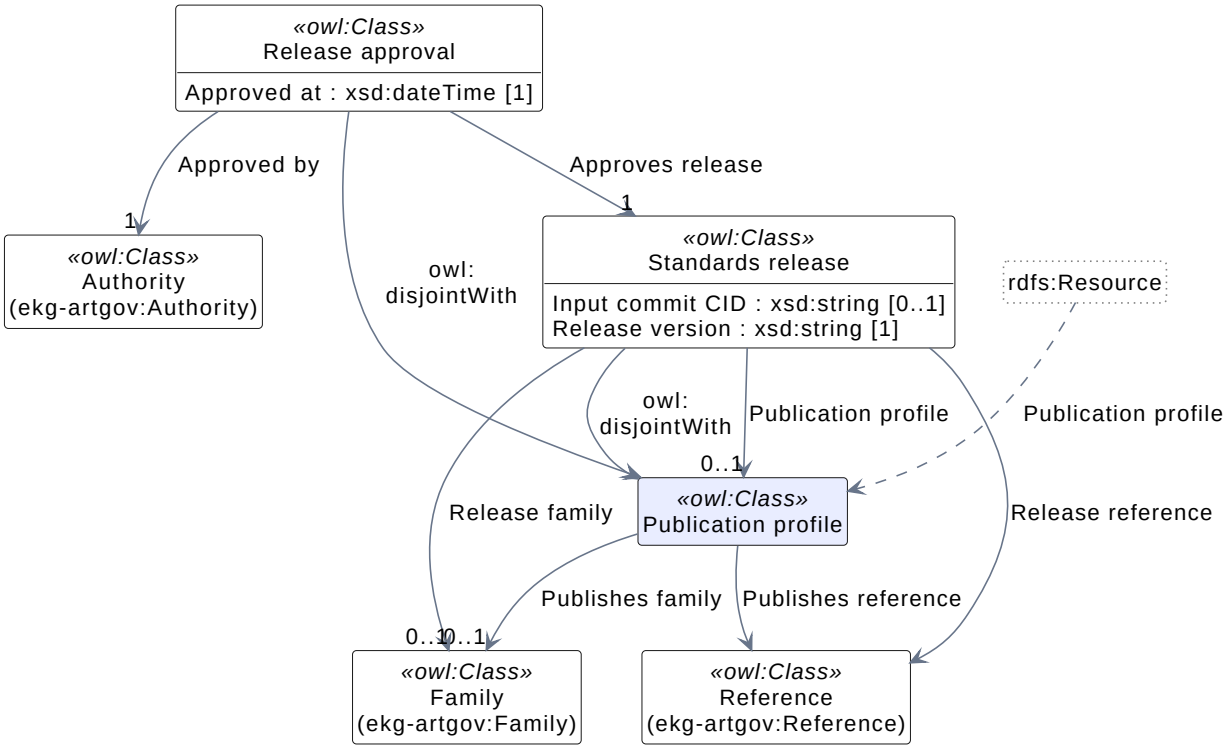


Diagram note

5.2.3 Publication run

An execution that generates publication artifacts for the scope selected by a profile, branch and input commit. The scope may be a family or explicit artifact references. How far it has come is its lifecycle state; by default a run follows the execution lifecycle.

ekg-artpub:PublicationRun	
https://ekgf.org/ontology/artifact-publication#PublicationRun	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Publication run

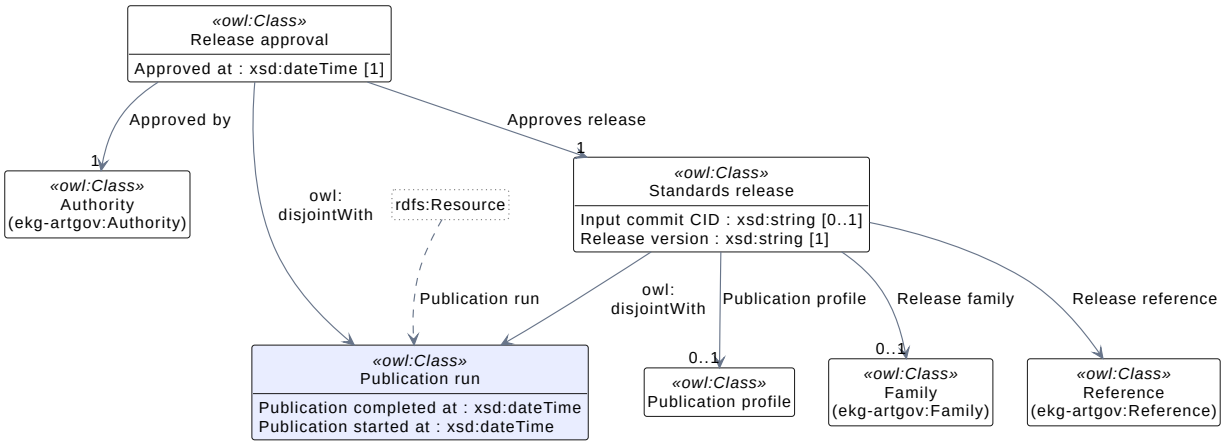


Diagram note

5.2.4 Publication target

A configured destination for a generated publication output, such as a website, PDF, RDF download, archive or package.

ekg-artpub:PublicationTarget	
https://ekgf.org/ontology/artifact-publication#PublicationTarget	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Publication target

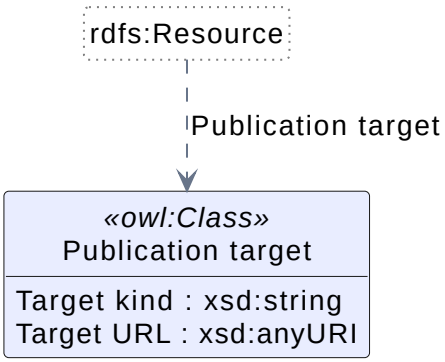


Diagram note

5.2.5 Release approval

An affirmative approval decision by one authority about one exact standards release at a recorded time. Several authorities can approve the same release through separate records. An approval does not extend to other releases or future family members, and does not itself publish the release. Supporting evidence can be linked with rdfs:seeAlso. Required approvers and any withdrawal of approval are matters for the applicable governance policy; a governing or publishing role alone does not establish that approval occurred.

ekg-artpub:ReleaseApproval	
https://ekgf.org/ontology/artifact-publication#ReleaseApproval	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Release approval
owl:disjointWith	ekg-artgov:Family
owl:disjointWith	ekg-artgov:Reference
owl:disjointWith	ekg-artpub:PublicationProfile
owl:disjointWith	ekg-artpub:PublicationRun
owl:disjointWith	ekg-artpub:StandardsRelease

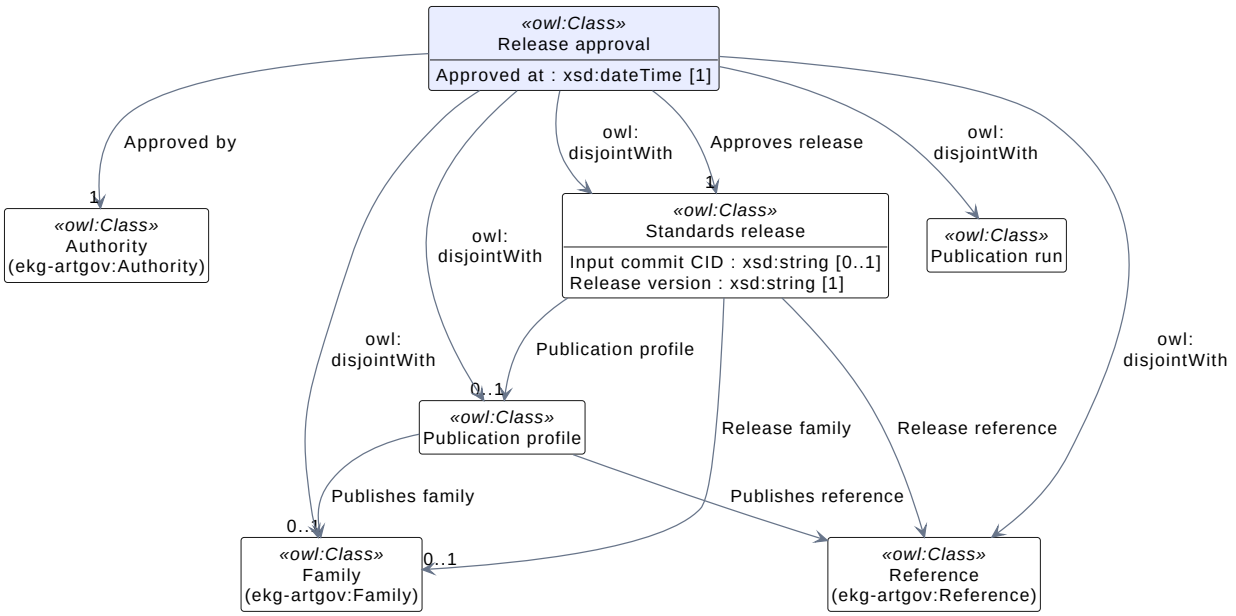


Diagram note

5.2.6 Standards release

A versioned snapshot of either one family or one or more explicit artifact references. Its identity, scope, version and content provenance are immutable; changed content requires a new release identity and new approvals. A release may record a branch, input commit, publication run and generated artifacts where available, but externally produced releases do not require a local generator run. Its lifecycle state may change from candidate to published, superseded or withdrawn. A version label is scoped to the release's publication context, not a global identifier. Family membership is optional and is not inferred from the references in a release.

ekg-artpub:StandardsRelease	
https://ekgf.org/ontology/artifact-publication#StandardsRelease	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Standards release

ekg-artpub:StandardsRelease	
https://ekgf.org/ontology/artifact-publication#StandardsRelease	
RDF predicate	**Value**
owl:disjointWith	ekg-artgov:Family
owl:disjointWith	ekg-artgov:Reference
owl:disjointWith	ekg-artpub:PublicationArtifact
owl:disjointWith	ekg-artpub:PublicationProfile
owl:disjointWith	ekg-artpub:PublicationRun

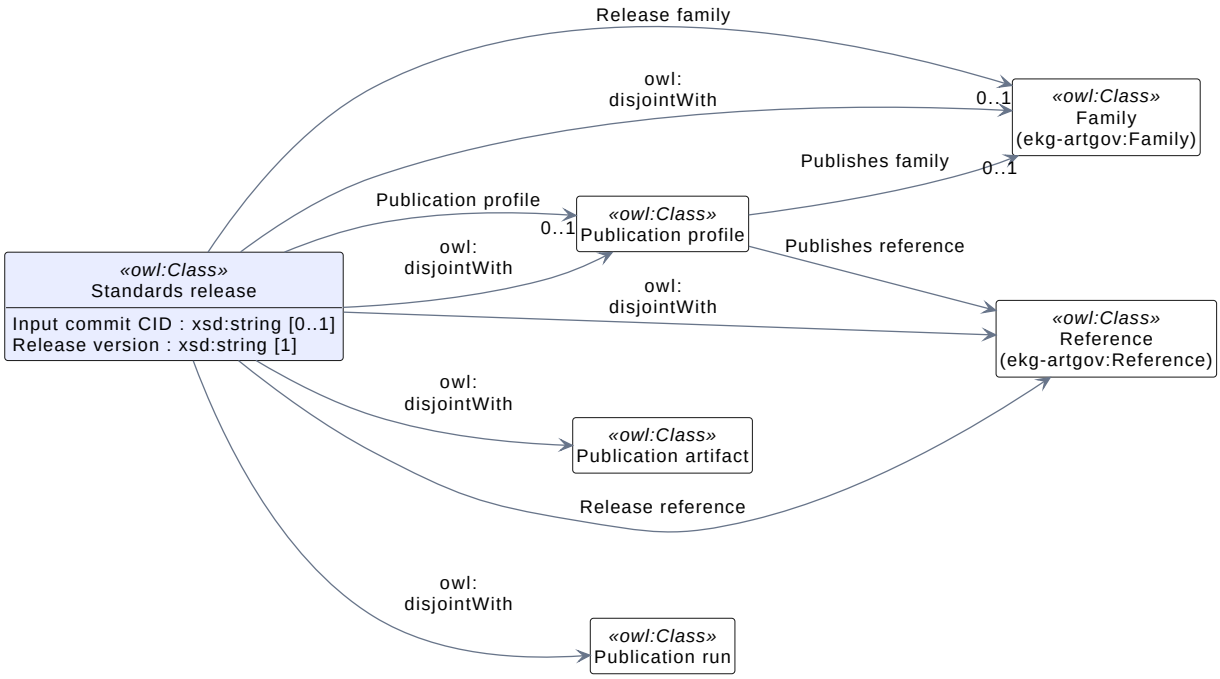


Diagram note

5.3 Properties

5.3.1 Approved at

Timestamp at which the authority approved the identified release. This is distinct from when it was generated or published.

ekg-artpub:approvedAt	
https://ekgf.org/ontology/artifact-publication#approvedAt	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-artpub:ReleaseApproval
rdfs:label	Approved at
rdfs:range	xsd:dateTime

5.3.2 Approved by

Identifies the authority role that issued this approval. The organization playing the role has its own identity. A separate decision record is used for each approving authority.

ekg-artpub:approvedBy	
https://ekgf.org/ontology/artifact-publication#approvedBy	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-artpub:ReleaseApproval
rdfs:label	Approved by
rdfs:range	ekg-artgov:Authority

5.3.3 Approves release

Identifies the exact release approved by this decision. Approval is never attached to the enduring family or artifact reference.

ekg-artpub:approvesRelease	
https://ekgf.org/ontology/artifact-publication#approvesRelease	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-artpub:ReleaseApproval
rdfs:label	Approves release
rdfs:range	ekg-artpub:StandardsRelease

5.3.4 Artifact checksum

Checksum or digest of a generated publication artifact.

ekg-artpub:artifactChecksum	
https://ekgf.org/ontology/artifact-publication#artifactChecksum	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-artpub:PublicationArtifact
rdfs:label	Artifact checksum
rdfs:range	xsd:string

5.3.5 Artifact media type

Media type of a generated publication artifact, such as text/html, application/pdf or text/turtle.

ekg-artpub:artifactMediaType	
https://ekgf.org/ontology/artifact-publication#artifactMediaType	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-artpub:PublicationArtifact
rdfs:label	Artifact media type
rdfs:range	xsd:string

5.3.6 Artifact URL

Stable URL where a generated publication artifact can be read.

ekg-artpub:artifactUrl	
https://ekgf.org/ontology/artifact-publication#artifactUrl	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-artpub:PublicationArtifact
rdfs:label	Artifact URL
rdfs:range	xsd:anyURI

5.3.7 Generator version

Version of the publication generator used by a publication profile, run, artifact or release.

ekg-artpub:generatorVersion	
https://ekgf.org/ontology/artifact-publication#generatorVersion	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:label	Generator version
rdfs:range	xsd:string

5.3.8 Input commit CID

Content identifier of the branch commit used as input for a publication run or release.

ekg-artpub:inputCommitCid	
https://ekgf.org/ontology/artifact-publication#inputCommitCid	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:label	Input commit CID
rdfs:range	xsd:string

5.3.9 Publication artifact

Links a publication run or a standards release to a generated publication artifact.

ekg-artpub:publicationArtifact	
https://ekgf.org/ontology/artifact-publication#publicationArtifact	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:label	Publication artifact
rdfs:range	ekg-artpub:PublicationArtifact

5.3.10 Publication branch

Links a publication run or a standards release to the branch whose commit was published.

ekg-artpub:publicationBranch	
https://ekgf.org/ontology/artifact-publication#publicationBranch	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:label	Publication branch
rdfs:range	rdfs:Resource

5.3.11 Publication completed at

Timestamp at which a publication run completed.

ekg-artpub:publicationCompletedAt	
https://ekgf.org/ontology/artifact-publication#publicationCompletedAt	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-artpub:PublicationRun
rdfs:label	Publication completed at
rdfs:range	xsd:dateTime

5.3.12 Publication profile

Links a publication run, target, artifact or release to the profile that governed generation.

ekg-artpub:publicationProfile	
https://ekgf.org/ontology/artifact-publication#publicationProfile	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:label	Publication profile

ekg-artpub:publicationProfile	
https://ekgf.org/ontology/artifact-publication#publicationProfile	
RDF predicate	**Value**
rdfs:range	ekg-artpub:PublicationProfile

5.3.13 Publication release

Links a publication run or a generated artifact to the standards release that incorporated it.

ekg-artpub:publicationRelease	
https://ekgf.org/ontology/artifact-publication#publicationRelease	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:label	Publication release
rdfs:range	ekg-artpub:StandardsRelease

5.3.14 Publication run

Links a standards release or a generated artifact to the publication run that produced it.

ekg-artpub:publicationRun	
https://ekgf.org/ontology/artifact-publication#publicationRun	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:label	Publication run
rdfs:range	ekg-artpub:PublicationRun

5.3.15 Publication started at

Timestamp at which a publication run started.

ekg-artpub:publicationStartedAt	
https://ekgf.org/ontology/artifact-publication#publicationStartedAt	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-artpub:PublicationRun
rdfs:label	Publication started at
rdfs:range	xsd:dateTime

5.3.16 Publication target

Links a publication profile, run or artifact to a configured output target.

ekg-artpub:publicationTarget	
https://ekgf.org/ontology/artifact-publication#publicationTarget	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:label	Publication target
rdfs:range	ekg-artpub:PublicationTarget

5.3.17 Publishes family

Links a publication profile to the one family that it publishes. Optional: a profile can instead select artifacts individually with publishesReference.

ekg-artpub:publishesFamily	
https://ekgf.org/ontology/artifact-publication#publishesFamily	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-artpub:PublicationProfile
rdfs:label	Publishes family
rdfs:range	ekg-artgov:Family

5.3.18 Publishes reference

Selects an artifact by its stable reference for publication without a family. A profile can select several references. This does not create a family or assert family membership.

ekg-artpub:publishesReference	
https://ekgf.org/ontology/artifact-publication#publishesReference	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-artpub:PublicationProfile
rdfs:label	Publishes reference
rdfs:range	ekg-artgov:Reference

5.3.19 Release family

Identifies the one family whose version this release records. Optional: an ungrouped release selects artifacts with releaseReference instead. This is recorded explicitly, so editing a publication profile cannot change an existing release's scope.

ekg-artpub:releaseFamily	
https://ekgf.org/ontology/artifact-publication#releaseFamily	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty

ekg-artpub:releaseFamily	
https://ekgf.org/ontology/artifact-publication#releaseFamily	
RDF predicate	**Value**
rdfs:domain	ekg-artpub:StandardsRelease
rdfs:label	Release family
rdfs:range	ekg-artgov:Family

5.3.20 Release reference

Identifies an artifact included in a release without family grouping. Each selected artifact is represented by a stable reference; the release supplies its version and content provenance. A release can select several references. These are source artifacts, distinct from generated outputs linked by publicationArtifact.

ekg-artpub:releaseReference	
https://ekgf.org/ontology/artifact-publication#releaseReference	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-artpub:StandardsRelease
rdfs:label	Release reference
rdfs:range	ekg-artgov:Reference

5.3.21 Release version

Public version label of a standards release, whether it releases a family or explicitly selected artifact references. The label belongs to the release, not the enduring family or reference, and need not be globally unique.

ekg-artpub:releaseVersion	
https://ekgf.org/ontology/artifact-publication#releaseVersion	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-artpub:StandardsRelease
rdfs:label	Release version
rdfs:range	xsd:string

5.3.22 Released at

Timestamp at which a standards release was published.

ekg-artpub:releasedAt	
https://ekgf.org/ontology/artifact-publication#releasedAt	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty

ekg-artpub:releasedAt	
https://ekgf.org/ontology/artifact-publication#releasedAt	
RDF predicate	**Value**
rdfs:domain	ekg-artpub:StandardsRelease
rdfs:label	Released at
rdfs:range	xsd:dateTime

5.3.23 Target kind

Kind of publication target, such as website, pdf, rdf-download, archive or package.

ekg-artpub:targetKind	
https://ekgf.org/ontology/artifact-publication#targetKind	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-artpub:PublicationTarget
rdfs:label	Target kind
rdfs:range	xsd:string

5.3.24 Target URL

Base URL or destination URL of a publication target.

ekg-artpub:targetUrl	
https://ekgf.org/ontology/artifact-publication#targetUrl	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-artpub:PublicationTarget
rdfs:label	Target URL
rdfs:range	xsd:anyURI

5.4 Shapes

5.4.1 Publication Profile Shape

Validates that a publication profile explicitly selects a family or artifact references using the supported scope alternatives.

ekg-artpub:PublicationProfileShape	
https://ekgf.org/ontology/artifact-publication#PublicationProfileShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Publication Profile Shape

ekg-artpub:PublicationProfileShape	
https://ekgf.org/ontology/artifact-publication#PublicationProfileShape	
RDF predicate	**Value**
sh:class	ekg-artpub:PublicationProfile
sh:message	A publication profile selects either one family or one or more artifact references.
sh:nodeKind	sh:IRI
sh:property	(sh:class: ekg-artgov:Family; sh:maxCount: 1; sh:nodeKind: sh:IRI; sh:path: ekg-artpub:publishesFamily)
sh:property	(sh:class: ekg-artgov:Reference; sh:nodeKind: sh:IRI; sh:path: ekg-artpub:publishesReference)
sh:targetClass	ekg-artpub:PublicationProfile
sh:targetObjectsOf	ekg-artpub:publicationProfile
sh:targetSubjectsOf	ekg-artpub:publishesFamily
sh:targetSubjectsOf	ekg-artpub:publishesReference
sh:xone	((sh:property: (sh:minCount: 1; sh:path: ekg-artpub:publishesFamily); sh:property: (sh:maxCount: 0; sh:path: ekg-artpub:publishesReference)), (sh:property: (sh:minCount: 1; sh:path: ekg-artpub:publishesReference); sh:property: (sh:maxCount: 0; sh:path: ekg-artpub:publishesFamily)))

5.4.2 Release Approval Shape

Validates an approval record identifying the exact release approved, the approving authority, and the approval time.

ekg-artpub:ReleaseApprovalShape	
https://ekgf.org/ontology/artifact-publication#ReleaseApprovalShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Release Approval Shape
sh:class	ekg-artpub:ReleaseApproval
sh:nodeKind	sh:IRI
sh:not	(sh:or: ((sh:class: ekg-artgov:Family), (sh:class: ekg-artgov:Reference), (sh:class: ekg-artpub:StandardsRelease), (sh:class: ekg-artpub:PublicationProfile), (sh:class: ekg-artpub:PublicationRun)))
sh:property	(sh:class: ekg-artgov:Authority; sh:maxCount: 1; sh:minCount: 1; sh:nodeKind: sh:IRI; sh:path: ekg-artpub:approvedBy)
sh:property	(sh:class: ekg-artpub:StandardsRelease; sh:maxCount: 1; sh:minCount: 1; sh:nodeKind: sh:IRI; sh:path: ekg-artpub:approvesRelease)
sh:property	(sh:datatype: xsd:dateTime; sh:maxCount: 1; sh:minCount: 1; sh:path: ekg-artpub:approvedAt)
sh:targetClass	ekg-artpub:ReleaseApproval
sh:targetSubjectsOf	ekg-artpub:approvedAt
sh:targetSubjectsOf	ekg-artpub:approvedBy

ekg-artpub:ReleaseApprovalShape	
https://ekgf.org/ontology/artifact-publication#ReleaseApprovalShape	
RDF predicate	**Value**
sh:targetSubjectsOf	ekg-artpub:approvesRelease

5.4.3 Standards Release Shape

Validates a standards release's family or reference scope, version label, and optional publication and content-provenance records.

ekg-artpub:StandardsReleaseShape	
https://ekgf.org/ontology/artifact-publication#StandardsReleaseShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Standards Release Shape
sh:class	ekg-artpub:StandardsRelease
sh:nodeKind	sh:IRI
sh:not	(sh:or: ((sh:class: ekg-artgov:Family), (sh:class: ekg-artgov:Reference), (sh:class: ekg-artpub:PublicationProfile), (sh:class: ekg-artpub:PublicationRun), (sh:class: ekg-artpub:PublicationArtifact), (sh:class: ekg-artpub:ReleaseApproval)))
sh:property	(sh:class: ekg-artgov:Family; sh:maxCount: 1; sh:nodeKind: sh:IRI; sh:path: ekg-artpub:releaseFamily)
sh:property	(sh:class: ekg-artpub:PublicationProfile; sh:maxCount: 1; sh:nodeKind: sh:IRI; sh:path: ekg-artpub:publicationProfile)
sh:property	(sh:datatype: xsd:string; sh:maxCount: 1; sh:minLength: 1; sh:path: ekg-artpub:inputCommitCid)
sh:property	(sh:class: ekg-artgov:Reference; sh:nodeKind: sh:IRI; sh:path: ekg-artpub:releaseReference)
sh:property	(sh:datatype: xsd:string; sh:maxCount: 1; sh:minCount: 1; sh:minLength: 1; sh:path: ekg-artpub:releaseVersion)
sh:property	(sh:class: ekg-artpub:PublicationRun; sh:maxCount: 1; sh:node: (sh:property: (sh:datatype: xsd:string; sh:maxCount: 1; sh:minCount: 1; sh:minLength: 1; sh:path: ekg-artpub:inputCommitCid); sh:property: (sh:class: ekg-artpub:PublicationProfile; sh:maxCount: 1; sh:minCount: 1; sh:nodeKind: sh:IRI; sh:path: ekg-artpub:publicationProfile)); sh:nodeKind: sh:IRI; sh:path: ekg-artpub:publicationRun)
sh:sparql	(sh:message: A release and every linked publication profile must select the same family or the same artifact references.; sh:select: SPARQL SELECT query 1)
sh:sparql	(sh:message: A release linked to a publication run must record the same input commit.; sh:select: SPARQL SELECT query 2)
sh:targetClass	ekg-artpub:StandardsRelease
sh:targetObjectsOf	ekg-artpub:approvesRelease
sh:targetObjectsOf	ekg-artpub:publicationRelease
sh:targetSubjectsOf	ekg-artpub:releaseFamily

ekg-artpub:StandardsReleaseShape	
https://ekgf.org/ontology/artifact-publication#StandardsReleaseShape	
RDF predicate	**Value**
sh:targetSubjectsOf	ekg-artpub:releaseReference
sh:targetSubjectsOf	ekg-artpub:releaseVersion
sh:xone	((sh:property: (sh:maxCount: 0; sh:path: ekg-artpub:releaseReference); sh:property: (sh:minCount: 1; sh:path: ekg-artpub:releaseFamily)), (sh:property: (sh:maxCount: 0; sh:path: ekg-artpub:releaseFamily); sh:property: (sh:minCount: 1; sh:path: ekg-artpub:releaseReference)))

5.4.3.1 SPARQL query 1

A release and every linked publication profile must select the same family or the same artifact references.

```
PREFIX artpub: <https://ekgf.org/ontology/artifact-publication#>
SELECT $this WHERE {
    $this (artpub:publicationProfile|artpub:publicationRun/artpub:publicationProfile) ?pro
file .
    {
        { $this artpub:releaseFamily ?scope .
          FILTER NOT EXISTS { ?profile artpub:publishesFamily ?scope } }
        UNION
        { ?profile artpub:publishesFamily ?scope .
          FILTER NOT EXISTS { $this artpub:releaseFamily ?scope } }
        UNION
        { $this artpub:releaseReference ?scope .
          FILTER NOT EXISTS { ?profile artpub:publishesReference ?scope } }
        UNION
        { ?profile artpub:publishesReference ?scope .
          FILTER NOT EXISTS { $this artpub:releaseReference ?scope } }
    }
}
```

5.4.3.2 SPARQL query 2

A release linked to a publication run must record the same input commit.

```
PREFIX artpub: <https://ekgf.org/ontology/artifact-publication#>
SELECT $this WHERE {
    $this artpub:publicationRun/artpub:inputCommitCid ?commit .
    FILTER NOT EXISTS { $this artpub:inputCommitCid ?commit }
}
```

6 Authorization Ontology

6.1 Overview

Authorization Ontology

Identifier	Value
Ontology IRI / Namespace	`https://ekgf.org/ontology/authorization#`
Prefix	`ekg-authorization`

The EKGF Authorization Ontology.

Modeling "Role Based Access Control" which is not the same as EKG-level entitlement calculation but could be part thereof.

6.2 Classes

6.2.1 Authorization Group

A grouping used by authorization rules to express role-based access-control requirements. It is distinct from a persona and does not by itself calculate EKG-wide entitlements.

`ekg-authorization:AuthorizationGroup`	
`https://ekgf.org/ontology/authorization#AuthorizationGroup`	
RDF predicate	**Value**
`rdf:type`	`owl:Class`
`rdfs:label`	Authorization Group

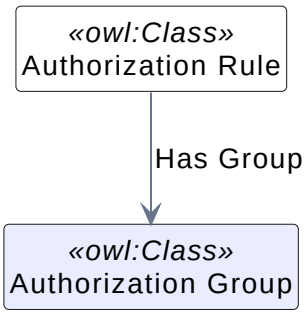


Diagram note

6.2.2 Authorization Rule

A rule associating authorization groups with personas in the role-based access-control model. The vocabulary records these relationships without defining a rule evaluation algorithm.

ekg-authorization:AuthorizationRule	
https://ekgf.org/ontology/authorization#AuthorizationRule	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Authorization Rule

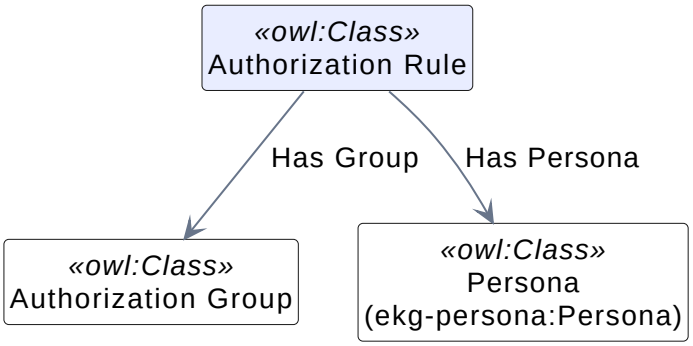


Diagram note

6.3 Properties

6.3.1 Has Group

Associates an authorization rule with an authorization group participating in that rule.

ekg-authorization:hasGroup	
https://ekgf.org/ontology/authorization#hasGroup	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-authorization:AuthorizationRule
rdfs:label	Has Group
rdfs:range	ekg-authorization:AuthorizationGroup

6.3.2 Has Persona

Associates an authorization rule with a persona to which the rule relates.

ekg-authorization:hasPersona	
https://ekgf.org/ontology/authorization#hasPersona	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-authorization:AuthorizationRule
rdfs:label	Has Persona
rdfs:range	ekg-persona:Persona

7 Capability Ontology

7.1 Overview

Capability Ontology

Identifier	Value
Ontology IRI / Namespace	`https://ekgf.org/ontology/capability#`
Prefix	`ekg-capability`

A vocabulary for capabilities and capability domains, with example capabilities for sharing data and managing data products. The current vocabulary does not define the relationships assigning capabilities to domains.

7.2 Classes

7.2.1 Capability

An ability represented in the capability model, such as the ability to share data or manage data products.

`ekg-capability:Capability`	
`https://ekgf.org/ontology/capability#Capability`	
RDF predicate	**Value**
`rdf:type`	`owl:Class`
`rdfs:label`	Capability

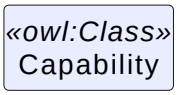


Diagram note

7.2.2 Capability Domain

A subject area used to organize capabilities in a capability model. This vocabulary does not yet declare the relationship assigning capabilities to a domain.

`ekg-capability:CapabilityDomain`	
`https://ekgf.org/ontology/capability#CapabilityDomain`	
RDF predicate	**Value**
`rdf:type`	`owl:Class`
`rdfs:label`	Capability Domain

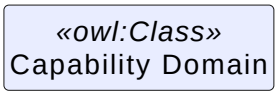


Diagram note

7.2.3 Sub-capability

A more specific capability represented at a subordinate level of a capability model. The current vocabulary declares it independently and does not assert a subclass relationship to Capability.

ekg-capability:SubCapability	
https://ekgf.org/ontology/capability#SubCapability	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Sub-capability

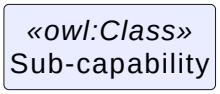


Diagram note

7.3 Named Individuals

7.3.1 Data sharing and marketplace

Instance of: ekg-capability:Capability

The example capability covering the sharing of data and its provision through a marketplace.

ekg-capability:Data_Sharing_and_Marketplace	
https://ekgf.org/ontology/capability#Data_Sharing_and_Marketplace	
RDF predicate	**Value**
dc:creator	Jacobus Geluk
dc:date	28 August 2023, 12:55:43 UTC
rdf:type	owl:NamedIndividual
rdf:type	ekg-capability:Capability
rdfs:label	Data sharing and marketplace

7.3.2 Designing and managing data products

Instance of: ekg-capability:SubCapability

The example sub-capability covering the design and management of data products.

ekg-capability:Designing_and_Managing_Data_Products	
https://ekgf.org/ontology/capability#Designing_and_Managing_Data_Products	
RDF predicate	**Value**
dc:creator	Jacobus Geluk
dc:date	28 August 2023, 12:56:51 UTC
rdf:type	owl:NamedIndividual
rdf:type	ekg-capability:SubCapability
rdfs:label	Designing and managing data products

8 Communication Ontology

8.1 Overview

Communication Ontology

Identifier	Value
Ontology IRI / Namespace	`https://ekgf.org/ontology/communication#`
Prefix	`ekg-communication`

Models communication recipients separately from people and organizations. Audiences, lists, tags, and campaign snapshots are collections with reified memberships. Consent, withdrawal, suppression, legal classification, and derivation evidence attach to memberships rather than being flattened onto recipients.

8.2 Classes

8.2.1 Audience

A targeting collection that may cluster recipients, endpoints, tags, lists, channel eligibility, and exclusion rules.

`ekg-communication:Audience`	
`https://ekgf.org/ontology/communication#Audience`	
RDF predicate	**Value**
`rdf:type`	`owl:Class`
`rdfs:label`	Audience
`rdfs:subClassOf`	ekg-communication:RecipientCollection

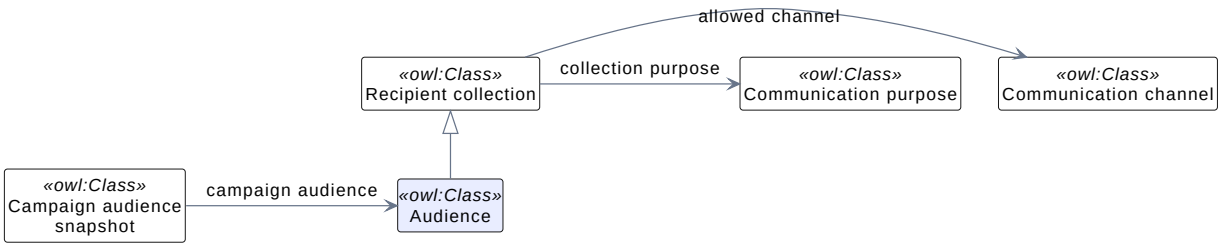


Diagram note

8.2.2 Audience membership

A membership in an audience.

`ekg-communication:AudienceMembership`	
`https://ekgf.org/ontology/communication#AudienceMembership`	
RDF predicate	**Value**
`rdf:type`	`owl:Class`
`rdfs:label`	Audience membership

ekg-communication:AudienceMembership	
https://ekgf.org/ontology/communication#AudienceMembership	
RDF predicate	**Value**
rdfs:subClassOf	ekg-communication:Membership

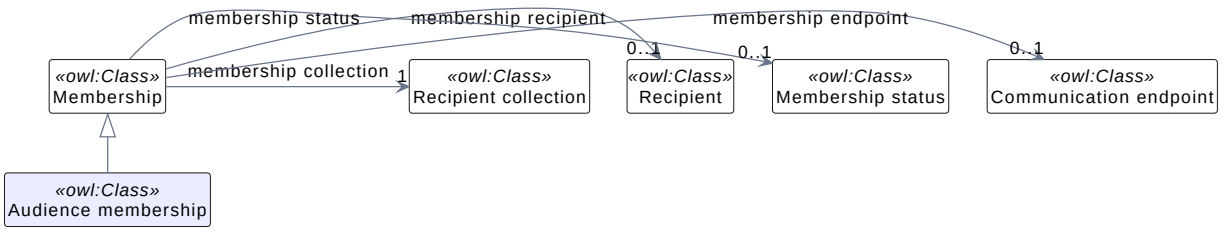


Diagram note

8.2.3 Campaign audience snapshot

An immutable collection of recipients or endpoints resolved for a specific campaign or send.

ekg-communication:CampaignAudienceSnapshot	
https://ekgf.org/ontology/communication#CampaignAudienceSnapshot	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Campaign audience snapshot
rdfs:subClassOf	ekg-communication:RecipientCollection

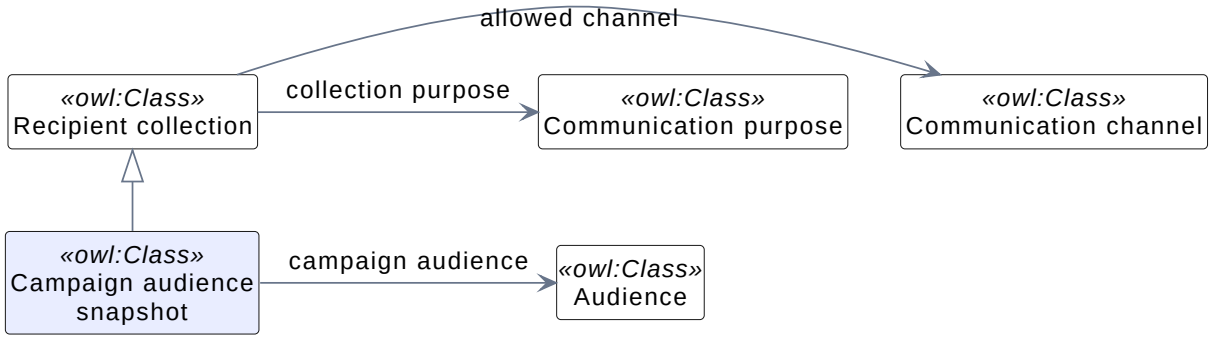


Diagram note

8.2.4 Campaign recipient

A frozen membership in a campaign audience snapshot, carrying per-send state and delivery correlation.

ekg-communication:CampaignRecipient	
https://ekgf.org/ontology/communication#CampaignRecipient	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Campaign recipient

ekg-communication:CampaignRecipient	
https://ekgf.org/ontology/communication#CampaignRecipient	
RDF predicate	**Value**
rdfs:subClassOf	ekg-communication:Membership

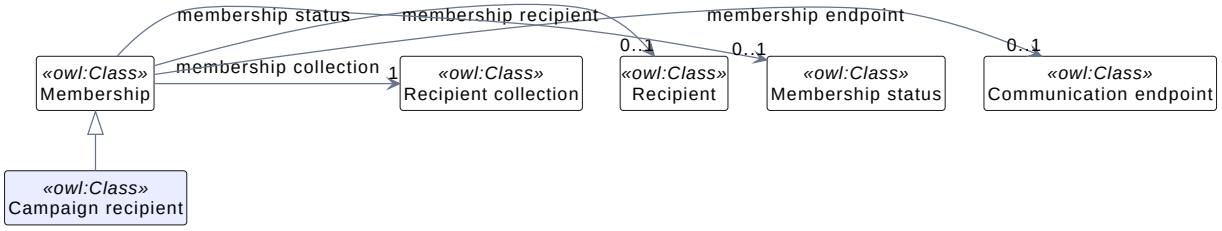


Diagram note

8.2.5 Communication attempt

One concrete attempt to communicate with a recipient or endpoint through a communication program.

ekg-communication:CommunicationAttempt	
https://ekgf.org/ontology/communication#CommunicationAttempt	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Communication attempt

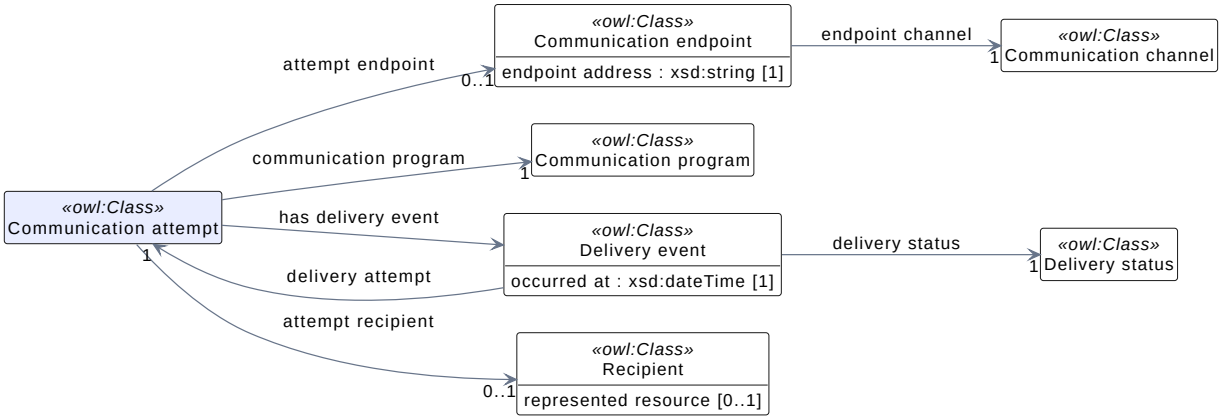


Diagram note

8.2.6 Communication channel

Controlled transport category such as email, SMS, voice, postal mail, push, webhook, or in-app notification.

ekg-communication:CommunicationChannel	
https://ekgf.org/ontology/communication#CommunicationChannel	
RDF predicate	**Value**
rdf:type	owl:Class

ekg-communication:CommunicationChannel	
https://ekgf.org/ontology/communication#CommunicationChannel	
RDF predicate	**Value**
rdfs:label	Communication channel

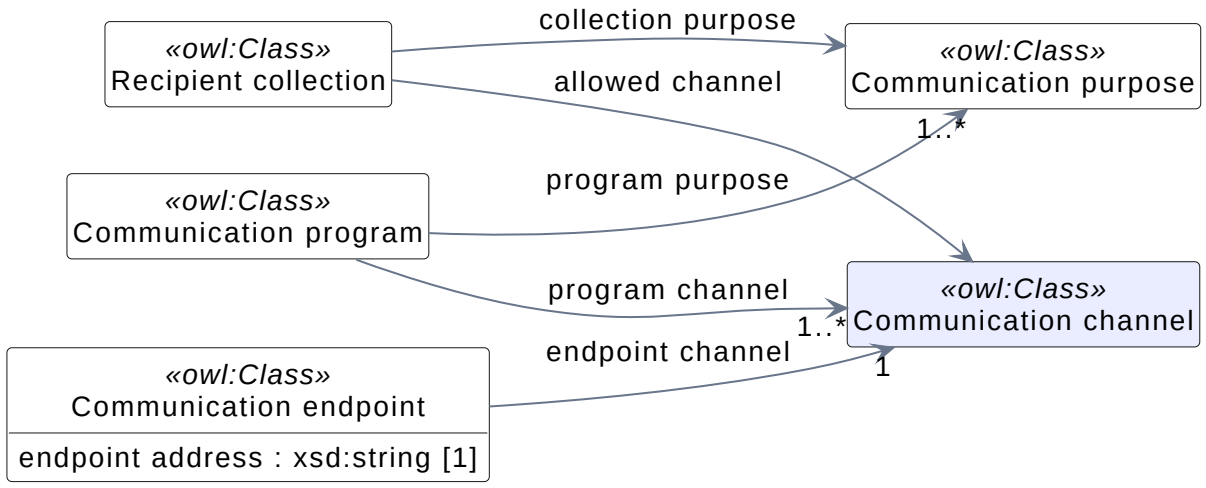


Diagram note

8.2.7 Communication endpoint

A concrete channel address or locator used to reach a recipient.

ekg-communication:CommunicationEndpoint	
https://ekgf.org/ontology/communication#CommunicationEndpoint	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Communication endpoint

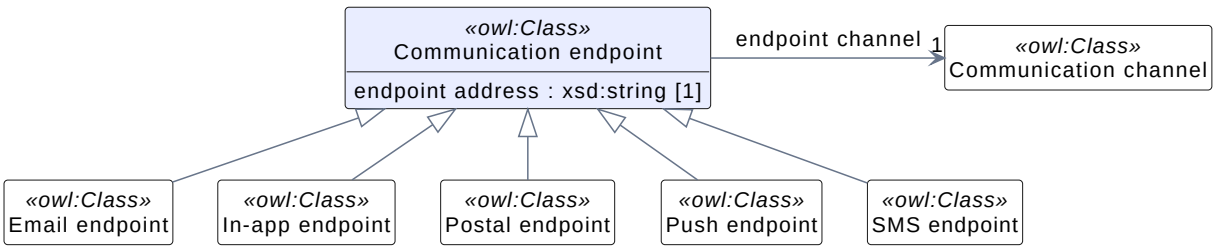


Diagram note

8.2.8 Communication program

A reusable communication surface combining channel, purpose, sender policy, authorization policy, and suppression rules.

ekg-communication:CommunicationProgram	
https://ekgf.org/ontology/communication#CommunicationProgram	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Communication program

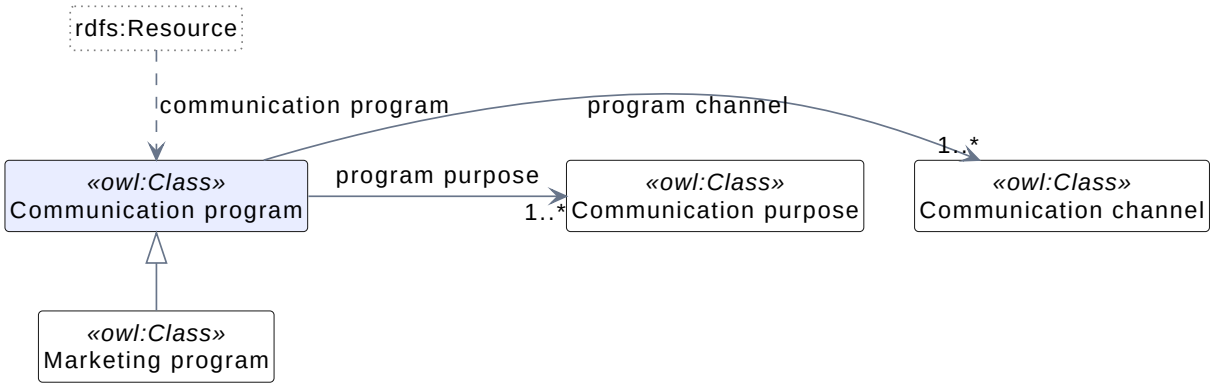


Diagram note

8.2.9 Communication purpose

Controlled purpose such as marketing, transactional, operational, legal notice, support, or security.

ekg-communication:CommunicationPurpose	
https://ekgf.org/ontology/communication#CommunicationPurpose	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Communication purpose

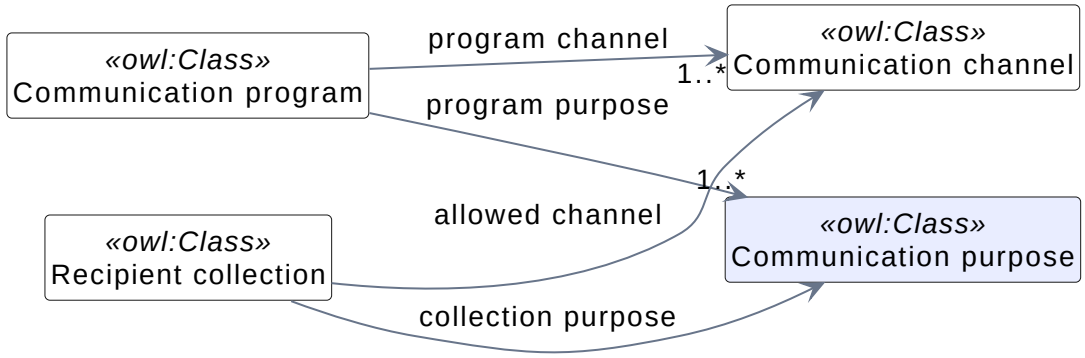


Diagram note

8.2.10 Delivery event

Provider or system event describing the outcome of a communication attempt.

ekg-communication:DeliveryEvent	
https://ekgf.org/ontology/communication#DeliveryEvent	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Delivery event

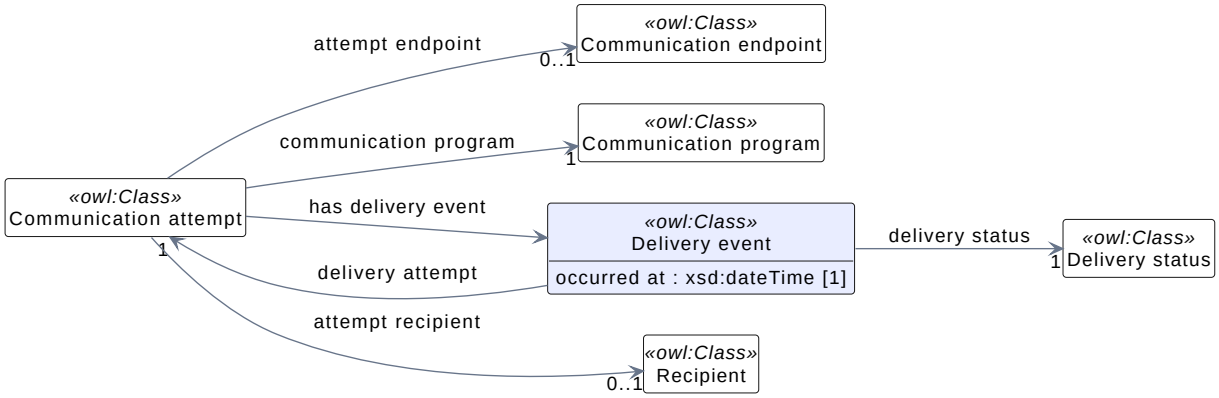


Diagram note

8.2.11 Delivery status

Controlled delivery status such as pending, sent, delivered, bounced, complained, suppressed, unsubscribed, failed, or skipped.

ekg-communication:DeliveryStatus	
https://ekgf.org/ontology/communication#DeliveryStatus	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Delivery status

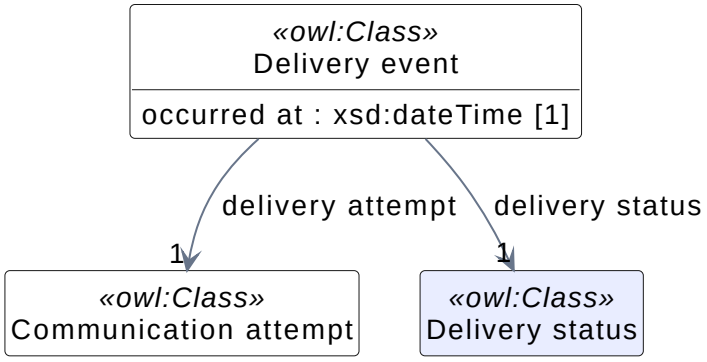


Diagram note

8.2.12 Email endpoint

An email address endpoint.

ekg-communication:EmailEndpoint	
https://ekgf.org/ontology/communication#EmailEndpoint	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Email endpoint
rdfs:subClassOf	ekg-communication:CommunicationEndpoint

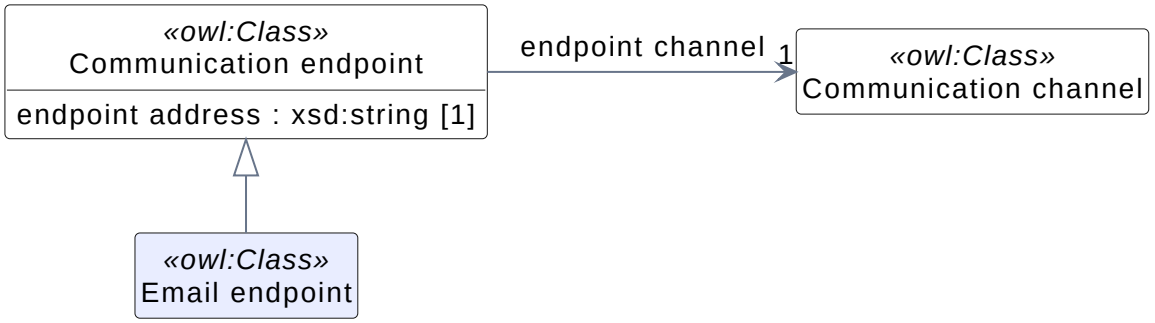


Diagram note

8.2.13 Email recipient

A recipient addressable by email. This class does not imply that the recipient is a person.

ekg-communication:EmailRecipient	
https://ekgf.org/ontology/communication#EmailRecipient	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Email recipient
rdfs:subClassOf	ekg-communication:Recipient

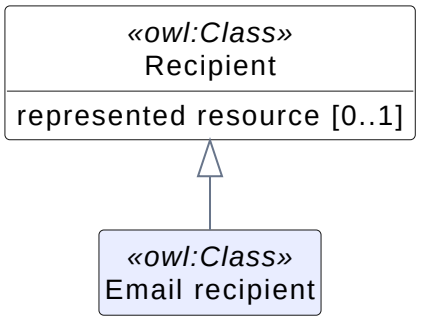


Diagram note

8.2.14 Group recipient

A recipient representing a group, team, role, or shared function.

ekg-communication:GroupRecipient	
https://ekgf.org/ontology/communication#GroupRecipient	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Group recipient
rdfs:subClassOf	ekg-communication:Recipient

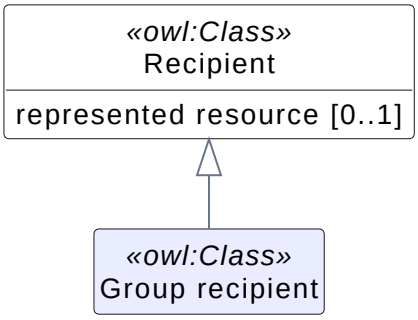


Diagram note

8.2.15 Human recipient

A recipient representing a natural person in a communication context.

ekg-communication:HumanRecipient	
https://ekgf.org/ontology/communication#HumanRecipient	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Human recipient
rdfs:subClassOf	ekg-communication:Recipient

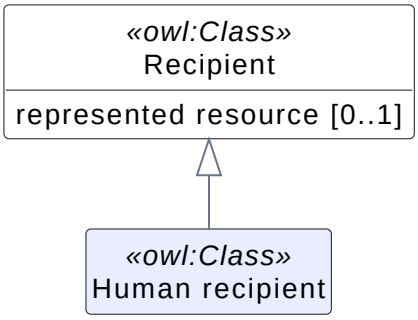


Diagram note

8.2.16 In-app endpoint

An account, user, workspace, or inbox locator for in-application communication.

ekg-communication:InAppEndpoint	
https://ekgf.org/ontology/communication#InAppEndpoint	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	In-app endpoint
rdfs:subClassOf	ekg-communication:CommunicationEndpoint

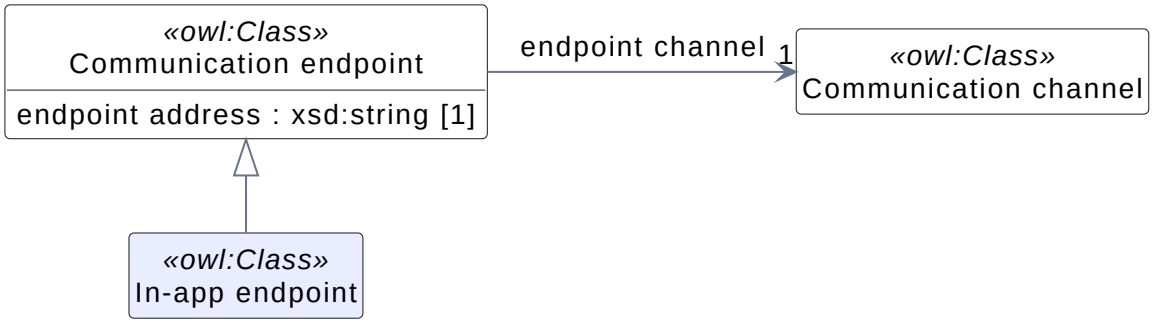


Diagram note

8.2.17 In-app recipient

A recipient addressable by in-application notification or message center delivery.

ekg-communication:InAppRecipient	
https://ekgf.org/ontology/communication#InAppRecipient	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	In-app recipient
rdfs:subClassOf	ekg-communication:Recipient

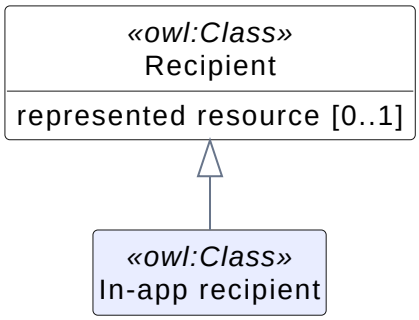


Diagram note

8.2.18 Legal basis

A controlled classification for recording a legal basis or exemption associated with a communication purpose, channel, recipient category, and jurisdiction. The classification alone does not establish permission to communicate.

ekg-communication:LegalBasis	
https://ekgf.org/ontology/communication#LegalBasis	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Legal basis

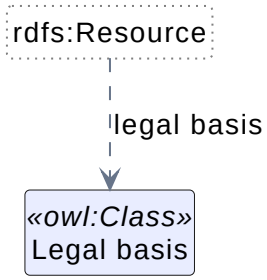


Diagram note

8.2.19 List membership

A membership in a recipient list or communication scope.

ekg-communication:ListMembership	
https://ekgf.org/ontology/communication#ListMembership	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	List membership
rdfs:subClassOf	ekg-communication:Membership

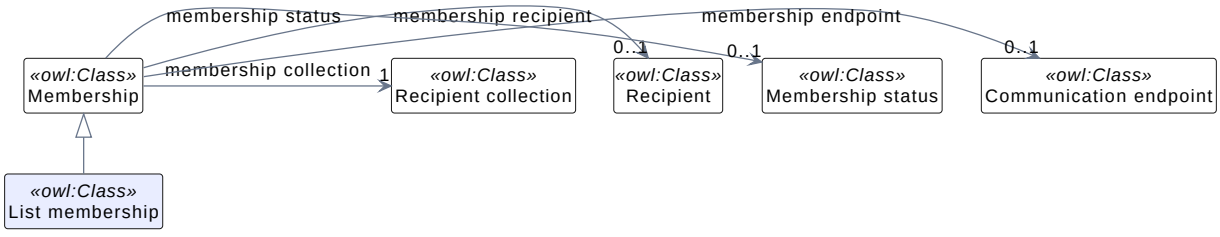


Diagram note

8.2.20 Mailbox recipient

A recipient representing a mailbox, shared inbox, alias, or distribution address.

ekg-communication:MailboxRecipient	
https://ekgf.org/ontology/communication#MailboxRecipient	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Mailbox recipient
rdfs:subClassOf	ekg-communication:Recipient

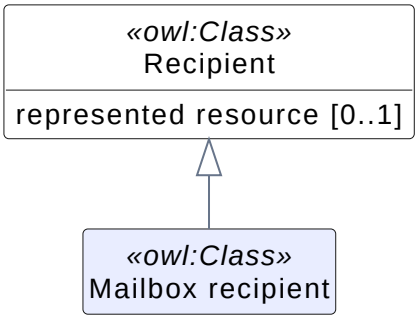


Diagram note

8.2.21 Marketing program

A communication program whose purpose is marketing.

ekg-communication:MarketingProgram	
https://ekgf.org/ontology/communication#MarketingProgram	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Marketing program
rdfs:subClassOf	ekg-communication:CommunicationProgram

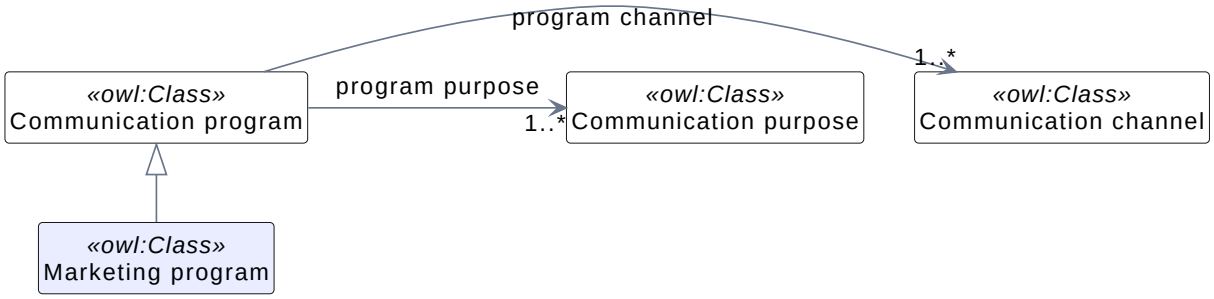


Diagram note

8.2.22 Membership

A reified edge connecting a recipient or endpoint to a recipient collection, with status and evidence.

ekg-communication:Membership	
https://ekgf.org/ontology/communication#Membership	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Membership

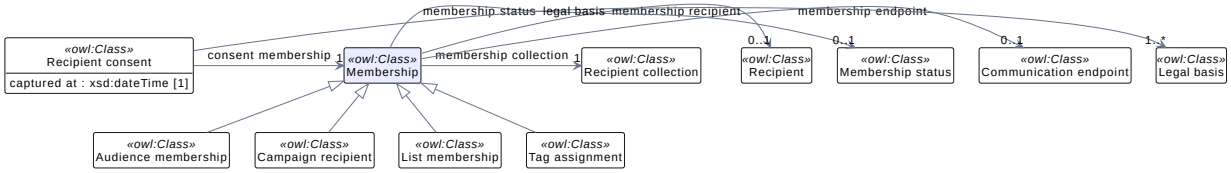


Diagram note

8.2.23 Membership derivation evidence

Evidence that a membership was derived from another collection, tag, rule, import, or operator action.

ekg-communication:MembershipDerivationEvidence	
https://ekgf.org/ontology/communication#MembershipDerivationEvidence	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Membership derivation evidence
rdfs:subClassOf	ekg-communication:MembershipEvidence

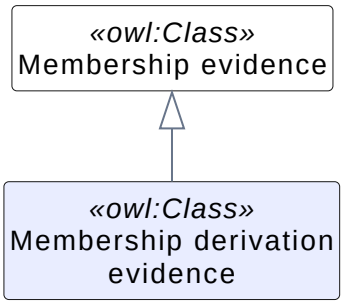


Diagram note

8.2.24 Membership event

Append-only event associated with a membership, such as add, remove, consent, withdrawal, or suppression.

ekg-communication:MembershipEvent	
https://ekgf.org/ontology/communication#MembershipEvent	
RDF predicate	**Value**
rdf:type	owl:Class

ekg-communication:MembershipEvent	
https://ekgf.org/ontology/communication#MembershipEvent	
RDF predicate	**Value**
rdfs:label	Membership event

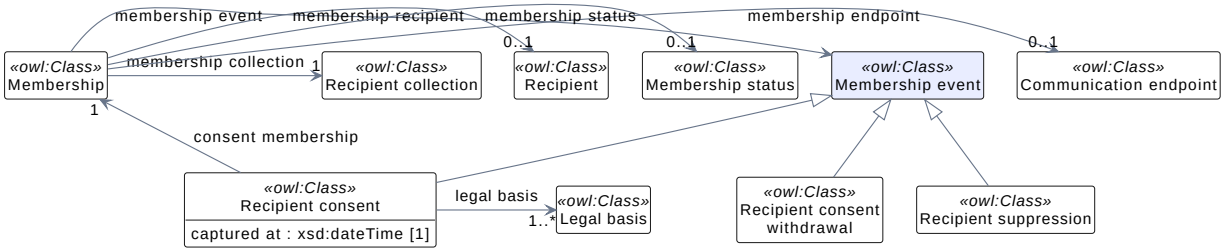


Diagram note

8.2.25 Membership evidence

Evidence or provenance supporting a membership, consent, withdrawal, suppression, or derivation.

ekg-communication:MembershipEvidence	
https://ekgf.org/ontology/communication#MembershipEvidence	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Membership evidence

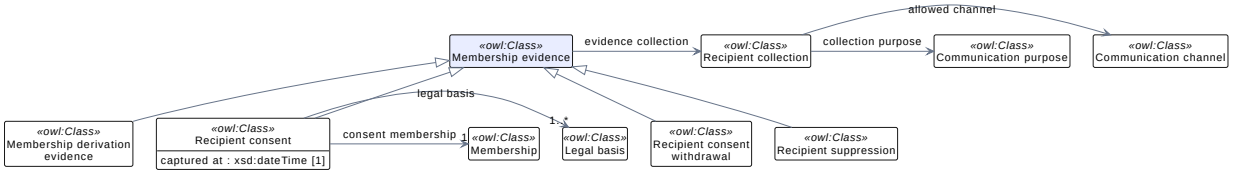


Diagram note

8.2.26 Membership status

Controlled current status of a membership.

ekg-communication:MembershipStatus	
https://ekgf.org/ontology/communication#MembershipStatus	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Membership status

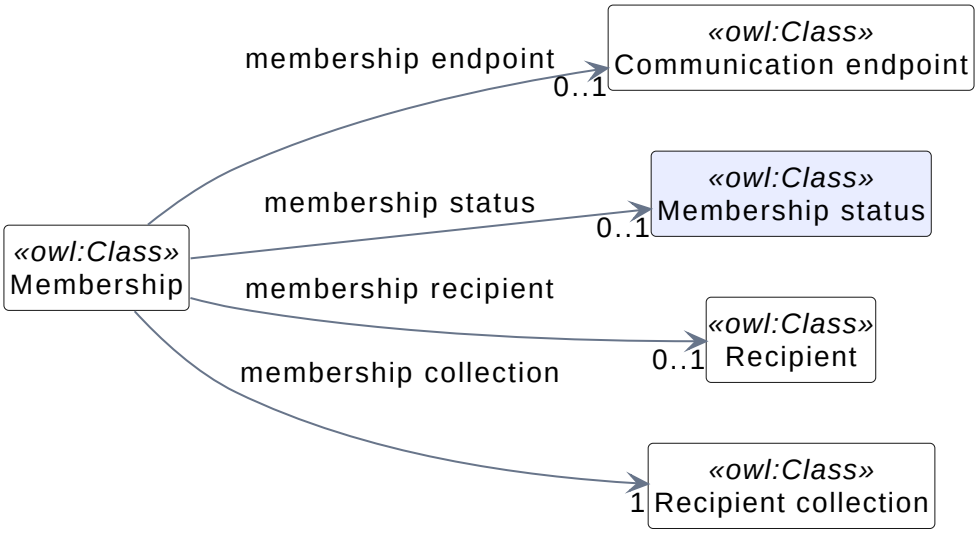


Diagram note

8.2.27 Organization recipient

A recipient representing an organization or other legal entity.

ekg-communication:OrganizationRecipient	
https://ekgf.org/ontology/communication#OrganizationRecipient	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Organization recipient
rdfs:subClassOf	ekg-communication:Recipient

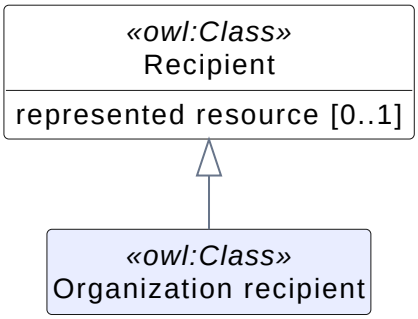


Diagram note

8.2.28 Postal endpoint

A postal address endpoint.

ekg-communication:PostalEndpoint	
https://ekgf.org/ontology/communication#PostalEndpoint	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Postal endpoint
rdfs:subClassOf	ekg-communication:CommunicationEndpoint

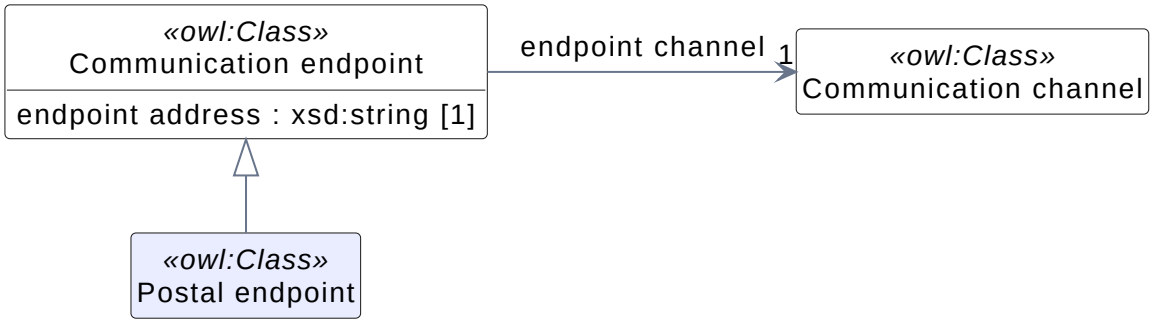


Diagram note

8.2.29 Postal recipient

A recipient addressable by postal mail or courier delivery.

ekg-communication:PostalRecipient	
https://ekgf.org/ontology/communication#PostalRecipient	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Postal recipient
rdfs:subClassOf	ekg-communication:Recipient

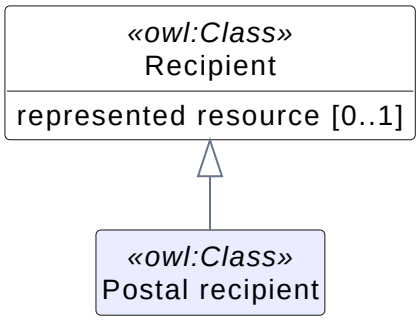


Diagram note

8.2.30 Push endpoint

A device, subscription, or token endpoint for push notification delivery.

ekg-communication:PushEndpoint	
https://ekgf.org/ontology/communication#PushEndpoint	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Push endpoint
rdfs:subClassOf	ekg-communication:CommunicationEndpoint

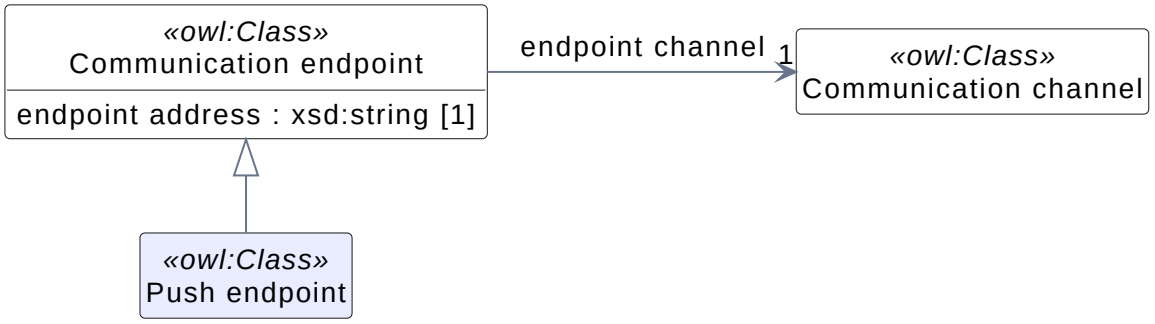


Diagram note

8.2.31 Push recipient

A recipient addressable by mobile, desktop, or browser push notification.

ekg-communication:PushRecipient	
https://ekgf.org/ontology/communication#PushRecipient	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Push recipient
rdfs:subClassOf	ekg-communication:Recipient

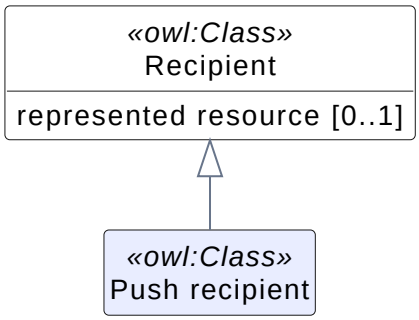


Diagram note

8.2.32 Recipient

An addressable communication target. A recipient may represent a person, organization, group, service, mailbox, or other operational target.

ekg-communication:Recipient	
https://ekgf.org/ontology/communication#Recipient	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Recipient

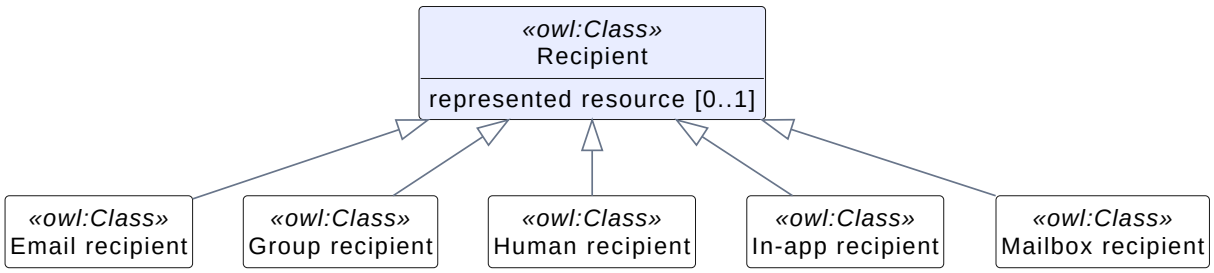


Diagram note

8.2.33 Recipient collection

A collection of recipients or endpoints. Lists, tags, audiences, and campaign snapshots are specializations.

ekg-communication:RecipientCollection	
https://ekgf.org/ontology/communication#RecipientCollection	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Recipient collection

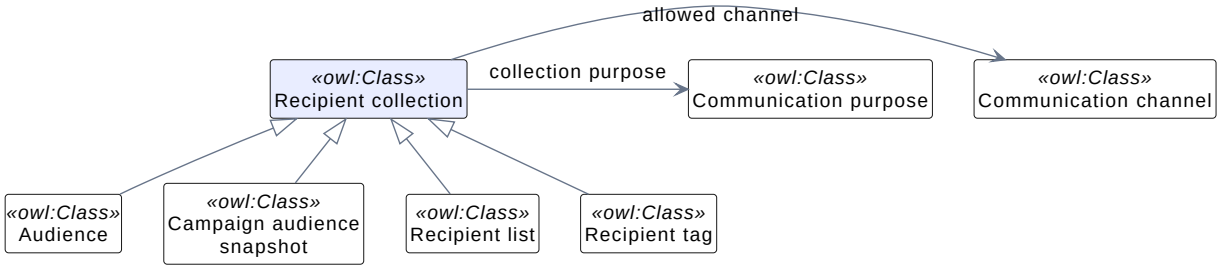


Diagram note

8.2.34 Recipient consent

Evidence that a recipient or endpoint has consented, or that another lawful basis was captured, for a membership scope.

ekg-communication:RecipientConsent	
https://ekgf.org/ontology/communication#RecipientConsent	
RDF predicate	**Value**
rdf:type	owl:Class

ekg-communication:RecipientConsent	
https://ekgf.org/ontology/communication#RecipientConsent	
RDF predicate	**Value**
rdfs:label	Recipient consent
rdfs:subClassOf	ekg-communication:MembershipEvent
rdfs:subClassOf	ekg-communication:MembershipEvidence

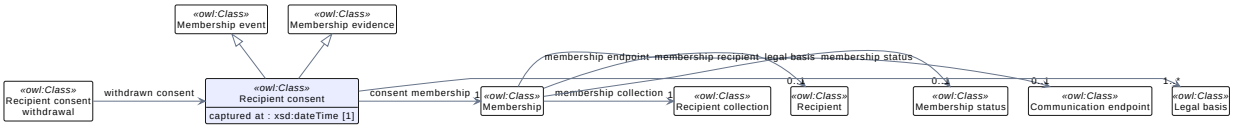


Diagram note

8.2.35 Recipient consent withdrawal

Evidence that a recipient withdrew consent or opted out for a membership scope. The membership may remain visible with a non-sendable status.

ekg-communication:RecipientConsentWithdrawal	
https://ekgf.org/ontology/communication#RecipientConsentWithdrawal	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Recipient consent withdrawal
rdfs:subClassOf	ekg-communication:MembershipEvent
rdfs:subClassOf	ekg-communication:MembershipEvidence

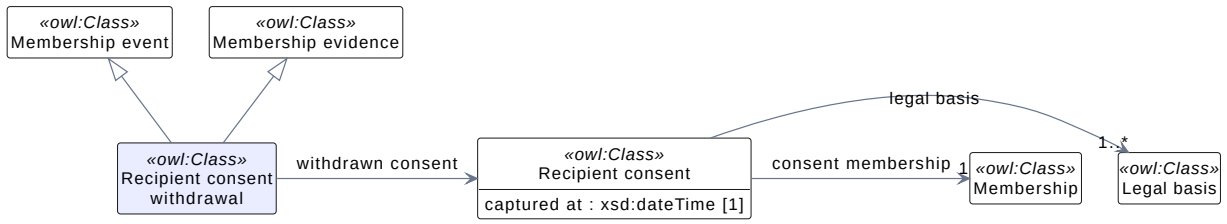


Diagram note

8.2.36 Recipient legal category

Controlled legal category such as natural person, employee, organization, technical service, group mailbox, or public authority.

ekg-communication:RecipientLegalCategory	
https://ekgf.org/ontology/communication#RecipientLegalCategory	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Recipient legal category

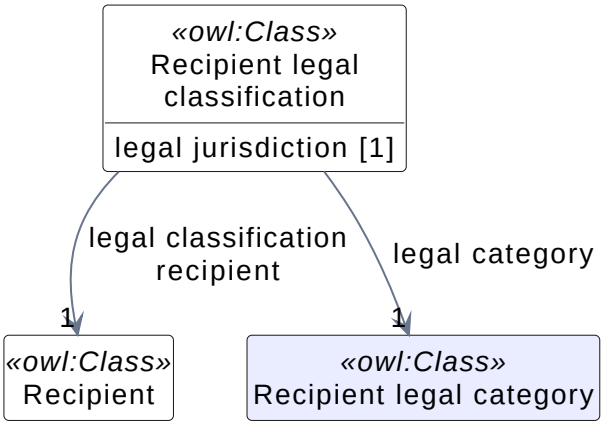


Diagram note

8.2.37 Recipient legal classification

Temporal legal classification of a recipient in a jurisdiction for communication, consent, and retention decisions.

ekg-communication:RecipientLegalClassification	
https://ekgf.org/ontology/communication#RecipientLegalClassification	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Recipient legal classification

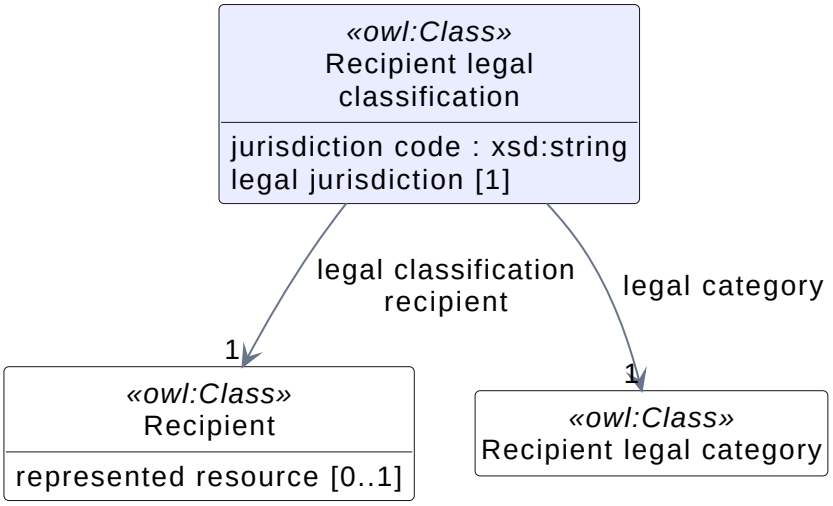


Diagram note

8.2.38 Recipient list

A named recipient collection normally used as a communication enrollment or list scope.

ekg-communication:RecipientList	
https://ekgf.org/ontology/communication#RecipientList	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Recipient list
rdfs:subClassOf	ekg-communication:RecipientCollection

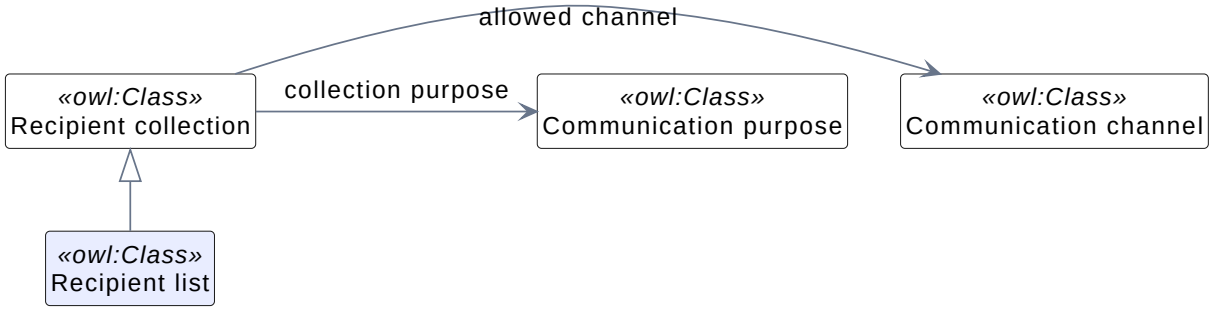


Diagram note

8.2.39 Recipient suppression

Evidence that future communication is suppressed because of unsubscribe, bounce, complaint, policy, or operator action.

ekg-communication:RecipientSuppression	
https://ekgf.org/ontology/communication#RecipientSuppression	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Recipient suppression
rdfs:subClassOf	ekg-communication:MembershipEvent
rdfs:subClassOf	ekg-communication:MembershipEvidence

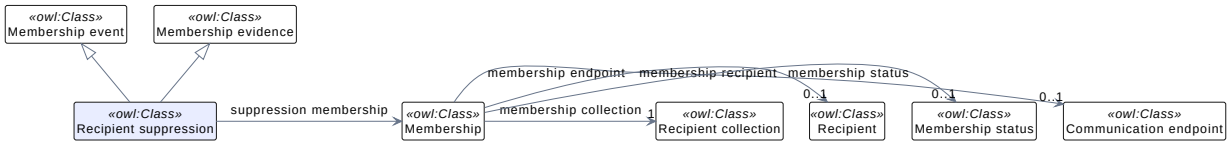


Diagram note

8.2.40 Recipient tag

A lightweight classification collection such as special-peer, reviewer, or customer.

ekg-communication:RecipientTag	
https://ekgf.org/ontology/communication#RecipientTag	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Recipient tag
rdfs:subClassOf	ekg-communication:RecipientCollection

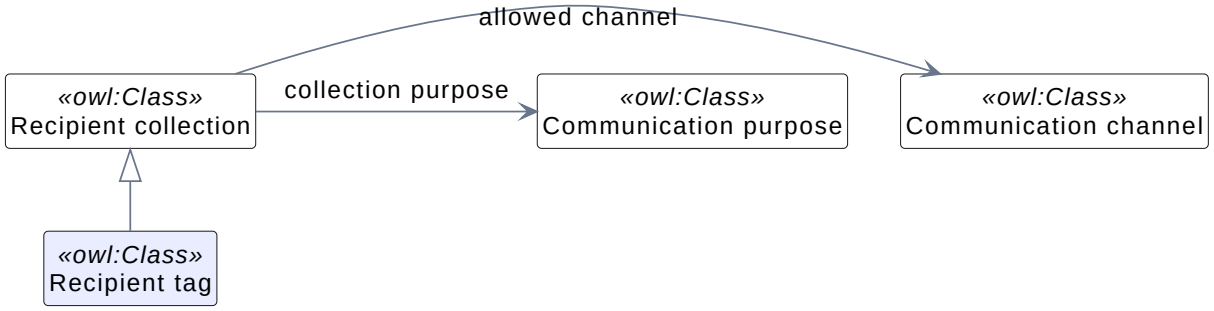


Diagram note

8.2.41 Service recipient

A recipient representing a technical system, webhook, service account, robot, or automation endpoint.

ekg-communication:ServiceRecipient	
https://ekgf.org/ontology/communication#ServiceRecipient	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Service recipient
rdfs:subClassOf	ekg-communication:Recipient

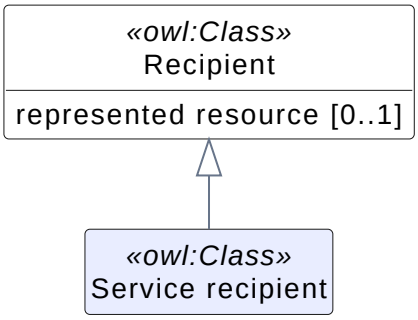


Diagram note

8.2.42 SMS endpoint

A phone-number endpoint for SMS or similar text-message delivery.

ekg-communication:SmsEndpoint	
https://ekgf.org/ontology/communication#SmsEndpoint	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	SMS endpoint
rdfs:subClassOf	ekg-communication:CommunicationEndpoint

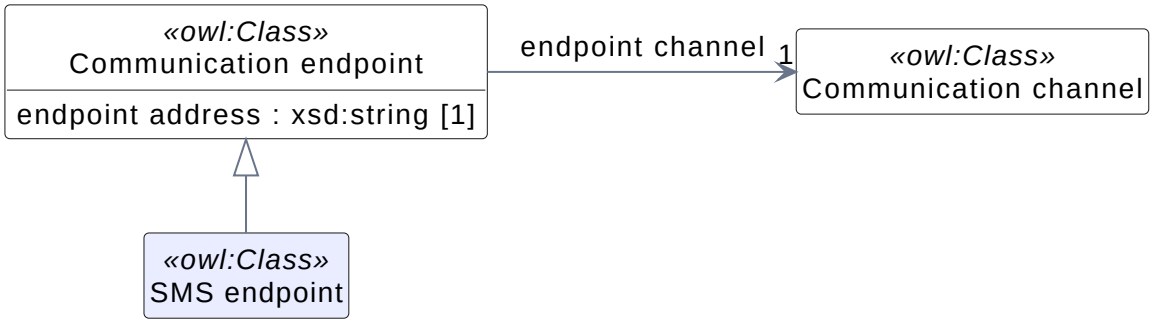


Diagram note

8.2.43 SMS recipient

A recipient addressable by SMS or similar text-message delivery.

ekg-communication:SmsRecipient	
https://ekgf.org/ontology/communication#SmsRecipient	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	SMS recipient
rdfs:subClassOf	ekg-communication:Recipient

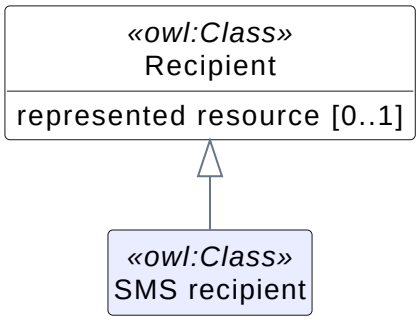


Diagram note

8.2.44 Tag assignment

A membership assigning a recipient or endpoint to a tag.

ekg-communication:TagAssignment	
https://ekgf.org/ontology/communication#TagAssignment	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Tag assignment
rdfs:subClassOf	ekg-communication:Membership

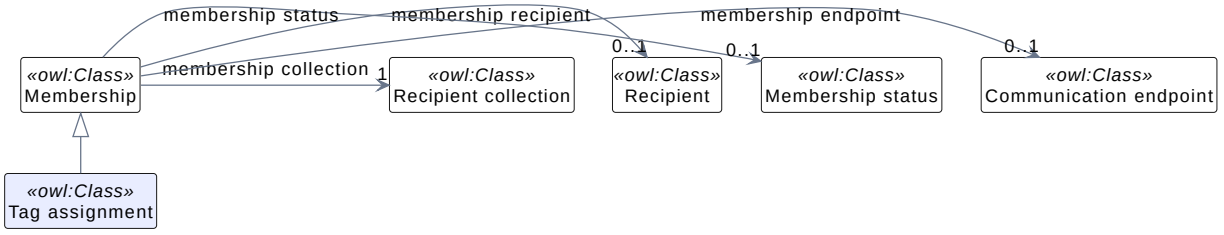


Diagram note

8.2.45 Voice endpoint

A phone-number endpoint for voice calls.

ekg-communication:VoiceEndpoint	
https://ekgf.org/ontology/communication#VoiceEndpoint	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Voice endpoint
rdfs:subClassOf	ekg-communication:CommunicationEndpoint

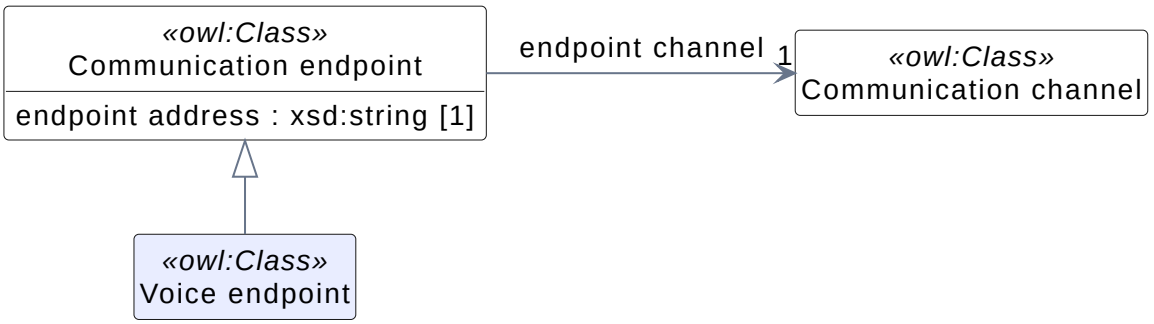


Diagram note

8.2.46 Voice recipient

A recipient addressable by voice call.

ekg-communication:VoiceRecipient	
https://ekgf.org/ontology/communication#VoiceRecipient	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Voice recipient
rdfs:subClassOf	ekg-communication:Recipient

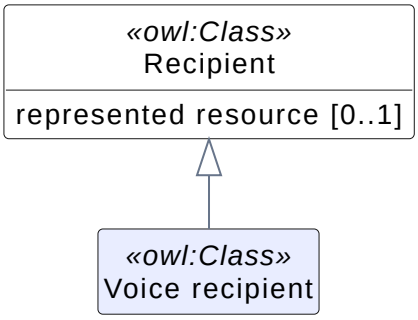


Diagram note

8.2.47 Webhook endpoint

A URL or callback endpoint for machine-to-machine communication.

ekg-communication:WebhookEndpoint	
https://ekgf.org/ontology/communication#WebhookEndpoint	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Webhook endpoint
rdfs:subClassOf	ekg-communication:CommunicationEndpoint

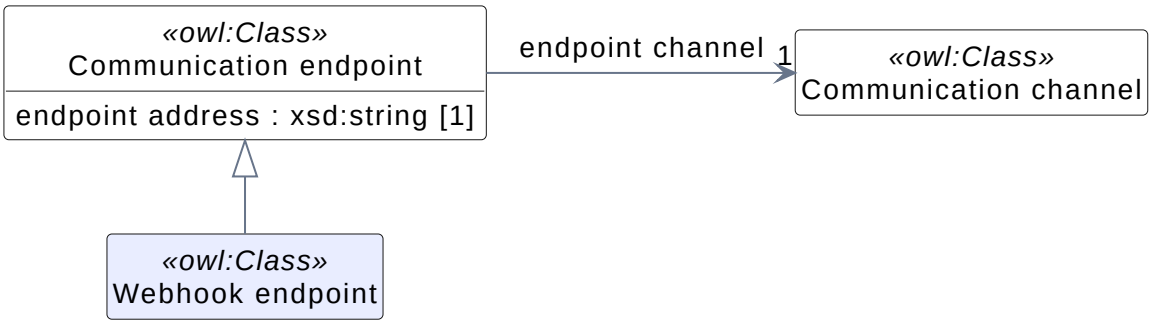


Diagram note

8.2.48 Webhook recipient

A recipient addressable by webhook or machine-to-machine callback.

ekg-communication:WebhookRecipient	
https://ekgf.org/ontology/communication#WebhookRecipient	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Webhook recipient
rdfs:subClassOf	ekg-communication:Recipient

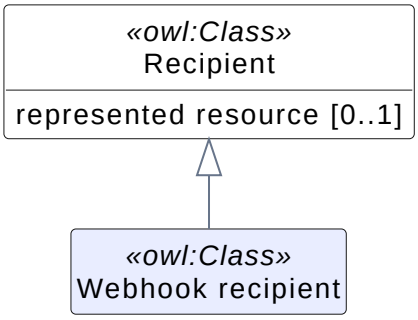


Diagram note

8.3 Properties

8.3.1 allowed channel

A channel allowed or expected for members of a collection.

ekg-communication:allowedChannel	
https://ekgf.org/ontology/communication#allowedChannel	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-communication:RecipientCollection
rdfs:label	allowed channel
rdfs:range	ekg-communication:CommunicationChannel

8.3.2 attempt endpoint

Concrete endpoint used by a communication attempt.

ekg-communication:attemptEndpoint	
https://ekgf.org/ontology/communication#attemptEndpoint	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-communication:CommunicationAttempt
rdfs:label	attempt endpoint
rdfs:range	ekg-communication:CommunicationEndpoint

8.3.3 attempt recipient

Recipient targeted by a communication attempt.

ekg-communication:attemptRecipient	
https://ekgf.org/ontology/communication#attemptRecipient	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-communication:CommunicationAttempt
rdfs:label	attempt recipient
rdfs:range	ekg-communication:Recipient

8.3.4 campaign audience

Source audience resolved into a campaign snapshot.

ekg-communication:campaignAudience	
https://ekgf.org/ontology/communication#campaignAudience	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-communication:CampaignAudienceSnapshot
rdfs:label	campaign audience
rdfs:range	ekg-communication:Audience

8.3.5 captured at

Timestamp when consent or lawful-basis evidence was captured.

ekg-communication:capturedAt	
https://ekgf.org/ontology/communication#capturedAt	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-communication:RecipientConsent
rdfs:label	captured at
rdfs:range	xsd:dateTime

8.3.6 collection purpose

The purpose associated with a list, audience, tag, or snapshot when the collection has a communication scope.

ekg-communication:collectionPurpose	
https://ekgf.org/ontology/communication#collectionPurpose	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-communication:RecipientCollection
rdfs:label	collection purpose
rdfs:range	ekg-communication:CommunicationPurpose

8.3.7 communication program

Communication program associated with a collection, snapshot, membership, or attempt.

ekg-communication:communicationProgram	
https://ekgf.org/ontology/communication#communicationProgram	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	rdfs:Resource
rdfs:label	communication program
rdfs:range	ekg-communication:CommunicationProgram

8.3.8 consent membership

Membership scope covered by a consent or lawful-basis record.

ekg-communication:consentMembership	
https://ekgf.org/ontology/communication#consentMembership	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-communication:RecipientConsent
rdfs:label	consent membership
rdfs:range	ekg-communication:Membership

8.3.9 delivery attempt

Communication attempt described by a delivery event.

ekg-communication:deliveryAttempt	
https://ekgf.org/ontology/communication#deliveryAttempt	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-communication:DeliveryEvent
rdfs:label	delivery attempt

ekg-communication:deliveryAttempt	
https://ekgf.org/ontology/communication#deliveryAttempt	
RDF predicate	**Value**
rdfs:range	ekg-communication:CommunicationAttempt

8.3.10 delivery status

Delivery outcome status for a delivery event.

ekg-communication:deliveryStatus	
https://ekgf.org/ontology/communication#deliveryStatus	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-communication:DeliveryEvent
rdfs:label	delivery status
rdfs:range	ekg-communication:DeliveryStatus

8.3.11 derived from membership

Links a derived membership to another membership that caused or justified it.

ekg-communication:derivedFromMembership	
https://ekgf.org/ontology/communication#derivedFromMembership	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-communication:Membership
rdfs:label	derived from membership
rdfs:range	ekg-communication:Membership

8.3.12 effective from

Timestamp from which a classification, membership status, consent, suppression, or policy is effective.

ekg-communication:effectiveFrom	
https://ekgf.org/ontology/communication#effectiveFrom	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	rdfs:Resource
rdfs:label	effective from
rdfs:range	xsd:dateTime

8.3.13 effective through

Timestamp through which a classification, membership status, consent, suppression, or policy is effective.

ekg-communication:effectiveThrough	
https://ekgf.org/ontology/communication#effectiveThrough	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	rdfs:Resource
rdfs:label	effective through
rdfs:range	xsd:dateTime

8.3.14 endpoint address

Channel-specific endpoint address, such as an email address, phone number, postal address key, token, URL, or account identifier.

ekg-communication:endpointAddress	
https://ekgf.org/ontology/communication#endpointAddress	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-communication:CommunicationEndpoint
rdfs:label	endpoint address
rdfs:range	xsd:string

8.3.15 endpoint channel

The transport channel used by an endpoint.

ekg-communication:endpointChannel	
https://ekgf.org/ontology/communication#endpointChannel	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-communication:CommunicationEndpoint
rdfs:label	endpoint channel
rdfs:range	ekg-communication:CommunicationChannel

8.3.16 evidence collection

Collection, tag, list, or audience referenced by membership evidence.

ekg-communication:evidenceCollection	
https://ekgf.org/ontology/communication#evidenceCollection	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty

ekg-communication:evidenceCollection	
https://ekgf.org/ontology/communication#evidenceCollection	
RDF predicate	**Value**
rdfs:domain	ekg-communication:MembershipEvidence
rdfs:label	evidence collection
rdfs:range	ekg-communication:RecipientCollection

8.3.17 evidence source

Source resource, import, operator action, rule, tag, form, webhook, or system fact that supports evidence.

ekg-communication:evidenceSource	
https://ekgf.org/ontology/communication#evidenceSource	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-communication:MembershipEvidence
rdfs:label	evidence source
rdfs:range	rdfs:Resource

8.3.18 has delivery event

Links an attempt to one delivery event.

ekg-communication:hasDeliveryEvent	
https://ekgf.org/ontology/communication#hasDeliveryEvent	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-communication:CommunicationAttempt
rdfs:label	has delivery event
rdfs:range	ekg-communication:DeliveryEvent

8.3.19 has endpoint

Links a recipient to one concrete channel endpoint.

ekg-communication:hasEndpoint	
https://ekgf.org/ontology/communication#hasEndpoint	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-communication:Recipient
rdfs:label	has endpoint
rdfs:range	ekg-communication:CommunicationEndpoint

8.3.20 has membership

Links a collection to a reified membership row.

ekg-communication:hasMembership	
https://ekgf.org/ontology/communication#hasMembership	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-communication:RecipientCollection
rdfs:label	has membership
rdfs:range	ekg-communication:Membership

8.3.21 jurisdiction code

Human-readable or ISO-style jurisdiction code used when no jurisdiction resource is available.

ekg-communication:jurisdictionCode	
https://ekgf.org/ontology/communication#jurisdictionCode	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-communication:RecipientLegalClassification
rdfs:label	jurisdiction code
rdfs:range	xsd:string

8.3.22 legal basis

Legal basis or exemption associated with consent evidence, a program, or a send decision.

ekg-communication:legalBasis	
https://ekgf.org/ontology/communication#legalBasis	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	rdfs:Resource
rdfs:label	legal basis
rdfs:range	ekg-communication:LegalBasis

8.3.23 legal category

Legal category assigned to a recipient in a jurisdiction and time interval.

ekg-communication:legalCategory	
https://ekgf.org/ontology/communication#legalCategory	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-communication:RecipientLegalClassification
rdfs:label	legal category
rdfs:range	ekg-communication:RecipientLegalCategory

8.3.24 legal classification recipient

Recipient classified for legal, consent, or retention decisions.

ekg-communication:legalClassificationRecipient	
https://ekgf.org/ontology/communication#legalClassificationRecipient	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-communication:RecipientLegalClassification
rdfs:label	legal classification recipient
rdfs:range	ekg-communication:Recipient

8.3.25 legal jurisdiction

Jurisdiction in which a legal classification applies.

ekg-communication:legalJurisdiction	
https://ekgf.org/ontology/communication#legalJurisdiction	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-communication:RecipientLegalClassification
rdfs:label	legal jurisdiction
rdfs:range	rdfs:Resource

8.3.26 membership collection

The collection that owns a membership.

ekg-communication:membershipCollection	
https://ekgf.org/ontology/communication#membershipCollection	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-communication:Membership
rdfs:label	membership collection

ekg-communication:membershipCollection	
https://ekgf.org/ontology/communication#membershipCollection	
RDF predicate	**Value**
rdfs:range	ekg-communication:RecipientCollection

8.3.27 membership endpoint

The endpoint that belongs to a collection when membership is channel- or endpoint-specific.

ekg-communication:membershipEndpoint	
https://ekgf.org/ontology/communication#membershipEndpoint	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-communication:Membership
rdfs:label	membership endpoint
rdfs:range	ekg-communication:CommunicationEndpoint

8.3.28 membership event

Links a membership to an append-only event.

ekg-communication:membershipEvent	
https://ekgf.org/ontology/communication#membershipEvent	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-communication:Membership
rdfs:label	membership event
rdfs:range	ekg-communication:MembershipEvent

8.3.29 membership evidence

Links a membership to supporting evidence.

ekg-communication:membershipEvidence	
https://ekgf.org/ontology/communication#membershipEvidence	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-communication:Membership
rdfs:label	membership evidence
rdfs:range	ekg-communication:MembershipEvidence

8.3.30 membership recipient

The recipient that belongs to a collection.

ekg-communication:membershipRecipient	
https://ekgf.org/ontology/communication#membershipRecipient	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-communication:Membership
rdfs:label	membership recipient
rdfs:range	ekg-communication:Recipient

8.3.31 membership status

Current status of a membership. Withdrawn consent and suppression are represented as status plus evidence, not by deleting the membership.

ekg-communication:membershipStatus	
https://ekgf.org/ontology/communication#membershipStatus	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-communication:Membership
rdfs:label	membership status
rdfs:range	ekg-communication:MembershipStatus

8.3.32 occurred at

Timestamp when an event, attempt, or delivery outcome occurred.

ekg-communication:occurredAt	
https://ekgf.org/ontology/communication#occurredAt	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	rdfs:Resource
rdfs:label	occurred at
rdfs:range	xsd:dateTime

8.3.33 policy version

Policy, terms, or notice version associated with consent, legal basis, suppression, or retention.

ekg-communication:policyVersion	
https://ekgf.org/ontology/communication#policyVersion	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty

ekg-communication:policyVersion	
https://ekgf.org/ontology/communication#policyVersion	
RDF predicate	**Value**
rdfs:domain	rdfs:Resource
rdfs:label	policy version
rdfs:range	xsd:string

8.3.34 program channel

The transport channel used by a communication program.

ekg-communication:programChannel	
https://ekgf.org/ontology/communication#programChannel	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-communication:CommunicationProgram
rdfs:label	program channel
rdfs:range	ekg-communication:CommunicationChannel

8.3.35 program purpose

The communication purpose served by a communication program.

ekg-communication:programPurpose	
https://ekgf.org/ontology/communication#programPurpose	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-communication:CommunicationProgram
rdfs:label	program purpose
rdfs:range	ekg-communication:CommunicationPurpose

8.3.36 provenance

Human-readable provenance string for imports, operator actions, forms, or system events.

ekg-communication:provenance	
https://ekgf.org/ontology/communication#provenance	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	rdfs:Resource
rdfs:label	provenance
rdfs:range	xsd:string

8.3.37 provider message id

Provider-side message identifier for an attempt or delivery event.

ekg-communication:providerMessageId	
https://ekgf.org/ontology/communication#providerMessageId	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	rdfs:Resource
rdfs:label	provider message id
rdfs:range	xsd:string

8.3.38 provider recipient id

Provider-side recipient identifier for an attempt or delivery event.

ekg-communication:providerRecipientId	
https://ekgf.org/ontology/communication#providerRecipientId	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	rdfs:Resource
rdfs:label	provider recipient id
rdfs:range	xsd:string

8.3.39 recorded at

Timestamp when a fact or evidence record was recorded.

ekg-communication:recordedAt	
https://ekgf.org/ontology/communication#recordedAt	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	rdfs:Resource
rdfs:label	recorded at
rdfs:range	xsd:dateTime

8.3.40 represented resource

Links a communication recipient to the person, organization, service, mailbox, account, or other resource it represents, when known.

ekg-communication:representedResource	
https://ekgf.org/ontology/communication#representedResource	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-communication:Recipient
rdfs:label	represented resource
rdfs:range	rdfs:Resource

8.3.41 retention period

Duration for which a fact, evidence record, or communication event should be retained.

ekg-communication:retentionPeriod	
https://ekgf.org/ontology/communication#retentionPeriod	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	rdfs:Resource
rdfs:label	retention period
rdfs:range	xsd:duration

8.3.42 retention policy

Retention policy resource applicable to a classification, membership, evidence record, attempt, or delivery event.

ekg-communication:retentionPolicy	
https://ekgf.org/ontology/communication#retentionPolicy	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	rdfs:Resource
rdfs:label	retention policy
rdfs:range	rdfs:Resource

8.3.43 suppression membership

Membership scope suppressed by a suppression event.

ekg-communication:suppressionMembership	
https://ekgf.org/ontology/communication#suppressionMembership	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-communication:RecipientSuppression

ekg-communication:suppressionMembership	
https://ekgf.org/ontology/communication#suppressionMembership	
RDF predicate	**Value**
rdfs:label	suppression membership
rdfs:range	ekg-communication:Membership

8.3.44 withdrawn consent

Consent record withdrawn by a later withdrawal event.

ekg-communication:withdrawnConsent	
https://ekgf.org/ontology/communication#withdrawnConsent	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-communication:RecipientConsentWithdrawal
rdfs:label	withdrawn consent
rdfs:range	ekg-communication:RecipientConsent

8.4 Named Individuals

8.4.1 Active membership

Instance of: ekg-communication:MembershipStatus

The membership-status value for a membership currently classified as active. Channel and purpose eligibility still depend on the applicable program and policy.

ekg-communication:ActiveMembership	
https://ekgf.org/ontology/communication#ActiveMembership	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:MembershipStatus
rdfs:label	Active membership

8.4.2 Bounced

Instance of: ekg-communication:DeliveryStatus

The delivery-status value recording that an attempt was reported as bounced.

ekg-communication:Bounced	
https://ekgf.org/ontology/communication#Bounced	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:DeliveryStatus

ekg-communication:Bounced	
https://ekgf.org/ontology/communication#Bounced	
RDF predicate	**Value**
rdfs:label	Bounced

8.4.3 Complained

Instance of: ekg-communication:DeliveryStatus

The delivery-status value recording a complaint associated with an attempt.

ekg-communication:Complained	
https://ekgf.org/ontology/communication#Complained	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:DeliveryStatus
rdfs:label	Complained

8.4.4 Consent legal basis

Instance of: ekg-communication:LegalBasis

A controlled legal-basis value for recording a classification based on consent. The value records the asserted classification; selecting it does not itself establish permission to communicate.

ekg-communication:ConsentLegalBasis	
https://ekgf.org/ontology/communication#ConsentLegalBasis	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:LegalBasis
rdfs:label	Consent legal basis

8.4.5 Consent withdrawn membership

Instance of: ekg-communication:MembershipStatus

The membership-status value recording that consent associated with the membership has been withdrawn.

ekg-communication:ConsentWithdrawnMembership	
https://ekgf.org/ontology/communication#ConsentWithdrawnMembership	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:MembershipStatus
rdfs:label	Consent withdrawn membership

8.4.6 Contract legal basis

Instance of: ekg-communication:LegalBasis

A controlled legal-basis value for recording a classification based on a contract. The value records the asserted classification; selecting it does not itself establish permission to communicate.

ekg-communication:ContractLegalBasis	
https://ekgf.org/ontology/communication#ContractLegalBasis	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:LegalBasis
rdfs:label	Contract legal basis

8.4.7 Delivered

Instance of: ekg-communication:DeliveryStatus

The delivery-status value recording that an attempt was reported as delivered.

ekg-communication:Delivered	
https://ekgf.org/ontology/communication#Delivered	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:DeliveryStatus
rdfs:label	Delivered

8.4.8 Email channel

Instance of: ekg-communication:CommunicationChannel

The communication-channel value identifying email transport.

ekg-communication:EmailChannel	
https://ekgf.org/ontology/communication#EmailChannel	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:CommunicationChannel
rdfs:label	Email channel

8.4.9 Employee category

Instance of: ekg-communication:RecipientLegalCategory

The recipient legal-category value identifying an employee in a recorded jurisdictional classification.

ekg-communication:EmployeeCategory	
https://ekgf.org/ontology/communication#EmployeeCategory	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:RecipientLegalCategory
rdfs:label	Employee category

8.4.10 Employee operational legal basis

Instance of: ekg-communication:LegalBasis

A controlled legal-basis value for recording a classification based on an employee operational context. The value records the asserted classification; selecting it does not itself establish permission to communicate.

ekg-communication:EmployeeOperationalLegalBasis	
https://ekgf.org/ontology/communication#EmployeeOperationalLegalBasis	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:LegalBasis
rdfs:label	Employee operational legal basis

8.4.11 Failed

Instance of: ekg-communication:DeliveryStatus

The delivery-status value recording that an attempt failed.

ekg-communication:Failed	
https://ekgf.org/ontology/communication#Failed	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:DeliveryStatus
rdfs:label	Failed

8.4.12 Group mailbox category

Instance of: ekg-communication:RecipientLegalCategory

The recipient legal-category value identifying a shared or group mailbox in a recorded jurisdictional classification.

ekg-communication:GroupMailboxCategory	
https://ekgf.org/ontology/communication#GroupMailboxCategory	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual

ekg-communication:GroupMailboxCategory	
https://ekgf.org/ontology/communication#GroupMailboxCategory	
RDF predicate	**Value**
rdf:type	ekg-communication:RecipientLegalCategory
rdfs:label	Group mailbox category

8.4.13 In-app notification channel

Instance of: ekg-communication:CommunicationChannel

The communication-channel value identifying an in-application notification or message-center transport.

ekg-communication:InAppNotificationChannel	
https://ekgf.org/ontology/communication#InAppNotificationChannel	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:CommunicationChannel
rdfs:label	In-app notification channel

8.4.14 Inactive membership

Instance of: ekg-communication:MembershipStatus

The membership-status value for a membership currently classified as inactive.

ekg-communication:InactiveMembership	
https://ekgf.org/ontology/communication#InactiveMembership	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:MembershipStatus
rdfs:label	Inactive membership

8.4.15 Legal notice purpose

Instance of: ekg-communication:CommunicationPurpose

The communication-purpose value identifying communications classified as legal notices.

ekg-communication:LegalNoticePurpose	
https://ekgf.org/ontology/communication#LegalNoticePurpose	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:CommunicationPurpose
rdfs:label	Legal notice purpose

8.4.16 Legal obligation legal basis

Instance of: ekg-communication:LegalBasis

A controlled legal-basis value for recording a classification based on a legal obligation. The value records the asserted classification; selecting it does not itself establish permission to communicate.

ekg-communication:LegalObligationLegalBasis	
https://ekgf.org/ontology/communication#LegalObligationLegalBasis	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:LegalBasis
rdfs:label	Legal obligation legal basis

8.4.17 Legitimate interest legal basis

Instance of: ekg-communication:LegalBasis

A controlled legal-basis value for recording a classification based on legitimate interests. The value records the asserted classification; selecting it does not itself establish permission to communicate.

ekg-communication:LegitimateInterestLegalBasis	
https://ekgf.org/ontology/communication#LegitimateInterestLegalBasis	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:LegalBasis
rdfs:label	Legitimate interest legal basis

8.4.18 Marketing purpose

Instance of: ekg-communication:CommunicationPurpose

The communication-purpose value identifying marketing communications.

ekg-communication:MarketingPurpose	
https://ekgf.org/ontology/communication#MarketingPurpose	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:CommunicationPurpose
rdfs:label	Marketing purpose

8.4.19 Natural person category

Instance of: ekg-communication:RecipientLegalCategory

The recipient legal-category value identifying a natural person in a recorded jurisdictional classification.

ekg-communication:NaturalPersonCategory	
https://ekgf.org/ontology/communication#NaturalPersonCategory	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:RecipientLegalCategory
rdfs:label	Natural person category

8.4.20 Operational purpose

Instance of: ekg-communication:CommunicationPurpose

The communication-purpose value identifying operational communications.

ekg-communication:OperationalPurpose	
https://ekgf.org/ontology/communication#OperationalPurpose	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:CommunicationPurpose
rdfs:label	Operational purpose

8.4.21 Organization category

Instance of: ekg-communication:RecipientLegalCategory

The recipient legal-category value identifying an organization in a recorded jurisdictional classification.

ekg-communication:OrganizationCategory	
https://ekgf.org/ontology/communication#OrganizationCategory	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:RecipientLegalCategory
rdfs:label	Organization category

8.4.22 Pending

Instance of: ekg-communication:DeliveryStatus

The delivery-status value recording an attempt awaiting further processing or an outcome.

ekg-communication:Pending	
https://ekgf.org/ontology/communication#Pending	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:DeliveryStatus
rdfs:label	Pending

8.4.23 Postal mail channel

Instance of: ekg-communication:CommunicationChannel

The communication-channel value identifying postal-mail transport.

ekg-communication:PostalMailChannel	
https://ekgf.org/ontology/communication#PostalMailChannel	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:CommunicationChannel
rdfs:label	Postal mail channel

8.4.24 Public authority category

Instance of: ekg-communication:RecipientLegalCategory

The recipient legal-category value identifying a public authority in a recorded jurisdictional classification.

ekg-communication:PublicAuthorityCategory	
https://ekgf.org/ontology/communication#PublicAuthorityCategory	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:RecipientLegalCategory
rdfs:label	Public authority category

8.4.25 Push notification channel

Instance of: ekg-communication:CommunicationChannel

The communication-channel value identifying push-notification transport.

ekg-communication:PushNotificationChannel	
https://ekgf.org/ontology/communication#PushNotificationChannel	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:CommunicationChannel
rdfs:label	Push notification channel

8.4.26 Security purpose

Instance of: ekg-communication:CommunicationPurpose

The communication-purpose value identifying security communications.

ekg-communication:SecurityPurpose	
https://ekgf.org/ontology/communication#SecurityPurpose	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:CommunicationPurpose
rdfs:label	Security purpose

8.4.27 Sent

Instance of: ekg-communication:DeliveryStatus

The delivery-status value recording that an attempt was sent, separately from evidence of delivery.

ekg-communication:Sent	
https://ekgf.org/ontology/communication#Sent	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:DeliveryStatus
rdfs:label	Sent

8.4.28 Skipped

Instance of: ekg-communication:DeliveryStatus

The delivery-status value recording that an intended attempt was skipped rather than sent.

ekg-communication:Skipped	
https://ekgf.org/ontology/communication#Skipped	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:DeliveryStatus
rdfs:label	Skipped

8.4.29 SMS channel

Instance of: ekg-communication:CommunicationChannel

The communication-channel value identifying SMS transport.

ekg-communication:SmsChannel	
https://ekgf.org/ontology/communication#SmsChannel	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:CommunicationChannel
rdfs:label	SMS channel

8.4.30 Support purpose

Instance of: ekg-communication:CommunicationPurpose

The communication-purpose value identifying support communications.

ekg-communication:SupportPurpose	
https://ekgf.org/ontology/communication#SupportPurpose	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:CommunicationPurpose
rdfs:label	Support purpose

8.4.31 Suppressed

Instance of: ekg-communication:DeliveryStatus

The delivery-status value recording that an attempt was suppressed by a policy or suppression decision.

ekg-communication:Suppressed	
https://ekgf.org/ontology/communication#Suppressed	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:DeliveryStatus
rdfs:label	Suppressed

8.4.32 Suppressed membership

Instance of: ekg-communication:MembershipStatus

The membership-status value recording that communication under the membership is suppressed.

ekg-communication:SuppressedMembership	
https://ekgf.org/ontology/communication#SuppressedMembership	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:MembershipStatus
rdfs:label	Suppressed membership

8.4.33 Technical service category

Instance of: ekg-communication:RecipientLegalCategory

The recipient legal-category value identifying a technical service in a recorded jurisdictional classification.

ekg-communication:TechnicalServiceCategory	
https://ekgf.org/ontology/communication#TechnicalServiceCategory	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:RecipientLegalCategory
rdfs:label	Technical service category

8.4.34 Technical service legal basis

Instance of: ekg-communication:LegalBasis

A controlled legal-basis value for recording a classification based on a technical-service context. The value records the asserted classification; selecting it does not itself establish permission to communicate.

ekg-communication:TechnicalServiceLegalBasis	
https://ekgf.org/ontology/communication#TechnicalServiceLegalBasis	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:LegalBasis
rdfs:label	Technical service legal basis

8.4.35 Transactional purpose

Instance of: ekg-communication:CommunicationPurpose

The communication-purpose value identifying communications associated with a transaction.

ekg-communication:TransactionalPurpose	
https://ekgf.org/ontology/communication#TransactionalPurpose	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:CommunicationPurpose
rdfs:label	Transactional purpose

8.4.36 Unsubscribed

Instance of: ekg-communication:DeliveryStatus

The delivery-status value recording an unsubscribe event associated with an attempt.

ekg-communication:Unsubscribed	
https://ekgf.org/ontology/communication#Unsubscribed	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:DeliveryStatus

ekg-communication:Unsubscribed	
https://ekgf.org/ontology/communication#Unsubscribed	
RDF predicate	**Value**
rdfs:label	Unsubscribed

8.4.37 Voice call channel

Instance of: ekg-communication:CommunicationChannel

The communication-channel value identifying voice-call transport.

ekg-communication:VoiceCallChannel	
https://ekgf.org/ontology/communication#VoiceCallChannel	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:CommunicationChannel
rdfs:label	Voice call channel

8.4.38 Webhook channel

Instance of: ekg-communication:CommunicationChannel

The communication-channel value identifying a webhook or machine-to-machine callback transport.

ekg-communication:WebhookChannel	
https://ekgf.org/ontology/communication#WebhookChannel	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:CommunicationChannel
rdfs:label	Webhook channel

8.5 Shapes

8.5.1 Communication Attempt Shape

Requires exactly one communication program and at least one recipient or endpoint, with at most one of each target type.

ekg-communication:CommunicationAttemptShape	
https://ekgf.org/ontology/communication#CommunicationAttemptShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Communication Attempt Shape
sh:or	((sh:property: (sh:minCount: 1; sh:path: ekg-communication:attemptRecipient)), (sh:property: (sh:minCount: 1; sh:path: ekg-communication:attemptEndpoint)))

ekg-communication:CommunicationAttemptShape	
https://ekgf.org/ontology/communication#CommunicationAttemptShape	
RDF predicate	**Value**
sh:property	(sh:class: ekg-communication:CommunicationEndpoint; sh:maxCount: 1; sh:name: Attempt endpoint; sh:path: ekg-communication:attemptEndpoint)
sh:property	(sh:class: ekg-communication:Recipient; sh:maxCount: 1; sh:name: Attempt recipient; sh:path: ekg-communication:attemptRecipient)
sh:property	(sh:class: ekg-communication:CommunicationProgram; sh:maxCount: 1; sh:minCount: 1; sh:name: Communication program; sh:path: ekg-communication:communicationProgram)
sh:targetClass	ekg-communication:CommunicationAttempt

8.5.2 Communication Endpoint Shape

Validates a communication endpoint's channel and address literals.

ekg-communication:CommunicationEndpointShape	
https://ekgf.org/ontology/communication#CommunicationEndpointShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Communication Endpoint Shape
sh:property	(sh:datatype: xsd:string; sh:maxCount: 1; sh:minCount: 1; sh:name: Endpoint address; sh:path: ekg-communication:endpointAddress)
sh:property	(sh:class: ekg-communication:CommunicationChannel; sh:maxCount: 1; sh:minCount: 1; sh:name: Endpoint channel; sh:path: ekg-communication:endpointChannel)
sh:targetClass	ekg-communication:CommunicationEndpoint

8.5.3 Communication Program Shape

Validates a communication program's declared channel and purpose.

ekg-communication:CommunicationProgramShape	
https://ekgf.org/ontology/communication#CommunicationProgramShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Communication Program Shape
sh:property	(sh:class: ekg-communication:CommunicationChannel; sh:minCount: 1; sh:name: Program channel; sh:path: ekg-communication:programChannel)
sh:property	(sh:class: ekg-communication:CommunicationPurpose; sh:minCount: 1; sh:name: Program purpose; sh:path: ekg-communication:programPurpose)
sh:targetClass	ekg-communication:CommunicationProgram

8.5.4 Delivery Event Shape

Validates a delivery event's attempt, delivery status, and occurrence time.

ekg-communication:DeliveryEventShape	
https://ekgf.org/ontology/communication#DeliveryEventShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Delivery Event Shape
sh:property	(sh:class: ekg-communication:CommunicationAttempt; sh:maxCount: 1; sh:minCount: 1; sh:name: Delivery attempt; sh:path: ekg-communication:deliveryAttempt)
sh:property	(sh:class: ekg-communication:DeliveryStatus; sh:maxCount: 1; sh:minCount: 1; sh:name: Delivery status; sh:path: ekg-communication:deliveryStatus)
sh:property	(sh:datatype: xsd:dateTime; sh:maxCount: 1; sh:minCount: 1; sh:name: Occurred at; sh:path: ekg-communication:occurredAt)
sh:targetClass	ekg-communication:DeliveryEvent

8.5.5 Email Recipient Shape

Requires at least one email endpoint while allowing additional endpoints for other communication channels.

ekg-communication:EmailRecipientShape	
https://ekgf.org/ontology/communication#EmailRecipientShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Email Recipient Shape
sh:property	(sh:name: Email endpoint; sh:path: ekg-communication:hasEndpoint; sh:qualifiedMinCount: 1; sh:qualifiedValueShape: (sh:class: ekg-communication:EmailEndpoint))
sh:targetClass	ekg-communication:EmailRecipient

8.5.6 Membership Shape

Requires exactly one collection and at least one recipient or endpoint, with at most one of each target type, and validates the recorded membership status.

ekg-communication:MembershipShape	
https://ekgf.org/ontology/communication#MembershipShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Membership Shape
sh:or	((sh:property: (sh:minCount: 1; sh:path: ekg-communication:membershipRecipient)), (sh:property: (sh:minCount: 1; sh:path: ekg-communication:membershipEndpoint)))
sh:property	(sh:class: ekg-communication:RecipientCollection; sh:maxCount: 1; sh:minCount: 1; sh:name: Membership collection; sh:path: ekg-communication:membershipCollection)

ekg-communication:MembershipShape	
https://ekgf.org/ontology/communication#MembershipShape	
RDF predicate	**Value**
sh:property	(sh:class: ekg-communication:CommunicationEndpoint; sh:maxCount: 1; sh:name: Membership endpoint; sh:path: ekg-communication:membershipEndpoint)
sh:property	(sh:class: ekg-communication:Recipient; sh:maxCount: 1; sh:name: Membership recipient; sh:path: ekg-communication:membershipRecipient)
sh:property	(sh:class: ekg-communication:MembershipStatus; sh:maxCount: 1; sh:name: Membership status; sh:path: ekg-communication:membershipStatus)
sh:targetClass	ekg-communication:Membership

8.5.7 Recipient Collection Shape

Validates a recipient collection's declared channels and purpose.

ekg-communication:RecipientCollectionShape	
https://ekgf.org/ontology/communication#RecipientCollectionShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Recipient Collection Shape
sh:property	(sh:class: ekg-communication:CommunicationChannel; sh:name: Allowed channel; sh:path: ekg-communication:allowedChannel)
sh:property	(sh:class: ekg-communication:CommunicationPurpose; sh:name: Collection purpose; sh:path: ekg-communication:collectionPurpose)
sh:targetClass	ekg-communication:RecipientCollection

8.5.8 Recipient Consent Shape

Validates recorded consent or legal-basis evidence for a membership, including its capture time. Structural validity does not determine legal sufficiency.

ekg-communication:RecipientConsentShape	
https://ekgf.org/ontology/communication#RecipientConsentShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Recipient Consent Shape
sh:property	(sh:datatype: xsd:dateTime; sh:maxCount: 1; sh:minCount: 1; sh:name: Captured at; sh:path: ekg-communication:capturedAt)
sh:property	(sh:class: ekg-communication:Membership; sh:maxCount: 1; sh:minCount: 1; sh:name: Consent membership; sh:path: ekg-communication:consentMembership)
sh:property	(sh:class: ekg-communication:LegalBasis; sh:minCount: 1; sh:name: Legal basis; sh:path: ekg-communication:legalBasis)
sh:targetClass	ekg-communication:RecipientConsent

8.5.9 Recipient Legal Classification Shape

Validates a recipient's recorded jurisdiction and legal category. Structural validity does not establish the correctness of that legal classification.

ekg-communication:RecipientLegalClassificationShape	
https://ekgf.org/ontology/communication#RecipientLegalClassificationShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Recipient Legal Classification Shape
sh:property	(sh:class: ekg-communication:RecipientLegalCategory; sh:maxCount: 1; sh:minCount: 1; sh:name: Legal category; sh:path: ekg-communication:legalCategory)
sh:property	(sh:maxCount: 1; sh:minCount: 1; sh:name: Legal jurisdiction; sh:nodeKind: sh:IRI; sh:path: ekg-communication:legalJurisdiction)
sh:property	(sh:class: ekg-communication:Recipient; sh:maxCount: 1; sh:minCount: 1; sh:name: Recipient; sh:path: ekg-communication:legalClassificationRecipient)
sh:targetClass	ekg-communication:RecipientLegalClassification

8.5.10 Recipient Shape

Validates the resource represented by a communication recipient.

ekg-communication:RecipientShape	
https://ekgf.org/ontology/communication#RecipientShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Recipient Shape
sh:property	(sh:maxCount: 1; sh:name: Represented resource; sh:nodeKind: sh:IRI; sh:path: ekg-communication:representedResource)
sh:targetClass	ekg-communication:Recipient

9 Concept Ontology

9.1 Overview

Concept Ontology

Identifier	Value
Ontology IRI / Namespace	`https://ekgf.org/ontology/concept#`
Prefix	`ekg-concept`

Every organization describes the same things in many ways. A form says "Customer ID", a spreadsheet has a `cust_no` column, an API takes `customerId`, a query uses `?customer`, and an ontology defines a Customer class. The reverse also happens: one word means different things in different departments. Without an explicit record of what each of these means, and where it appears, nobody can tell whether two systems talk about the same thing.

The Concept Ontology gives each meaning its own identity: the **data element concept** (DEC), or **data element** for short. A DEC is a logical data element with all its various forms of physical manifestations and their sightings.

Despite its name, a DEC is not a `skos:Concept`. A `skos:Concept` is an entry in a taxonomy or thesaurus. Such an entry can be one of a data element's manifestations, just like a column or an API parameter.

The Concept Ontology lets an enterprise knowledge graph:

- **Preserve business language.** Capture terms as the business uses them, not as an external standard names them.
- **Connect meaning to every physical form.** Link the same data element to its business terms, API parameters, query variables, columns, ontology classes, SHACL shapes and taxonomy entries.
- **Keep meaning consistent.** Reuse one data element across use cases, stories, languages and systems wherever the meaning is the same.
- **Make the semantics explicit.** Represent different terms for the same data element, and the same term for different data elements, instead of hiding them.
- **Check reality against intent.** Compare where a data element was actually found with where stories declare that they use it.

9.1.1 Why this much detail: data lineage

Maintaining data elements with all their manifestations and all their sightings is a lot of work. Sightings do not record themselves: finding them takes discovery mechanisms, such as harvesters that scan datasets, files, statements and source code.

The reason to do this work is full, detailed data lineage: tracing a piece of data from system to system, for example from a form field through an API parameter and a query variable to the column where it is stored. That trace is only possible with an explicit record of every manifestation and every place where it appears. An organization that needs detailed data lineage needs this level of detail.

9.1.2 Meaning, manifestations and sightings

Layer	Resource	Answers
Meaning	`DataElementConcept`	What does the data element mean?

Layer	Resource	Answers
Manifestation	`ConceptManifestation`	How is it expressed, as a business term or as one of many kinds of technical term?
Sighting	`ManifestationOccurrence`	Where was each manifestation observed, and in which statement position?

A data element for a single attribute, such as a customer identifier, works the same way as a data element for a compound construct, such as a whole customer record with an OWL class and a SHACL shape as its manifestations. Data elements have no subtype hierarchy for classes, properties or shapes; their manifestations carry that detail. Matching names alone never establish that two manifestations express the same meaning.

9.1.3 Start with business language

At the start of a use case, capture what the business calls things and what those things mean in its context. Ontologies are not needed at this stage. The business owns its use cases and should recognize its language throughout their lifecycle. Capturing a data element before its technical manifestations are known is normal.

9.1.4 From capture to ontology mapping

As the use case matures, map its data elements to technical manifestations: ontology classes and properties, shapes, taxonomies, API parameters, columns and code identifiers. A technical manifestation points at the ontology resource or shape through `manifestationResource`, or records its exact token through `manifestationValue`. Harvesters record where each manifestation appears as sightings. As understanding grows, adjust data elements to better reflect the domain, and link them to more appropriate ontology resources.

9.1.5 Reuse across the use case tree

Data elements are grouped into vocabularies, represented by `ConceptSet`. A use case owns its vocabularies through `ekg-use-case:ownsConceptSet`, and its stories select data elements from those vocabularies and the ones of its ancestor use cases. Reuse an existing data element wherever its meaning fits, even when a new context needs a different name or technical form. Create a separate data element only for a different meaning. When owners establish that independently created data elements mean the same thing, they can merge them.

9.1.6 Data elements in stories

A story declares how it uses a data element through a story-owned usage: an input the caller supplies, an output the story returns, or a dependent data element used internally, such as a join or filter data element. The Story Ontology defines those usages and compares them with the sightings in a story's implementation. A data element used in an implementation but not declared by its story is reported for review.

9.1.7 Relationship to ISO/IEC 11179

`DataElementConcept` follows the distinction between meaning and representation in ISO/IEC 11179-31:2023, section 3.25, where a data element concept combines an object class and a property, such as Customer and Identifier. This ontology extends that scope to compound data constructs. That broader scope does not claim formal equivalence to the ISO model.

The EKG/Method explains how data elements, vocabularies and stories fit into the Use Case Tree Method.

9.1.8 Vocabulary class labels

Classes in this ontology have English title-case `rdfs:label` values, such as `Data Element Concept` and `Business Term`. Labels preserve acronyms such as OWL, RDF, SHACL and SPARQL, and language names

such as JavaScript. These labels name the vocabulary classes. Business-term manifestations supply the names used for individual data elements in their business languages and contexts.

9.2 Classes

9.2.1 Abbreviation

A shortened business term used as a business-language manifestation of a data element concept.

ekg-concept:Abbreviation	
https://ekgf.org/ontology/concept#Abbreviation	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Abbreviation
rdfs:subClassOf	ekg-concept:BusinessTerm

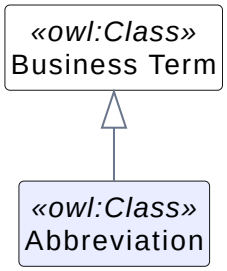


Diagram note

9.2.2 Business Term

A human-facing manifestation such as a preferred label, synonym, or abbreviation.

ekg-concept:BusinessTerm	
https://ekgf.org/ontology/concept#BusinessTerm	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Business Term
rdfs:subClassOf	ekg-concept:ConceptManifestation

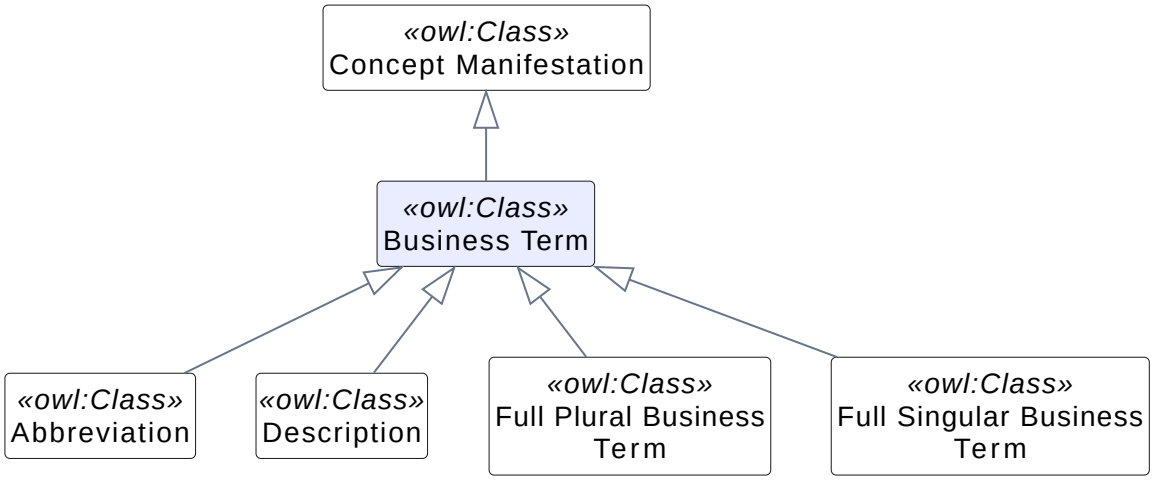


Diagram note

9.2.3 Concept Manifestation

A business or technical expression of a data element concept: a human term, abbreviation, synonym, API parameter, variable name, RDF resource, OWL class, OWL axiom, SHACL shape, SKOS concept, table, column, or source-code identifier. Its appearances in sources are recorded as ManifestationOccurrence instances.

ekg-concept:ConceptManifestation	
https://ekgf.org/ontology/concept#ConceptManifestation	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Concept Manifestation

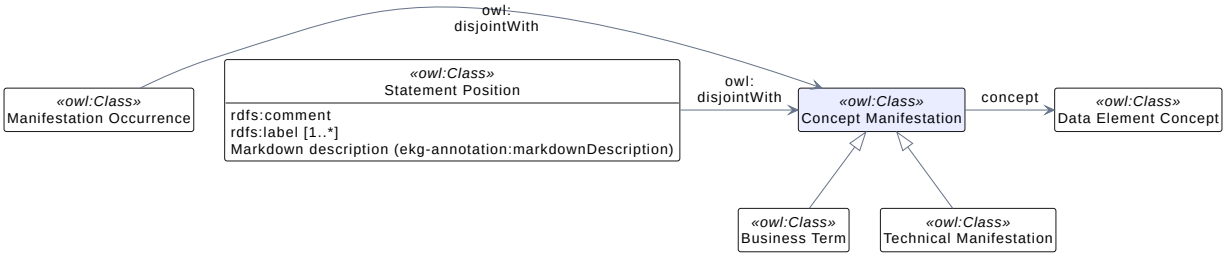


Diagram note

9.2.4 Concept Set

A governed set of related data element concepts.

ekg-concept:ConceptSet	
https://ekgf.org/ontology/concept#ConceptSet	
RDF predicate	**Value**
rdf:type	owl:Class

ekg-concept:ConceptSet	
https://ekgf.org/ontology/concept#ConceptSet	
RDF predicate	**Value**
rdfs:label	Concept Set

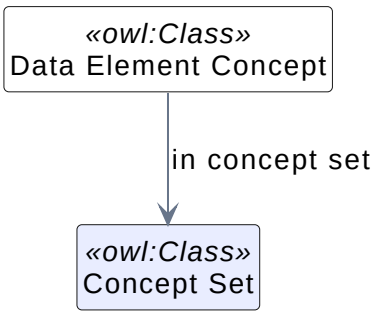


Diagram note

9.2.5 Cypher Identifier Manifestation

A technical manifestation expressing a data element concept through an identifier in Cypher source or queries. The manifestation is distinct from the reusable meaning of the concept and from its occurrences in source files.

ekg-concept:CypherIdentifierManifestation	
https://ekgf.org/ontology/concept#CypherIdentifierManifestation	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Cypher Identifier Manifestation
rdfs:subClassOf	ekg-concept:TechnicalManifestation

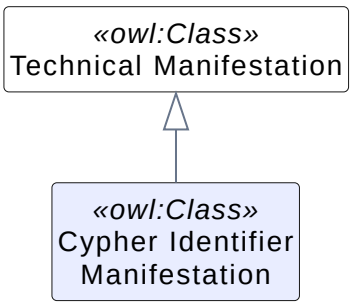


Diagram note

9.2.6 Data Element Concept

A data element concept describes a logical data element in three layers:

Layer	Resource	Answers
Meaning	DataElementConcept	What does the data element mean?

Layer	Resource	Answers
Manifestation	`ConceptManifestation`	How is it expressed, as a business term or as one of many kinds of technical term?
Sighting	`ManifestationOccurrence`	Where was each manifestation observed physically, and in which statement position?

For example, one concept can manifest as the business term "Product category", as a `product_category` column in a dataset and as a placeholder in a story's statement template. Each observed appearance is a separate occurrence: `foundIn` identifies the dataset, file, template or story implementation, and `statementPosition` records how the appearance is used within a statement. These sightings are the evidence of how and where the logical data element manifests physically. The concept keeps its identity when terms are renamed or new sightings are recorded.

Reuse across stories

For example, a customer-identifier concept can be an input to one story and an output from another. Both stories reference the same data element concept. Each story owns a separate usage object describing how it uses that concept. Requiredness and server provision describe an input usage, rather than the reusable meaning.

Vocabulary membership and ownership

`inConceptSet` links a concept to its vocabularies. A use case owns a concept set through `ekg-use-case:ownsConceptSet`. Stories select concepts from their use case's own vocabularies and those inherited from ancestor use cases. Referencing a concept does not give the story ownership of the concept.

Manifestations

A business term, an API name, and a SPARQL variable can express the same concept in different ways. `ConceptManifestation` resources record those expressions, and `ManifestationOccurrence` resources record their appearances in sources. A compound concept may also have a whole OWL class or SHACL shape as a manifestation.

Reuse one manifestation for the same expression of the same concept within the same language across story implementations; record individual appearances as occurrences. For example, twenty statement templates using `$m` as a placeholder for the same meaning can reference one `TemplateParameterManifestation` with label and `manifestationValue "$m"`. Its appearances in the template sources are separate occurrences. The manifestation's `templateLanguage` identifies the syntax in which that token is a placeholder. Native SPARQL variables use `SPARQLVariableManifestation`. SPARQL itself permits dollar-prefixed query variables, so the template language and the expression's role determine the distinction; spelling alone does not establish it or shared meaning.

The template processor must use a documented, explicitly supported language profile. For SPARQL value substitution, validated inputs must be serialized as RDF terms for their intended positions, rather than copied as arbitrary query fragments. The rendered statement must be parsed before execution. The generic manifestation class neither selects an engine nor certifies that an engine or its output is safe.

`occursWithin` links one observed occurrence to another, preserving the context in which an attribute manifestation appears within a compound manifestation. The same attribute concept can occur in different compounds in different sources; containment belongs to the occurrence records, not to the shared concepts or manifestations. A Story's concept usages declare its contract; occurrences record observed appearances. Neither side is inferred from the other. See the occurrence containment guide for an executable example and the limits of declaration/observation comparisons.

`TemplateLanguage` identifies the source language/profile, not its renderer or its generated output language. A template targeting SPARQL is still a document in the declared template language until rendered. The SPARQL Story Implementation Ontology declares template documents, renderers, and their implementation relationships.

Template-placeholder migration

`LegraDollarManifestation` is retired. Replace that type assertion with `TemplateParameterManifestation` and declare the manifestation's documented `templateLanguage` profile. Keep the manifestation's identity, token, concept links, and occurrence references. Do not infer its language from the `$` prefix. `LegacyTemplateManifestationShape` rejects the retired classification even when the current type and a language are also present.

See the Method's concept-vocabulary documentation.

ekg-concept:DataElementConcept	
https://ekgf.org/ontology/concept#DataElementConcept	
RDF predicate	**Value**
dct:created	22 April 2020, 17:56:44 UTC
rdf:type	owl:Class
rdfs:label	Data Element Concept
rdfs:seeAlso	https://www.iso.org/standard/78925.html

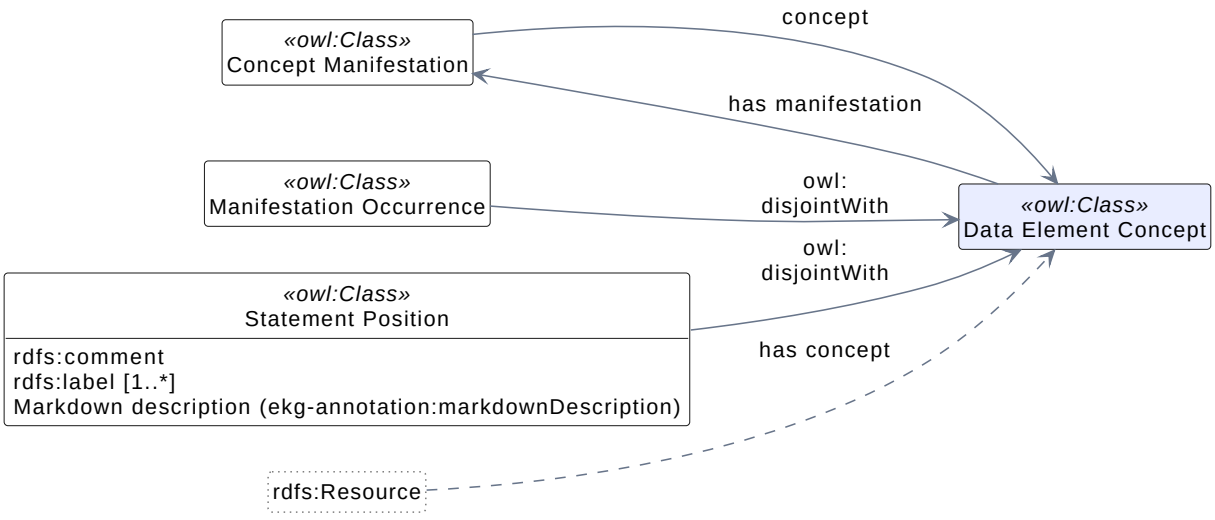


Diagram note

9.2.7 Description

A longer human-facing description of a concept — hover text, tooltips, and detail panes.

ekg-concept:Description	
https://ekgf.org/ontology/concept#Description	
RDF predicate	**Value**
rdf:type	owl:Class

ekg-concept:Description	
https://ekgf.org/ontology/concept#Description	
RDF predicate	**Value**
rdfs:label	Description
rdfs:subClassOf	ekg-concept:BusinessTerm

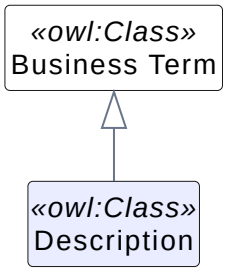


Diagram note

9.2.8 Full Plural Business Term

The plural noun form of a concept's business term — what a column header or collection heading shows.

ekg-concept:FullPluralBusinessTerm	
https://ekgf.org/ontology/concept#FullPluralBusinessTerm	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Full Plural Business Term
rdfs:subClassOf	ekg-concept:BusinessTerm

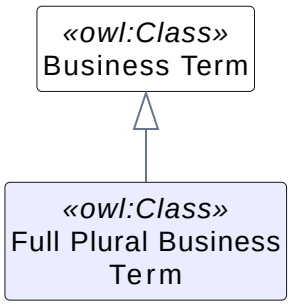


Diagram note

9.2.9 Full Singular Business Term

A business term expressing the full name of a data element concept in singular form.

ekg-concept:FullSingularBusinessTerm	
https://ekgf.org/ontology/concept#FullSingularBusinessTerm	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Full Singular Business Term
rdfs:subClassOf	ekg-concept:BusinessTerm

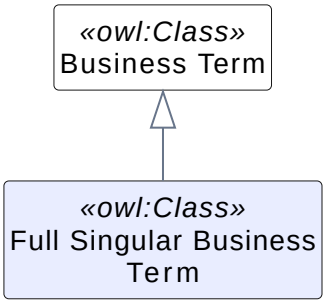


Diagram note

9.2.10 JSON Field Manifestation

A technical manifestation expressing a data element concept through a field in a JSON document or interface. The manifestation is distinct from the reusable meaning of the concept and from its occurrences in source files.

ekg-concept:JSONFieldManifestation	
https://ekgf.org/ontology/concept#JSONFieldManifestation	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	JSON Field Manifestation
rdfs:subClassOf	ekg-concept:TechnicalManifestation

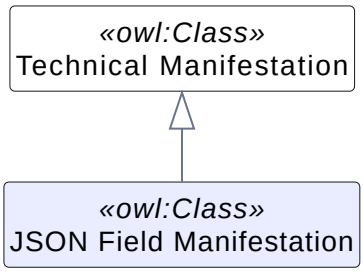


Diagram note

9.2.11 Java Identifier Manifestation

A technical manifestation expressing a data element concept through an identifier in Java source code. The manifestation is distinct from the reusable meaning of the concept and from its occurrences in source files.

ekg-concept:JavaIdentifierManifestation	
https://ekgf.org/ontology/concept#JavaIdentifierManifestation	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Java Identifier Manifestation
rdfs:subClassOf	ekg-concept:SourceCodeIdentifierManifestation

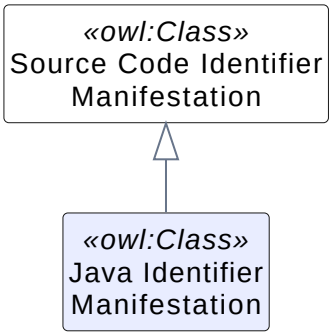


Diagram note

9.2.12 JavaScript Identifier Manifestation

A technical manifestation expressing a data element concept through an identifier in JavaScript source code. The manifestation is distinct from the reusable meaning of the concept and from its occurrences in source files.

ekg-concept:JavaScriptIdentifierManifestation	
https://ekgf.org/ontology/concept#JavaScriptIdentifierManifestation	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	JavaScript Identifier Manifestation
rdfs:subClassOf	ekg-concept:SourceCodeIdentifierManifestation

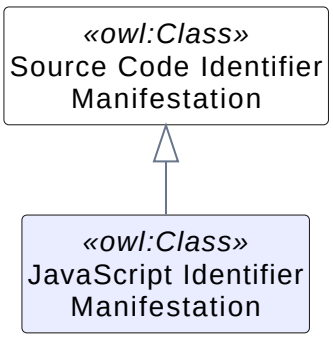


Diagram note

9.2.13 Literal Datatype Manifestation

A technical manifestation expressing a data element concept through a literal datatype. The manifestation is distinct from the reusable meaning of the concept and from its occurrences in source files.

ekg-concept:LiteralDatatypeManifestation	
https://ekgf.org/ontology/concept#LiteralDatatypeManifestation	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Literal Datatype Manifestation
rdfs:subClassOf	ekg-concept:TechnicalManifestation

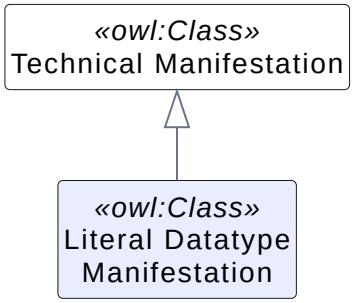


Diagram note

9.2.14 Manifestation Occurrence

An occurrence records where a manifestation was observed. `occurrenceOf` identifies the manifestation. `foundIn` identifies the artifacts containing the sighting, such as a dataset, file, statement template or story implementation. Repository, system, path, commit and line/column metadata locate it further. Separate appearances of the same manifestation have separate occurrence records. An occurrence is disjoint from the reusable DataElementConcept and ConceptManifestation roles.

The manifestation's class records how the concept appears, for example as a raw source field or a template placeholder. Within a query or update statement, `statementPosition` records the role of the appearance: a parameter, a projected result, a graph-template term or a graph-pattern term.

`occursWithin` records observed direct containment between occurrences. Containment is optional, may have several parents, and must be acyclic. Validate a coherent source/version context. A source analyzer is responsible for identifying that context and determining the actual structure. Paths and line/column literals alone do not identify a source artifact or source span; `foundIn` links the sighting to resources representing its artifacts.

Observing a manifestation in a shape specification records the specification's appearance. The requirements expressed by that specification remain distinct from observations of the data it constrains. Similarly, a Story's declared input, output and dependent usages remain distinct from observations in an implementation source. Observations never create those declarations. Statement positions identify the usage role a Story analysis expects; the Story Ontology compares that evidence with the declared usages and reports gaps as warnings. See the sightings guide.

ekg-concept:ManifestationOccurrence	
https://ekgf.org/ontology/concept#ManifestationOccurrence	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Manifestation Occurrence
owl:disjointWith	ekg-concept:ConceptManifestation
owl:disjointWith	ekg-concept:DataElementConcept

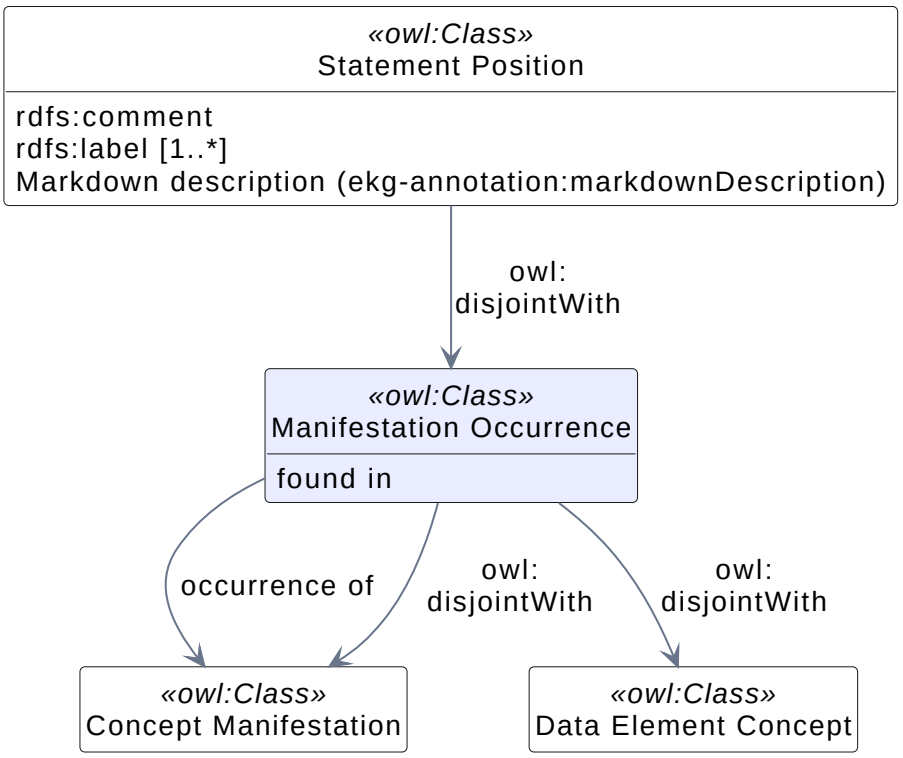


Diagram note

9.2.15 OWL Axiom Manifestation

A technical manifestation expressing a data element concept through an OWL axiom. The manifestation is distinct from the reusable meaning of the concept and from its occurrences in source files.

ekg-concept:OWLAxiomManifestation	
https://ekgf.org/ontology/concept#OWLAxiomManifestation	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	OWL Axiom Manifestation
rdfs:subClassOf	ekg-concept:TechnicalManifestation

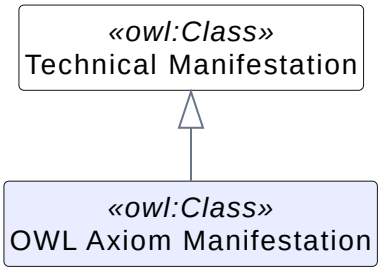


Diagram note

9.2.16 OWL Class Manifestation

A technical manifestation expressing a data element concept through an OWL class. The manifestation is distinct from the reusable meaning of the concept and from its occurrences in source files.

ekg-concept:OWLClassManifestation	
https://ekgf.org/ontology/concept#OWLClassManifestation	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	OWL Class Manifestation
rdfs:subClassOf	ekg-concept:TechnicalManifestation

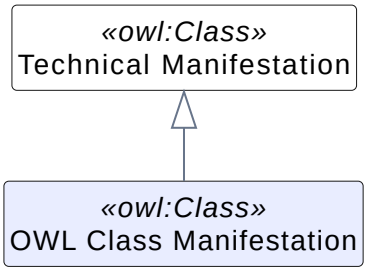


Diagram note

9.2.17 OWL Named Individual Manifestation

A technical manifestation expressing a data element concept through an OWL named individual. The manifestation is distinct from the reusable meaning of the concept and from its occurrences in source files.

ekg-concept:OWLNamedIndividualManifestation	
https://ekgf.org/ontology/concept#OWLNamedIndividualManifestation	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	OWL Named Individual Manifestation
rdfs:subClassOf	ekg-concept:TechnicalManifestation

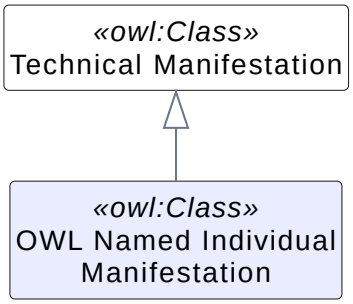


Diagram note

9.2.18 OWL Property Manifestation

A technical manifestation expressing a data element concept through an OWL property. The manifestation is distinct from the reusable meaning of the concept and from its occurrences in source files.

ekg-concept:OWLPropertyManifestation	
https://ekgf.org/ontology/concept#OWLPropertyManifestation	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	OWL Property Manifestation
rdfs:subClassOf	ekg-concept:TechnicalManifestation

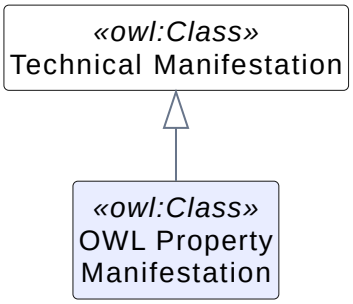


Diagram note

9.2.19 Python Identifier Manifestation

A technical manifestation expressing a data element concept through an identifier in Python source code. The manifestation is distinct from the reusable meaning of the concept and from its occurrences in source files.

ekg-concept:PythonIdentifierManifestation	
https://ekgf.org/ontology/concept#PythonIdentifierManifestation	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Python Identifier Manifestation

ekg-concept:PythonIdentifierManifestation	
https://ekgf.org/ontology/concept#PythonIdentifierManifestation	
RDF predicate	**Value**
rdfs:subClassOf	ekg-concept:SourceCodeIdentifierManifestation

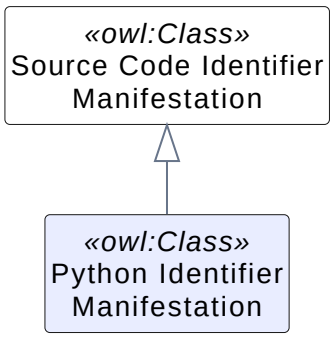


Diagram note

9.2.20 RDF Predicate Manifestation

A technical manifestation expressing a data element concept through a predicate used in RDF statements. The manifestation is distinct from the reusable meaning of the concept and from its occurrences in source files.

ekg-concept:RDFPredicateManifestation	
https://ekgf.org/ontology/concept#RDFPredicateManifestation	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	RDF Predicate Manifestation
rdfs:subClassOf	ekg-concept:TechnicalManifestation

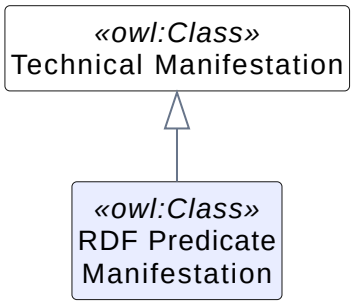


Diagram note

9.2.21 RDFS Class Manifestation

A technical manifestation expressing a data element concept through an RDFS class. The manifestation is distinct from the reusable meaning of the concept and from its occurrences in source files.

ekg-concept:RDFSClassManifestation	
https://ekgf.org/ontology/concept#RDFSClassManifestation	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	RDFS Class Manifestation
rdfs:subClassOf	ekg-concept:TechnicalManifestation

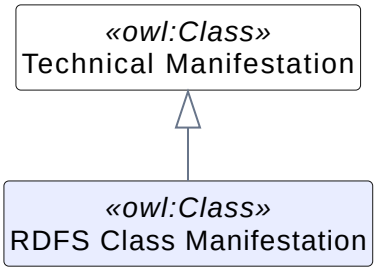


Diagram note

9.2.22 Raw Source Field Manifestation

A technical manifestation expressing a data element concept through a field in captured raw source data. The manifestation is distinct from the reusable meaning of the concept and from its occurrences in source files.

ekg-concept:RawSourceFieldManifestation	
https://ekgf.org/ontology/concept#RawSourceFieldManifestation	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Raw Source Field Manifestation
rdfs:subClassOf	ekg-concept:TechnicalManifestation

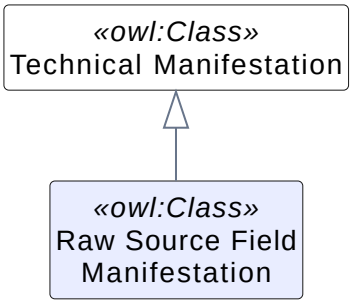


Diagram note

9.2.23 SHACL Shape Manifestation

A technical manifestation expressing a data element concept through a SHACL shape. The manifestation is distinct from the reusable meaning of the concept and from its occurrences in source files.

ekg-concept:SHACLShapeManifestation	
https://ekgf.org/ontology/concept#SHACLShapeManifestation	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	SHACL Shape Manifestation
rdfs:subClassOf	ekg-concept:TechnicalManifestation

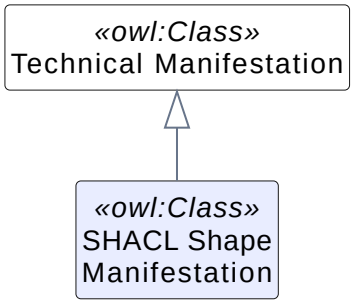


Diagram note

9.2.24 SHACL Shape Set Manifestation

A technical manifestation expressing a data element concept through a collection of SHACL shapes. The manifestation is distinct from the reusable meaning of the concept and from its occurrences in source files.

ekg-concept:SHACLShapeSetManifestation	
https://ekgf.org/ontology/concept#SHACLShapeSetManifestation	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	SHACL Shape Set Manifestation
rdfs:subClassOf	ekg-concept:TechnicalManifestation

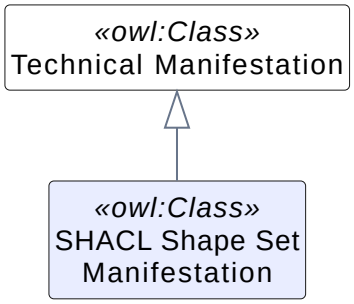


Diagram note

9.2.25 SKOS Concept Manifestation

A technical manifestation expressing a data element concept through a SKOS concept. The manifestation is distinct from the reusable meaning of the concept and from its occurrences in source files.

ekg-concept:SKOSConceptManifestation	
https://ekgf.org/ontology/concept#SKOSConceptManifestation	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	SKOS Concept Manifestation
rdfs:subClassOf	ekg-concept:TechnicalManifestation

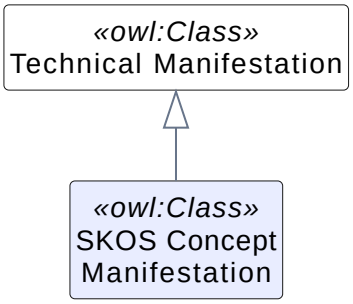


Diagram note

9.2.26 SKOS Concept Scheme Manifestation

A technical manifestation expressing a data element concept through a SKOS concept scheme. The manifestation is distinct from the reusable meaning of the concept and from its occurrences in source files.

ekg-concept:SKOSConceptSchemeManifestation	
https://ekgf.org/ontology/concept#SKOSConceptSchemeManifestation	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	SKOS Concept Scheme Manifestation
rdfs:subClassOf	ekg-concept:TechnicalManifestation

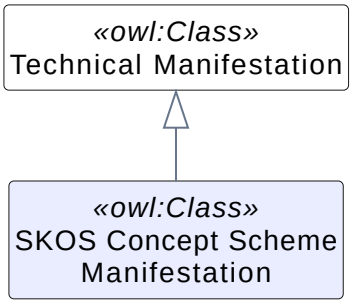


Diagram note

9.2.27 SPARQL Result Column Manifestation

A technical manifestation expressing a data element concept through a column in a SPARQL query result. The manifestation is distinct from the reusable meaning of the concept and from its occurrences in source files.

ekg-concept:SPARQLResultColumnManifestation	
https://ekgf.org/ontology/concept#SPARQLResultColumnManifestation	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	SPARQL Result Column Manifestation
rdfs:subClassOf	ekg-concept:TechnicalManifestation

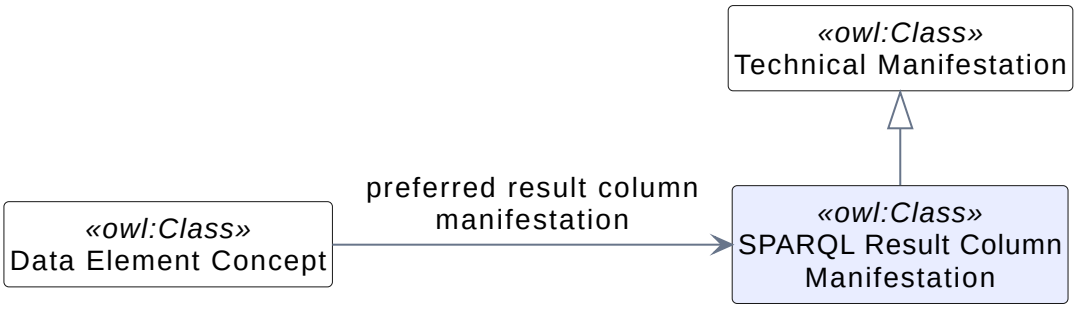


Diagram note

9.2.28 SPARQL Variable Manifestation

A technical manifestation expressing a data element concept through a variable in a SPARQL statement. The manifestation is distinct from the reusable meaning of the concept and from its occurrences in source files.

ekg-concept:SPARQLVariableManifestation	
https://ekgf.org/ontology/concept#SPARQLVariableManifestation	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	SPARQL Variable Manifestation
rdfs:subClassOf	ekg-concept:TechnicalManifestation

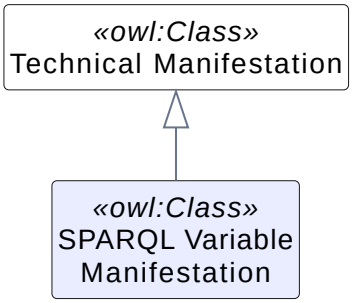


Diagram note

9.2.29 SQL Column Manifestation

A technical manifestation expressing a data element concept through a column in a relational database schema. The manifestation is distinct from the reusable meaning of the concept and from its occurrences in source files.

ekg-concept:SQLColumnManifestation	
https://ekgf.org/ontology/concept#SQLColumnManifestation	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	SQL Column Manifestation
rdfs:subClassOf	ekg-concept:TechnicalManifestation

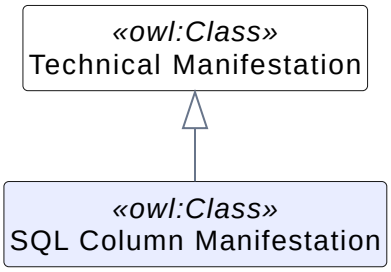


Diagram note

9.2.30 SQL Table Manifestation

A technical manifestation expressing a data element concept through a table in a relational database schema. The manifestation is distinct from the reusable meaning of the concept and from its occurrences in source files.

ekg-concept:SQLTableManifestation	
https://ekgf.org/ontology/concept#SQLTableManifestation	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	SQL Table Manifestation
rdfs:subClassOf	ekg-concept:TechnicalManifestation

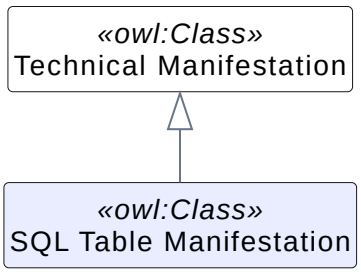


Diagram note

9.2.31 Source Code Identifier Manifestation

An identifier in a programming language, such as a variable, field, function, or class name. Use this class directly for languages without a dedicated subclass. manifestationValue records the exact identifier; ManifestationOccurrence records where it appears.

ekg-concept:SourceCodeIdentifierManifestation	
https://ekgf.org/ontology/concept#SourceCodeIdentifierManifestation	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Source Code Identifier Manifestation
rdfs:subClassOf	ekg-concept:TechnicalManifestation

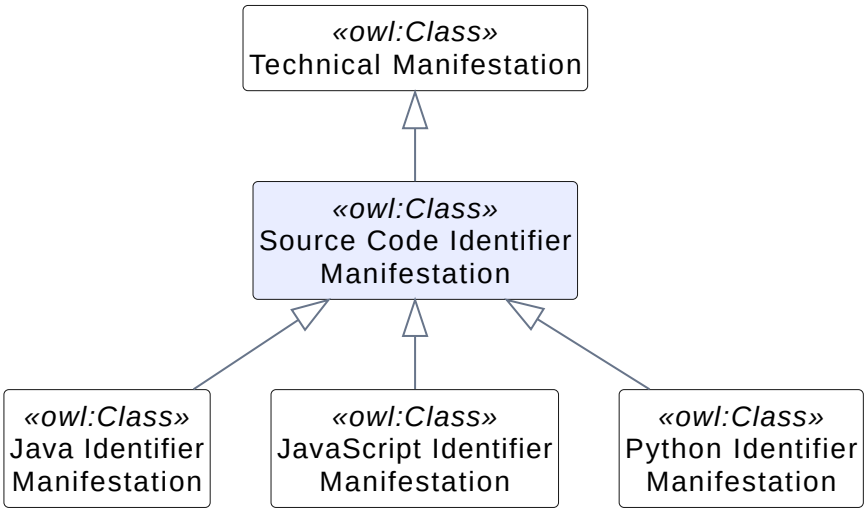


Diagram note

9.2.32 Statement Position

The four built-in positions are distinct named individuals. Applications can define more positions as IRIs with a label and a definition. A position compared with Story declarations must also identify the usage role it implies through ekg-story:expectedUsageRole.

The built-ins classify SPARQL query and update positions. They do not yet classify terms in DELETE templates or DELETE DATA; that usage role remains an open modeling decision, so analyzers must not record those terms as GraphTemplatePosition.

ekg-concept:StatementPosition	
https://ekgf.org/ontology/concept#StatementPosition	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Statement Position
owl:disjointWith	ekg-concept:ConceptManifestation

ekg-concept:StatementPosition	
https://ekgf.org/ontology/concept#StatementPosition	
RDF predicate	**Value**
owl:disjointWith	ekg-concept:DataElementConcept
owl:disjointWith	ekg-concept:ManifestationOccurrence

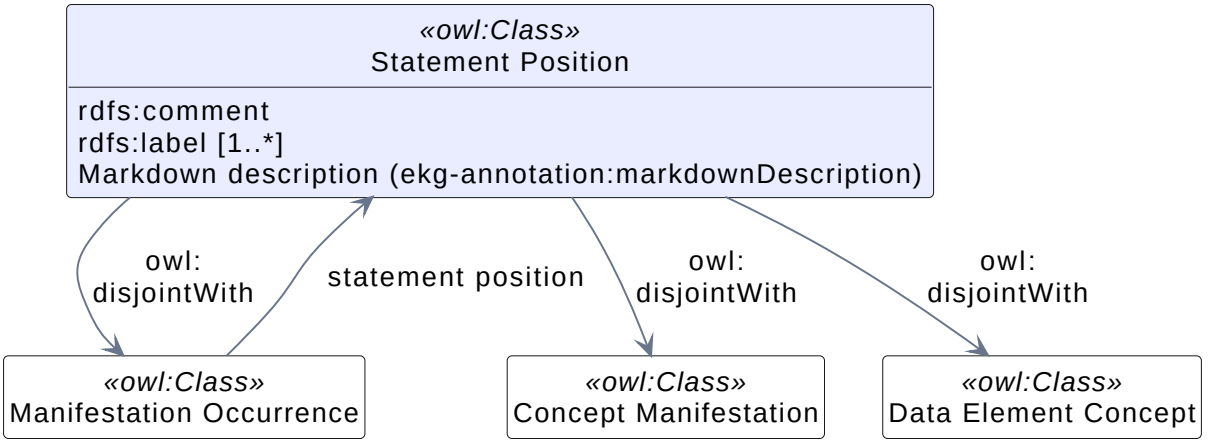


Diagram note

9.2.33 Tagline

A short subtitle or tagline shown beneath a concept's title.

ekg-concept:Tagline	
https://ekgf.org/ontology/concept#Tagline	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Tagline
rdfs:subClassOf	ekg-concept:BusinessTerm

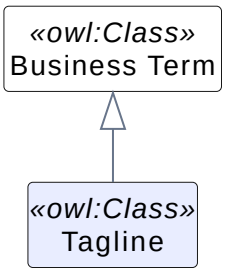


Diagram note

9.2.34 Technical Manifestation

A machine-facing manifestation such as an API token, query variable, code identifier, RDF resource, schema object, table, or column.

ekg-concept:TechnicalManifestation	
https://ekgf.org/ontology/concept#TechnicalManifestation	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Technical Manifestation
rdfs:subClassOf	ekg-concept:ConceptManifestation

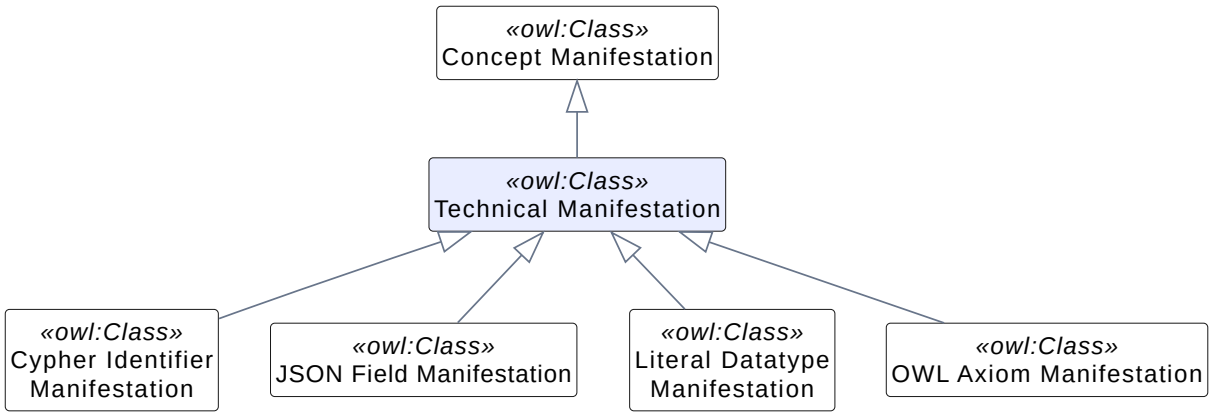


Diagram note

9.2.35 Template Language

A documented template language or substitution profile defining how a template document is interpreted. Its identity is distinct from a renderer implementing it and from the language of the rendered output. Declaring a profile does not approve it for execution.

ekg-concept:TemplateLanguage	
https://ekgf.org/ontology/concept#TemplateLanguage	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Template Language

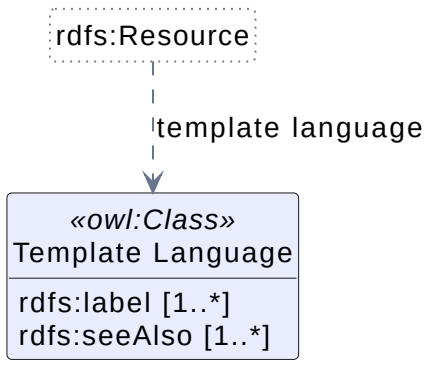


Diagram note

9.2.36 Template Parameter Manifestation

A concept's appearance as a placeholder in a statement template. A template processor substitutes validated input values before the resulting statement is parsed and executed. The template language determines the token syntax; its spelling alone does not identify a language or a native query variable.

ekg-concept:TemplateParameterManifestation	
https://ekgf.org/ontology/concept#TemplateParameterManifestation	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Template Parameter Manifestation
rdfs:subClassOf	ekg-concept:TechnicalManifestation
rdfs:subClassOf	(rdf:type: owl:Restriction; owl:minCardinality: 1; owl:onProperty: ekg-concept:templateLanguage)
rdfs:subClassOf	(rdf:type: owl:Restriction; owl:maxCardinality: 1; owl:onProperty: ekg-concept:templateLanguage)

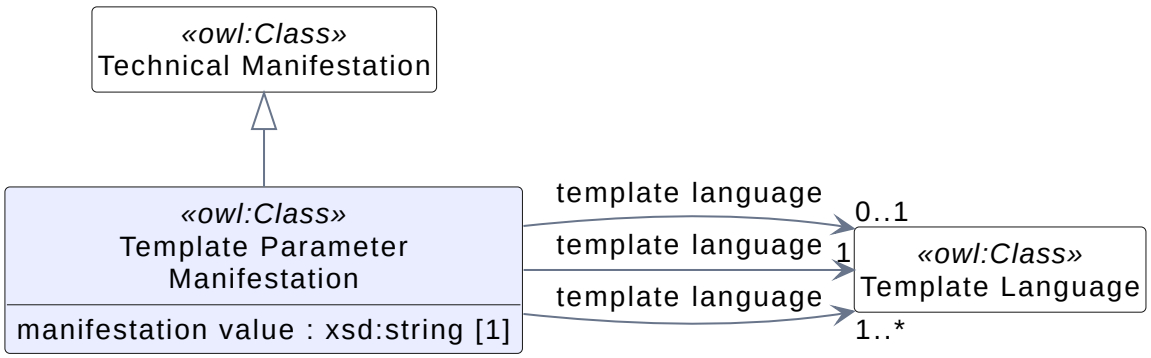


Diagram note

9.2.37 URI Parameter Manifestation

A technical manifestation expressing a data element concept through a URI parameter in an interface. The manifestation is distinct from the reusable meaning of the concept and from its occurrences in source files.

ekg-concept:URIParameterManifestation	
https://ekgf.org/ontology/concept#URIParameterManifestation	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	URI Parameter Manifestation
rdfs:subClassOf	ekg-concept:TechnicalManifestation

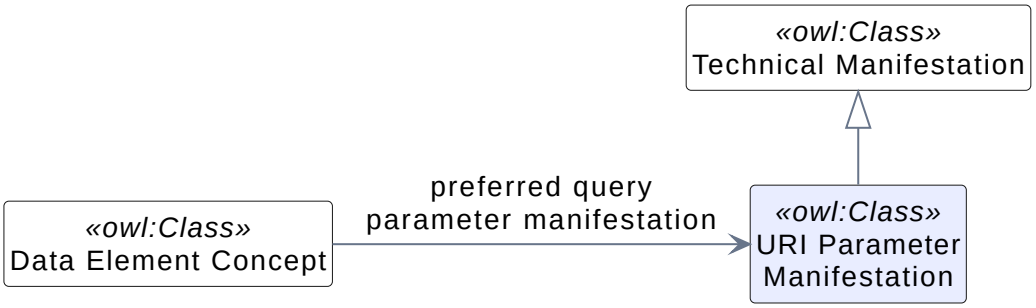


Diagram note

9.3 Properties

9.3.1 concept

Identifies the data element concept expressed by a concept manifestation; it is the inverse of hasManifestation.

ekg-concept:concept	
https://ekgf.org/ontology/concept#concept	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-concept:ConceptManifestation
rdfs:label	concept
rdfs:range	ekg-concept:DataElementConcept
owl:inverseOf	ekg-concept:hasManifestation

9.3.2 found in

Links an occurrence to each artifact in which it was sighted: a dataset, a dataset file, a statement template, a story implementation, a raw-data view, a code file or another identified source artifact. Assert the levels the harvester has resolved. One sighting in a statement template can be found in the template, in the story implementation using it, and in the dataset containing the template's file. Querying `foundIn` for a dataset then finds both column sightings and statement sightings, while the manifestation class and `statementPosition` distinguish how the concept appears in each.

Values are IRIs or blank nodes representing artifacts. They are not occurrences, concepts, manifestations or statement positions; containment between occurrences uses `occursWithin`. The property is not functional: an occurrence can be found in several containing artifacts at different granularities. Paths, lines and columns remain literals locating the sighting within an artifact. A harvester must resolve artifact identity explicitly; matching paths do not establish that two sightings share an artifact.

ekg-concept:foundIn	
https://ekgf.org/ontology/concept#foundIn	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty

ekg-concept:foundIn	
https://ekgf.org/ontology/concept#foundIn	
RDF predicate	**Value**
rdfs:domain	ekg-concept:ManifestationOccurrence
rdfs:label	found in
rdfs:range	rdfs:Resource

9.3.3 has concept

Associates a resource with a data element concept. It does not impose a domain on the resource using the concept.

ekg-concept:hasConcept	
https://ekgf.org/ontology/concept#hasConcept	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:label	has concept
rdfs:range	ekg-concept:DataElementConcept

9.3.4 has manifestation

Links a concept to a concrete manifestation such as a business term, API parameter, SPARQL variable, OWL class, SHACL shape, SKOS concept, source-code identifier, table, or column.

ekg-concept:hasManifestation	
https://ekgf.org/ontology/concept#hasManifestation	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-concept:DataElementConcept
rdfs:label	has manifestation
rdfs:range	ekg-concept:ConceptManifestation

9.3.5 in concept set

Associates a data element concept with a concept set containing it.

ekg-concept:inConceptSet	
https://ekgf.org/ontology/concept#inConceptSet	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-concept:DataElementConcept
rdfs:label	in concept set

ekg-concept:inConceptSet	
https://ekgf.org/ontology/concept#inConceptSet	
RDF predicate	**Value**
rdfs:range	ekg-concept:ConceptSet

9.3.6 manifestation resource

The RDF resource manifested by a technical manifestation, for example an OWL class, OWL property, SHACL shape, SKOS concept, datatype, named graph, table IRI, or external schema identifier.

ekg-concept:manifestationResource	
https://ekgf.org/ontology/concept#manifestationResource	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-concept:TechnicalManifestation
rdfs:label	manifestation resource
rdfs:range	rdfs:Resource

9.3.7 manifestation value

The exact lexical value for a manifestation when it is not naturally represented as an RDF resource. Examples: customerId, ?customer, _abc22, a SQL column token, or a Cypher identifier.

ekg-concept:manifestationValue	
https://ekgf.org/ontology/concept#manifestationValue	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-concept:ConceptManifestation
rdfs:label	manifestation value
rdfs:range	xsd:string

9.3.8 occurrence of

Identifies the concept manifestation that occurs at a particular source location.

ekg-concept:occurrenceOf	
https://ekgf.org/ontology/concept#occurrenceOf	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-concept:ManifestationOccurrence
rdfs:label	occurrence of
rdfs:range	ekg-concept:ConceptManifestation

9.3.9 occurs within

Links a ManifestationOccurrence to an occurrence directly containing it in the same observed source context. For example, a column occurrence can be contained in a worksheet occurrence. Following each endpoint's `occurrenceOf` and concept links then relates the observed attribute meaning to the observed compound meaning in that context.

Both endpoints are occurrences, not DataElementConcepts or ConceptManifestations. One occurrence can have several observed containers, and a container can have several children. This property is neither functional nor inverse-functional; shared endpoints do not merge identities. It is irreflexive and not transitive. Query `occursWithin+` to walk to all ancestors; the ontology does not materialize indirect ancestors as direct containment assertions.

An analyzer must establish and assert containment from the source structure. Matching paths, positions, types or manifestation identities do not create a link. A referenced definition and a reference to it are different source occurrences. Source-location and provenance properties remain on the observed occurrences; containment does not copy or inherit those values.

OccurrenceContainmentShape checks resource endpoints, occurrence roles and cycles. It does not read source content or prove that the asserted endpoints share the same source context. See the containment guide for validation and queries.

ekg-concept:occursWithin	
https://ekgf.org/ontology/concept#occursWithin	
RDF predicate	**Value**
rdf:type	owl:IrreflexiveProperty
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-concept:ManifestationOccurrence
rdfs:label	occurs within
rdfs:range	ekg-concept:ManifestationOccurrence

9.3.10 preferred business term

Identifies a preferred business-language term for a data element concept.

ekg-concept:preferredBusinessTerm	
https://ekgf.org/ontology/concept#preferredBusinessTerm	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-concept:DataElementConcept
rdfs:label	preferred business term
rdfs:range	ekg-concept:BusinessTerm
rdfs:subPropertyOf	ekg-concept:preferredManifestation

9.3.11 preferred manifestation

Identifies a preferred manifestation of a data element concept, specializing hasManifestation.

ekg-concept:preferredManifestation	
https://ekgf.org/ontology/concept#preferredManifestation	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-concept:DataElementConcept
rdfs:label	preferred manifestation
rdfs:range	ekg-concept:ConceptManifestation
rdfs:subPropertyOf	ekg-concept:hasManifestation

9.3.12 preferred query parameter manifestation

Identifies a preferred URI-parameter manifestation for a data element concept.

ekg-concept:preferredQueryParameterManifestation	
https://ekgf.org/ontology/concept#preferredQueryParameterManifestation	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-concept:DataElementConcept
rdfs:label	preferred query parameter manifestation
rdfs:range	ekg-concept:URIParameterManifestation
rdfs:subPropertyOf	ekg-concept:preferredManifestation

9.3.13 preferred result column manifestation

The manifestation whose label is the concept's public response-column name — what the edge projector renders in place of the SPARQL projection variable (story-signature design).

ekg-concept:preferredResultColumnManifestation	
https://ekgf.org/ontology/concept#preferredResultColumnManifestation	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-concept:DataElementConcept
rdfs:label	preferred result column manifestation
rdfs:range	ekg-concept:SPARQLResultColumnManifestation
rdfs:subPropertyOf	ekg-concept:preferredManifestation

9.3.14 source column

Records the column position of a manifestation occurrence in its source. The ontology does not specify whether the position is zero-based or one-based.

ekg-concept:sourceColumn	
https://ekgf.org/ontology/concept#sourceColumn	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-concept:ManifestationOccurrence
rdfs:label	source column
rdfs:range	xsd:integer

9.3.15 source commit

Records the source repository commit associated with a manifestation occurrence.

ekg-concept:sourceCommit	
https://ekgf.org/ontology/concept#sourceCommit	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-concept:ManifestationOccurrence
rdfs:label	source commit
rdfs:range	xsd:string

9.3.16 source line

Records the line position of a manifestation occurrence in its source. The ontology does not specify whether the position is zero-based or one-based.

ekg-concept:sourceLine	
https://ekgf.org/ontology/concept#sourceLine	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-concept:ManifestationOccurrence
rdfs:label	source line
rdfs:range	xsd:integer

9.3.17 source path

Records the source path at which a manifestation occurrence is located.

ekg-concept:sourcePath	
https://ekgf.org/ontology/concept#sourcePath	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-concept:ManifestationOccurrence

ekg-concept:sourcePath	
https://ekgf.org/ontology/concept#sourcePath	
RDF predicate	**Value**
rdfs:label	source path
rdfs:range	xsd:string

9.3.18 source repository

Records the repository IRI associated with a manifestation occurrence as an xsd:anyURI literal.

ekg-concept:sourceRepository	
https://ekgf.org/ontology/concept#sourceRepository	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-concept:ManifestationOccurrence
rdfs:label	source repository
rdfs:range	xsd:anyURI

9.3.19 source system

The system in which this occurrence of a manifestation was observed, identified by a resource IRI. A source repository is not required; the same manifestation may occur in multiple systems.

ekg-concept:sourceSystem	
https://ekgf.org/ontology/concept#sourceSystem	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-concept:ManifestationOccurrence
rdfs:label	source system
rdfs:range	rdfs:Resource

9.3.20 statement position

Classifies a position of an occurrence within the query or update statement, or the statement template rendering one, identified through `foundIn`:

Position	Appearance	Example
ParameterPosition	A value supplied for the invocation.	A template placeholder, or a variable the service pre-binds.
ProjectionPosition	A result the statement returns.	A projected SELECT variable or alias.
GraphTemplatePosition	A term the statement constructs or inserts.	A predicate in a CONSTRUCT or INSERT template.
GraphPatternPosition	A term matched against data.	A class or predicate in the WHERE clause.

One appearance can have several positions. A placeholder used in a triple pattern is a parameter appearing in the graph pattern; record both. Record the parameter position from the template language or the service's documented binding profile, never from a token's spelling: SPARQL treats `$name` and `?name` as one variable.

A positioned occurrence must identify at least one artifact through `foundIn`. Positions are evidence about a statement. They do not declare a story's concept usages; the Story Ontology maps each position to the usage role it leads an analysis to expect.

ekg-concept:statementPosition	
https://ekgf.org/ontology/concept#statementPosition	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-concept:ManifestationOccurrence
rdfs:label	statement position
rdfs:range	ekg-concept:StatementPosition

9.3.21 taxonomy kind

Use a string such as `thesaurus` to describe the kind of a SKOS concept scheme. This metadata concerns the scheme's organization; it does not identify a data element concept or classify an individual SKOS concept as a data element.

The property retains its string range and has no domain inference or closed list of permitted kinds. Moving it into the Concept Ontology does not introduce a new classification model or change the types of resources carrying it.

ekg-concept:taxonomyKind	
https://ekgf.org/ontology/concept#taxonomyKind	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:label	taxonomy kind
rdfs:range	xsd:string

9.3.22 template language

Identifies the documented template language or substitution profile of a template document or placeholder manifestation. This general metadata property imposes no domain; each resource kind constrains its own language cardinality.

ekg-concept:templateLanguage	
https://ekgf.org/ontology/concept#templateLanguage	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:label	template language
rdfs:range	ekg-concept:TemplateLanguage

9.4 Named Individuals

9.4.1 Graph Pattern

Instance of: ekg-concept:StatementPosition

An appearance in the graph pattern that the statement matches against data: its WHERE clause, including filters, bindings, inline data, optional, union, negation and subquery patterns, and the solution modifiers applied to those solutions.

ekg-concept:GraphPatternPosition	
https://ekgf.org/ontology/concept#GraphPatternPosition	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-concept:StatementPosition
rdfs:label	Graph Pattern

9.4.2 Graph Template

Instance of: ekg-concept:StatementPosition

An appearance in a graph template whose instantiations the statement returns or adds: a CONSTRUCT template, an INSERT template, or INSERT DATA. This is the statement's own SPARQL graph template, not a template document in a template language. Terms in DELETE templates and DELETE DATA are not classified by this position.

ekg-concept:GraphTemplatePosition	
https://ekgf.org/ontology/concept#GraphTemplatePosition	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-concept:StatementPosition
rdfs:label	Graph Template
owl:differentFrom	ekg-concept:GraphPatternPosition

9.4.3 Parameter

Instance of: ekg-concept:StatementPosition

An appearance whose value is supplied for an invocation rather than matched or produced by the statement: a template placeholder substituted before the statement is parsed, or a statement variable that the executing service pre-binds under its documented binding profile. Spelling does not establish this position; SPARQL treats $name and ?name as the same variable.

ekg-concept:ParameterPosition	
https://ekgf.org/ontology/concept#ParameterPosition	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual

ekg-concept:ParameterPosition	
https://ekgf.org/ontology/concept#ParameterPosition	
RDF predicate	**Value**
rdf:type	ekg-concept:StatementPosition
rdfs:label	Parameter
owl:differentFrom	ekg-concept:GraphPatternPosition
owl:differentFrom	ekg-concept:GraphTemplatePosition
owl:differentFrom	ekg-concept:ProjectionPosition

9.4.4 Projection

Instance of: ekg-concept:StatementPosition

An appearance as a result the statement returns: a variable or expression alias projected by its SELECT clause, or a DESCRIBE target. A projection inside a subquery belongs to the enclosing graph pattern, not to the statement's results.

ekg-concept:ProjectionPosition	
https://ekgf.org/ontology/concept#ProjectionPosition	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-concept:StatementPosition
rdfs:label	Projection
owl:differentFrom	ekg-concept:GraphPatternPosition
owl:differentFrom	ekg-concept:GraphTemplatePosition

9.5 Shapes

9.5.1 Legacy Template Manifestation Shape

Rejects instances of the retired product-specific template-placeholder class, including instances also given the current type or a language. Migration must remove the retired type and identify the placeholder's language profile.

ekg-concept:LegacyTemplateManifestationShape	
https://ekgf.org/ontology/concept#LegacyTemplateManifestationShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Legacy Template Manifestation Shape
sh:message	LegraDollarManifestation is retired. Replace that type with TemplateParameter Manifestation and declare its documented templateLanguage profile.
sh:not	(sh:class: ekg-concept:LegraDollarManifestation)
sh:targetClass	ekg-concept:LegraDollarManifestation

9.5.2 Occurrence Containment Shape

Targets subjects and objects of `occursWithin`, including untyped endpoints. Endpoints must be IRI or blank-node ManifestationOccurrences and must not also be DataElementConcepts or ConceptManifestations. RDFS inference with the Concept Ontology supplies endpoint types from the property's domain and range. Without inference, supply those occurrence types explicitly.

A SPARQL property-path constraint rejects self-links and longer containment cycles without adding transitive property semantics. Roots without a container, multiple children and multiple direct parents remain valid. This validator and its nested constraints belong to the Concept Ontology's shapes graph.

The shape validates supplied graph structure. It does not inspect source files, infer containment from coordinates, enforce source-provenance equality or check harvest completeness. An analyzer must supply observed same-context edges.

ekg-concept:OccurrenceContainmentShape	
https://ekgf.org/ontology/concept#OccurrenceContainmentShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Occurrence Containment Shape
sh:class	ekg-concept:ManifestationOccurrence
sh:nodeKind	sh:BlankNodeOrIRI
sh:not	(sh:or: ((sh:class: ekg-concept:DataElementConcept), (sh:class: ekg-concept:ConceptManifestation)))
sh:property	(sh:class: ekg-concept:ManifestationOccurrence; sh:nodeKind: sh:BlankNodeOrIRI; sh:path: ekg-concept:occursWithin)
sh:sparql	(sh:message: Occurrence containment must be acyclic; an occurrence cannot contain itself directly or through other occurrences.; sh:select: SPARQL SELECT query 1)
sh:targetObjectsOf	ekg-concept:occursWithin
sh:targetSubjectsOf	ekg-concept:occursWithin

9.5.2.1 SPARQL query 1

Occurrence containment must be acyclic; an occurrence cannot contain itself directly or through other occurrences.

```
PREFIX concept: <https://ekgf.org/ontology/concept#>
SELECT $this WHERE {
  $this concept:occursWithin+ $this .
}
```

9.5.3 Occurrence Sighting Shape

Targets subjects of `foundIn` and `statementPosition`, which must be ManifestationOccurrences and not concepts or manifestations. Artifact values must be resources other than occurrences, concepts, manifestations and positions. Position values must be StatementPosition IRIs. An occurrence with a statement position must identify at least one artifact. RDFS inference supplies occurrence and position types from the property domains and ranges; without inference, supply those types explicitly.

The shape checks the supplied graph. It does not read source files or prove that the asserted artifacts and positions match the source.

ekg-concept:OccurrenceSightingShape	
https://ekgf.org/ontology/concept#OccurrenceSightingShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Occurrence Sighting Shape
sh:class	ekg-concept:ManifestationOccurrence
sh:nodeKind	sh:BlankNodeOrIRI
sh:not	(sh:or: ((sh:class: ekg-concept:DataElementConcept), (sh:class: ekg-concept:ConceptManifestation)))
sh:property	(sh:message: foundIn must identify an artifact resource, not a literal, occurrence, concept, manifestation or statement position. Containment between occurrences uses occursWithin.; sh:nodeKind: sh:BlankNodeOrIRI; sh:not: (sh:or: ((sh:class: ekg-concept:ManifestationOccurrence), (sh:class: ekg-concept:DataElementConcept), (sh:class: ekg-concept:ConceptManifestation), (sh:class: ekg-concept:StatementPosition))); sh:path: ekg-concept:foundIn)
sh:property	(sh:class: ekg-concept:StatementPosition; sh:message: statementPosition must identify a StatementPosition IRI.; sh:nodeKind: sh:IRI; sh:path: ekg-concept:statementPosition)
sh:sparql	(sh:message: An occurrence with a statement position must identify, through foundIn, an artifact containing that statement.; sh:select: SPARQL SELECT query 1)
sh:targetSubjectsOf	ekg-concept:foundIn
sh:targetSubjectsOf	ekg-concept:statementPosition

9.5.3.1 SPARQL query 1

An occurrence with a statement position must identify, through foundIn, an artifact containing that statement.

```
PREFIX concept: <https://ekgf.org/ontology/concept#>
SELECT $this WHERE {
  $this concept:statementPosition ?position .
  FILTER NOT EXISTS { $this concept:foundIn ?artifact }
}
```

9.5.4 Statement Position Shape

Targets declared positions and every object of `statementPosition`. A position must be an IRI with a nonempty label and a nonempty definition in `rdfs:comment` or `markdownDescription`, and must not also be a concept, manifestation or occurrence. Load the Concept Ontology as the ontology graph to supply the built-in definitions.

ekg-concept:StatementPositionShape	
https://ekgf.org/ontology/concept#StatementPositionShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Statement Position Shape
sh:nodeKind	sh:IRI
sh:not	(sh:or: ((sh:class: ekg-concept:DataElementConcept), (sh:class: ekg-concept:ConceptManifestation), (sh:class: ekg-concept:ManifestationOccurrence)))
sh:or	((sh:property: (sh:minCount: 1; sh:path: rdfs:comment)), (sh:property: (sh:minCount: 1; sh:path: ekg-annotation:markdownDescription)))
sh:property	(sh:minLength: 1; sh:nodeKind: sh:Literal; sh:path: rdfs:comment)
sh:property	(sh:minCount: 1; sh:minLength: 1; sh:nodeKind: sh:Literal; sh:path: rdfs:label)
sh:property	(sh:minLength: 1; sh:nodeKind: sh:Literal; sh:path: ekg-annotation:markdownDescription)
sh:targetClass	ekg-concept:StatementPosition
sh:targetObjectsOf	ekg-concept:statementPosition

9.5.5 Template Language Shape

Requires a stable language/profile IRI, a human-readable name, and a reference to its documentation. These metadata do not certify renderer behavior or execution approval.

ekg-concept:TemplateLanguageShape	
https://ekgf.org/ontology/concept#TemplateLanguageShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Template Language Shape
sh:nodeKind	sh:IRI
sh:property	(sh:minCount: 1; sh:minLength: 1; sh:nodeKind: sh:Literal; sh:path: rdfs:label)
sh:property	(sh:minCount: 1; sh:nodeKind: sh:IRI; sh:path: rdfs:seeAlso)
sh:targetClass	ekg-concept:TemplateLanguage
sh:targetObjectsOf	ekg-concept:templateLanguage

9.5.6 Template Parameter Manifestation Shape

Requires a template-placeholder manifestation to identify its language and its exact nonempty token. A shared token is interpreted within that language, not from its punctuation alone.

ekg-concept:TemplateParameterManifestationShape	
https://ekgf.org/ontology/concept#TemplateParameterManifestationShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Template Parameter Manifestation Shape
sh:nodeKind	sh:BlankNodeOrIRI
sh:property	(sh:class: ekg-concept:TemplateLanguage; sh:maxCount: 1; sh:minCount: 1; sh:nodeKind: sh:IRI; sh:path: ekg-concept:templateLanguage)
sh:property	(sh:datatype: xsd:string; sh:maxCount: 1; sh:minCount: 1; sh:minLength: 1; sh:path: ekg-concept:manifestationValue)
sh:targetClass	ekg-concept:TemplateParameterManifestation

10 DataOps Ontology

10.1 Overview

DataOps Ontology

Identifier	Value
Ontology IRI / Namespace	`https://ekgf.org/ontology/dataops#`
Prefix	`ekg-dataops`

A vocabulary for DataOps pipelines, RDF load requests, their processing states, and the single-graph and multi-graph datasets produced during loading and assembly.

10.2 Classes

10.2.1 Failed RDF Load Request

A load request classified as failed because its loading operation did not complete successfully.

`ekg-dataops:FailedLoadRequest`	
`https://ekgf.org/ontology/dataops#FailedLoadRequest`	
RDF predicate	**Value**
`rdf:type`	`owl:Class`
`rdfs:label`	Failed RDF Load Request
`rdfs:subClassOf`	ekg-dataops:LoadRequest

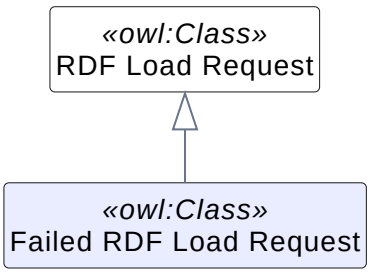


Diagram note

10.2.2 Finished RDF Load Request

A load request classified as finished after its loading operation has completed.

`ekg-dataops:FinishedLoadRequest`	
`https://ekgf.org/ontology/dataops#FinishedLoadRequest`	
RDF predicate	**Value**
`rdf:type`	`owl:Class`
`rdfs:label`	Finished RDF Load Request
`rdfs:subClassOf`	ekg-dataops:LoadRequest

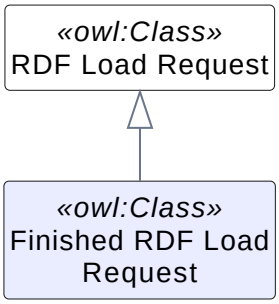


Diagram note

10.2.3 RDF Load Request

A LoadRequest represents an RDF file being loaded into the current RDF Graph Database.

ekg-dataops:LoadRequest	
https://ekgf.org/ontology/dataops#LoadRequest	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	RDF Load Request

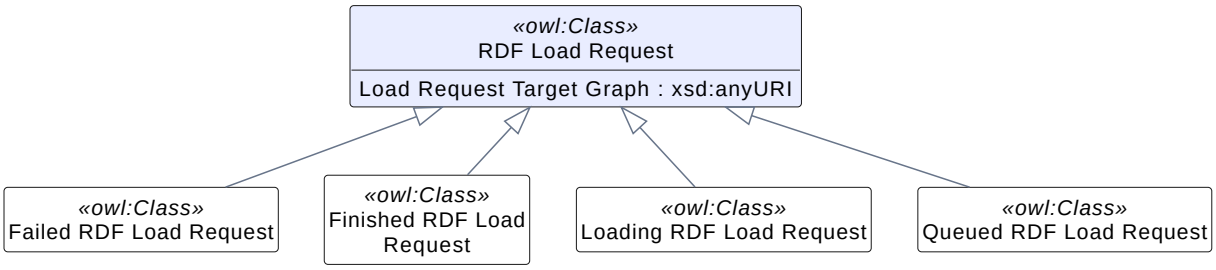


Diagram note

10.2.4 Loading RDF Load Request

A load request whose loading operation is in progress.

ekg-dataops:LoadingLoadRequest	
https://ekgf.org/ontology/dataops#LoadingLoadRequest	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Loading RDF Load Request
rdfs:subClassOf	ekg-dataops:LoadRequest

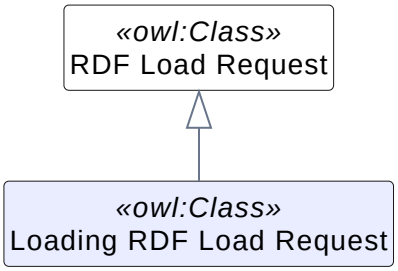


Diagram note

10.2.5 Logical RDF Dataset

A Multi-graph RDF Dataset can be constructed from one or more Single-graph RDF Datasets as they get loaded into a staging environment in the context of a pipeline. Usually the single-graph datasets get merged into the multi-graph dataset later on in the pipeline in order to construct a production-ready final RDF Dataset,

ekg-dataops:MultiGraphDataset	
https://ekgf.org/ontology/dataops#MultiGraphDataset	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Logical RDF Dataset
rdfs:subClassOf	dcat:Dataset

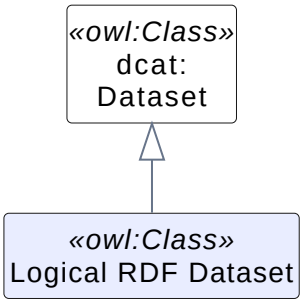


Diagram note

10.2.6 Pipeline

A DataOps process in which load requests bring source data into datasets and subsequent steps can assemble or transform that data.

ekg-dataops:Pipeline	
https://ekgf.org/ontology/dataops#Pipeline	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Pipeline

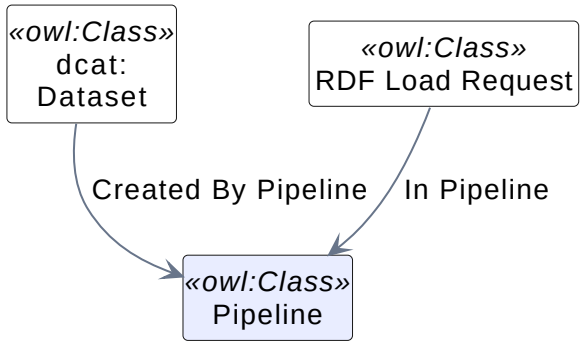


Diagram note

10.2.7 Queued RDF Load Request

A load request waiting to be processed by a pipeline.

ekg-dataops:QueuedLoadRequest	
https://ekgf.org/ontology/dataops#QueuedLoadRequest	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Queued RDF Load Request
rdfs:subClassOf	ekg-dataops:LoadRequest

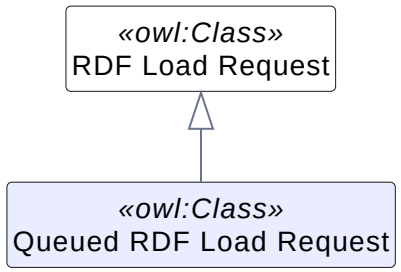


Diagram note

10.2.8 Single-graph RDF Dataset

A single-graph RDF dataset emerges to exist in an RDF graph dataset in the first stage of a pipeline, the load stage, where a given RDF file (usually an N-Triples or Turtle file in an S3 bucket) gets loaded into its own named graph where the IRI of the graph is equal to the S3 file URL. That would be a "single-graph RDF dataset" that usually ends up being merged into a "multi-graph RDF dataset" further down the road.

ekg-dataops:SingleGraphDataset	
https://ekgf.org/ontology/dataops#SingleGraphDataset	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Single-graph RDF Dataset

ekg-dataops:SingleGraphDataset	
https://ekgf.org/ontology/dataops#SingleGraphDataset	
RDF predicate	**Value**
rdfs:subClassOf	dcat:Dataset

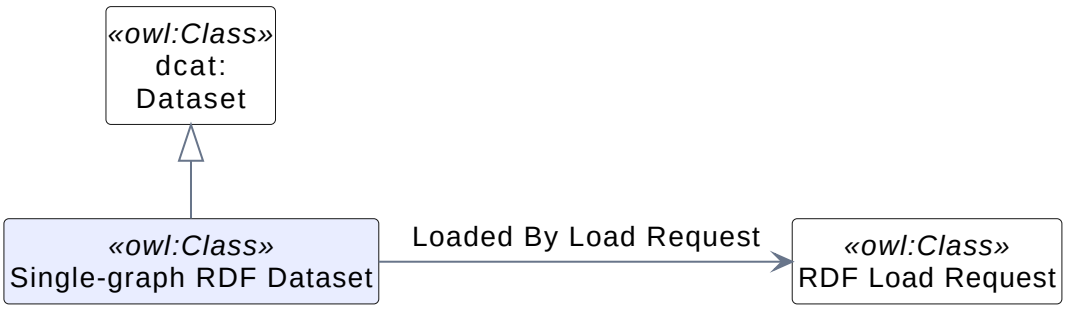


Diagram note

10.3 Properties

10.3.1 Created By Pipeline

Identifies the DataOps pipeline that produced a dataset.

ekg-dataops:createdByPipeline	
https://ekgf.org/ontology/dataops#createdByPipeline	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	dcat:Dataset
rdfs:label	Created By Pipeline
rdfs:range	ekg-dataops:Pipeline

10.3.2 Load Request Target Graph

Records the IRI of the target graph for a load request as an xsd:anyURI literal.

ekg-dataops:graph	
https://ekgf.org/ontology/dataops#graph	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-dataops:LoadRequest
rdfs:label	Load Request Target Graph
rdfs:range	xsd:anyURI

10.3.3 In Pipeline

Identifies the pipeline to which a load request belongs. Each load request has at most one pipeline under the declared functional-property axiom.

ekg-dataops:inPipeline	
https://ekgf.org/ontology/dataops#inPipeline	
RDF predicate	**Value**
rdf:type	owl:FunctionalProperty
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-dataops:LoadRequest
rdfs:label	In Pipeline
rdfs:range	ekg-dataops:Pipeline

10.3.4 Loaded By Load Request

Relates a single-graph dataset to a load request that loaded its data.

ekg-dataops:loadedByLoadRequest	
https://ekgf.org/ontology/dataops#loadedByLoadRequest	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-dataops:SingleGraphDataset
rdfs:label	Loaded By Load Request
rdfs:range	ekg-dataops:LoadRequest

10.3.5 Load Request Source IRI

Records the IRI of the source to be loaded by a load request as an xsd:anyURI literal.

ekg-dataops:source	
https://ekgf.org/ontology/dataops#source	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-dataops:LoadRequest
rdfs:label	Load Request Source IRI
rdfs:range	xsd:anyURI

11 File System Ontology

11.1 Overview

File System Ontology

Identifier	Value
Ontology IRI / Namespace	`https://ekgf.org/ontology/file-system#`
Prefix	`ekg-file-system`

A provider-neutral filesystem vocabulary: rooted views, directories, named directory entries, stable file series, immutable file versions whose content is a MIME-typed literal, symbolic links and service-style shortcuts, external provider objects, tombstones, declared capability profiles, and attachment assertions with explicit tracking modes. Names and parents belong to directory entries, never to files, so hard links, multiple parents, aliases, and provider shortcuts are representable without copying the target. Provider-specific API fields and credentials do not enter this ontology.

11.2 Classes

11.2.1 archive entry kind

The closed set of member entry kinds a decomposed container records.

`ekg-file-system:ArchiveEntryKind`	
`https://ekgf.org/ontology/file-system#ArchiveEntryKind`	
RDF predicate	**Value**
`rdf:type`	`owl:Class`
`rdfs:label`	archive entry kind

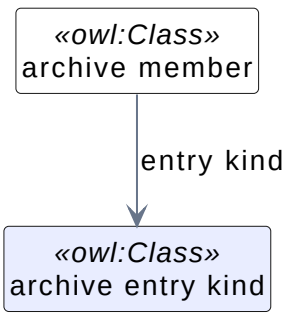


Diagram note

11.2.2 archive format

The closed set of container formats the archive adapter decomposes.

ekg-file-system:ArchiveFormat	
https://ekgf.org/ontology/file-system#ArchiveFormat	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	archive format

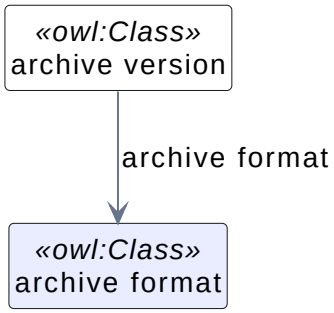


Diagram note

11.2.3 archive member

One entry of a decomposed container in container order: a regular file (targeting its member file version), a nested container (targeting its own archive version), a directory, a link, or a metadata-only entry. Duplicate paths keep distinct ordinals; records preserve the metadata deterministic reconstruction needs.

ekg-file-system:ArchiveMember	
https://ekgf.org/ontology/file-system#ArchiveMember	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	archive member

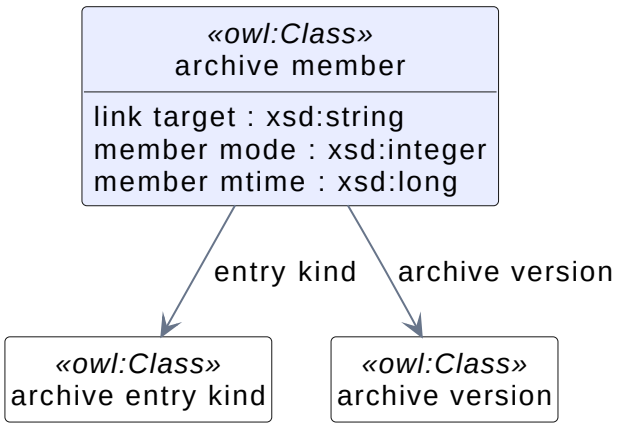


Diagram note

11.2.4 archive version

One immutable decomposed container state: a ZIP, TAR, wrapped archive, or Office package represented as an ordered member tree whose regular members are ordinary file versions. The upload quad points

at this node, never at a compound literal; download reconstructs the container through the deterministic canonical writer. Distinct from workspace directories: archive members are not flattened into the containing directory unless explicitly extracted.

ekg-file-system:ArchiveVersion	
https://ekgf.org/ontology/file-system#ArchiveVersion	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	archive version

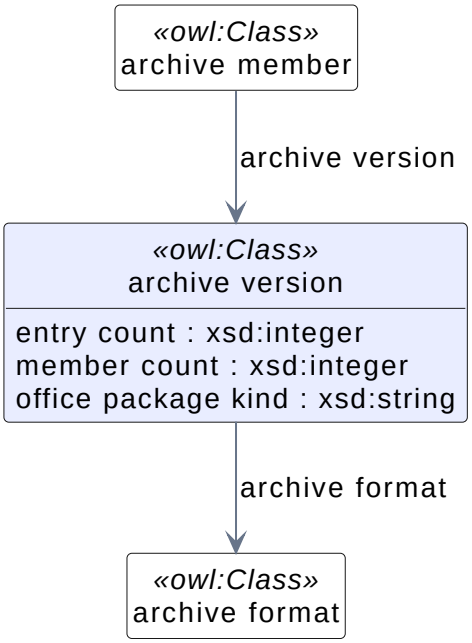


Diagram note

11.2.5 attachment

An assertion relating a graph resource to a file target, with its own author, time, mode, label, role, provenance, and lifecycle. Deleting an attachment deletes the assertion, never the file.

ekg-file-system:Attachment	
https://ekgf.org/ontology/file-system#Attachment	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	attachment

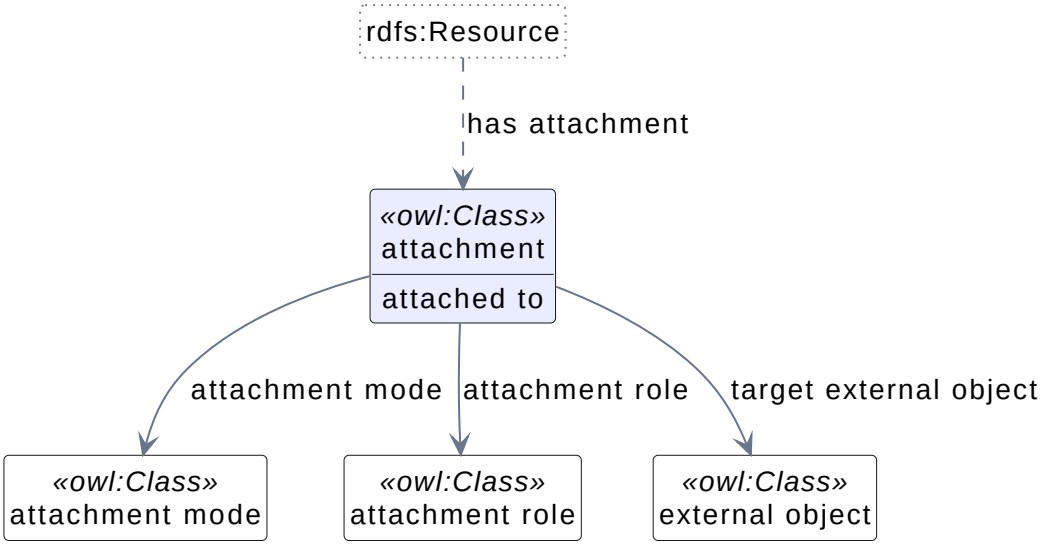


Diagram note

11.2.6 attachment mode

How an attachment resolves its target. The mode is mandatory and is one of the five declared individuals.

ekg-file-system:AttachmentMode	
https://ekgf.org/ontology/file-system#AttachmentMode	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	attachment mode

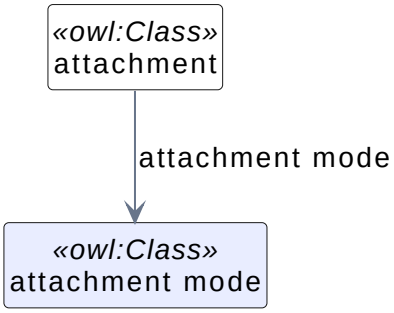


Diagram note

11.2.7 attachment role

The business role a file plays on the resource it is attached to (proposal, contract, invoice, evidence). Workspaces declare their own role individuals.

ekg-file-system:AttachmentRole	
https://ekgf.org/ontology/file-system#AttachmentRole	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	attachment role

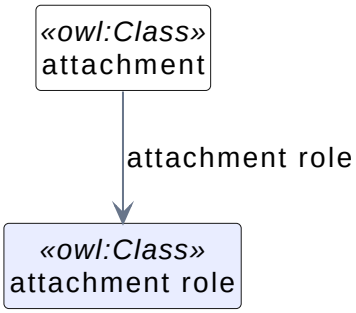


Diagram note

11.2.8 capability profile

Declared behavior of a filesystem namespace: case sensitivity, Unicode normalization, legal names, roots, versioning, links, trash, atomic move, server-side copy, range reads. Adapters map providers onto these semantics; unsupported operations fail explicitly.

ekg-file-system:CapabilityProfile	
https://ekgf.org/ontology/file-system#CapabilityProfile	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	capability profile

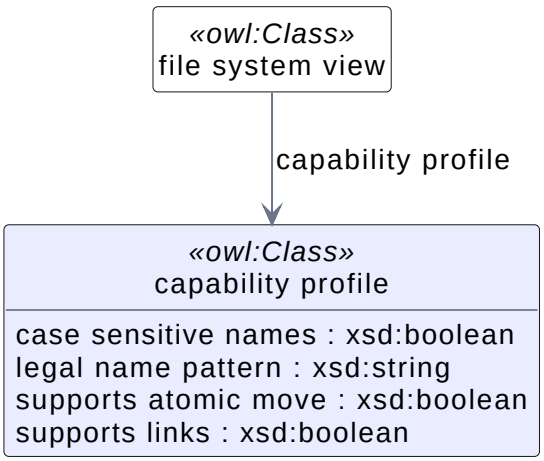


Diagram note

11.2.9 directory

A container resource. A directory may also attach to a business object; membership is asserted by directory entries, never by direct containment edges.

ekg-file-system:Directory	
https://ekgf.org/ontology/file-system#Directory	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	directory

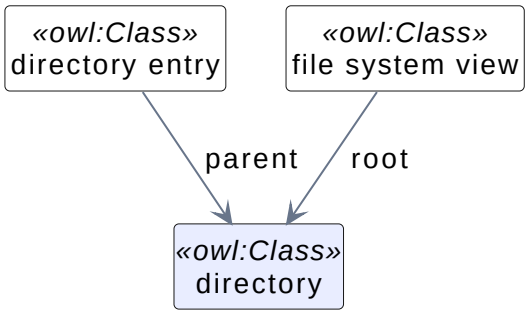


Diagram note

11.2.10 directory entry

One named parent/target assertion in a rooted view. The same target may appear through many entries and names without copying content.

ekg-file-system:DirectoryEntry	
https://ekgf.org/ontology/file-system#DirectoryEntry	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	directory entry

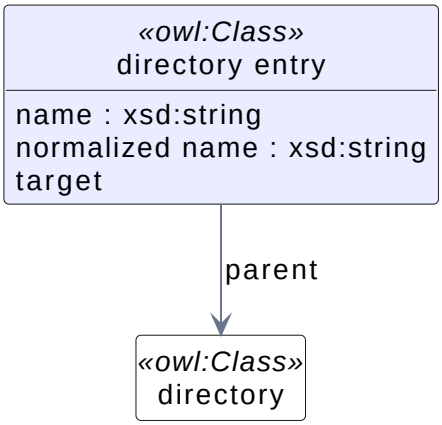


Diagram note

11.2.11 external object

A provider-stable file object identity (not its bytes). Connector workers map provider revisions of an external object onto new file versions.

ekg-file-system:ExternalObject	
https://ekgf.org/ontology/file-system#ExternalObject	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	external object

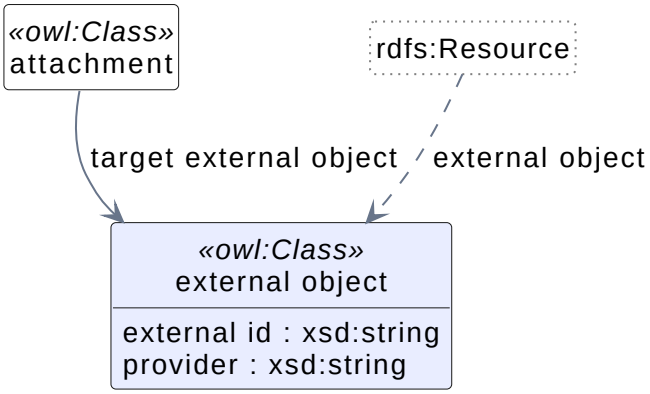


Diagram note

11.2.12 file series

The stable logical document users browse and edit. A series is not content-addressed: it survives byte changes, renames, moves, and provider version updates while its current version changes.

ekg-file-system:FileSeries	
https://ekgf.org/ontology/file-system#FileSeries	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	file series

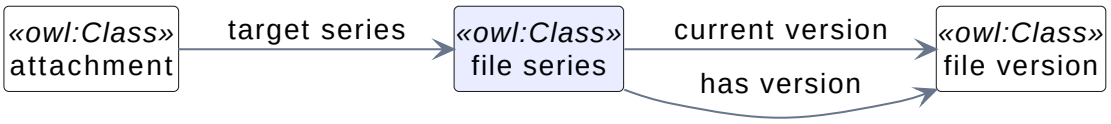


Diagram note

11.2.13 file system view

A branch/commit-scoped rooted projection consumed by Drive or another filesystem client. Paths are presentation addresses within one view and never resource identity.

ekg-file-system:FileSystemView	
https://ekgf.org/ontology/file-system#FileSystemView	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	file system view

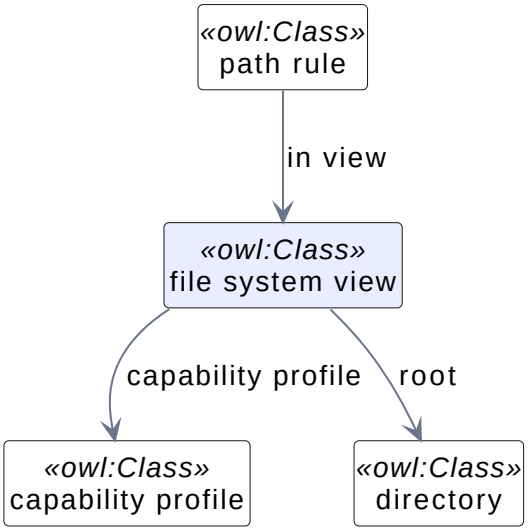


Diagram note

11.2.14 file version

One immutable observed file state: a MIME-typed content literal plus source, size, time, and observation provenance. Correcting metadata creates a new revision or superseding version; historical facts are never mutated in place.

ekg-file-system:FileVersion	
https://ekgf.org/ontology/file-system#FileVersion	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	file version

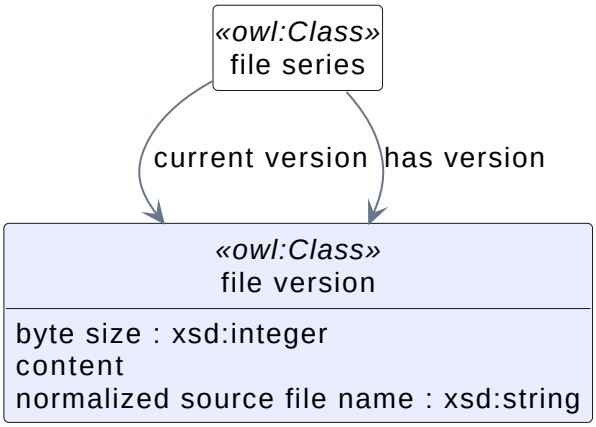


Diagram note

11.2.15 name family rule

An explicit rule declaring that a family of copied file names is one logical document stream. Ambiguous matches never resolve silently; they create a review task.

ekg-file-system:NameFamilyRule	
https://ekgf.org/ontology/file-system#NameFamilyRule	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	name family rule

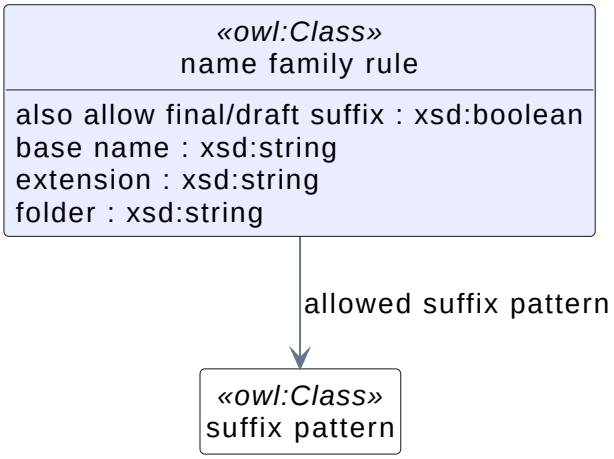


Diagram note

11.2.16 path rule

A path-tracking target: the file series currently occupying a path in a selected view. Zero or multiple matches fail resolution hard.

ekg-file-system:PathRule	
https://ekgf.org/ontology/file-system#PathRule	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	path rule

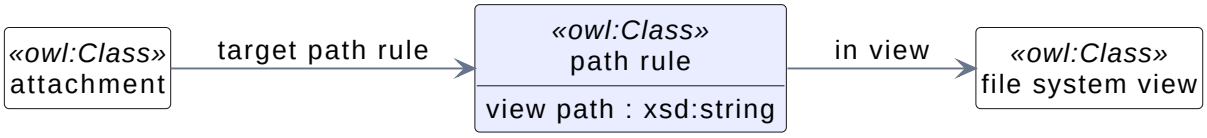


Diagram note

11.2.17 service shortcut

An entry target that refers to an external provider object (a Google Drive or OneDrive shortcut) rather than to content in the workspace.

ekg-file-system:ServiceShortcut	
https://ekgf.org/ontology/file-system#ServiceShortcut	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	service shortcut

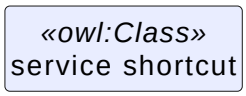


Diagram note

11.2.18 source kind

How a file version was observed.

ekg-file-system:SourceKind	
https://ekgf.org/ontology/file-system#SourceKind	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	source kind

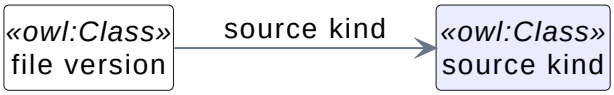


Diagram note

11.2.19 suffix pattern

A frozen family of copy-suffix conventions a name family rule may allow.

ekg-file-system:SuffixPattern	
https://ekgf.org/ontology/file-system#SuffixPattern	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	suffix pattern

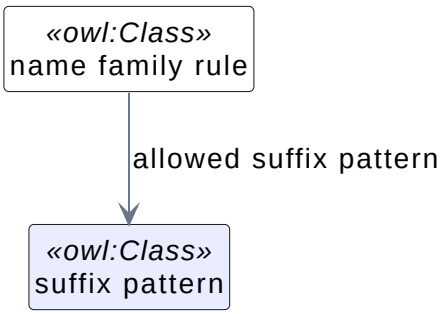


Diagram note

11.2.20 symbolic link

An entry target that names a path to resolve within the same view, POSIX-style. Resolution is bounded and loop-checked before projection.

ekg-file-system:SymbolicLink	
https://ekgf.org/ontology/file-system#SymbolicLink	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	symbolic link

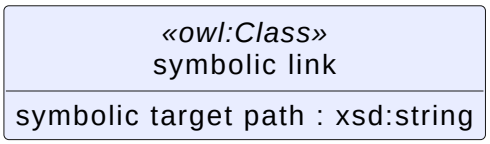


Diagram note

11.2.21 tombstone

A recorded deletion: the target it retires and when. Trash/restore flows read tombstones; content survives while retention or references keep it alive.

ekg-file-system:Tombstone	
https://ekgf.org/ontology/file-system#Tombstone	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	tombstone

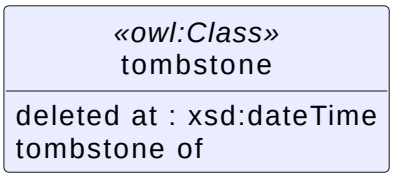


Diagram note

11.3 Properties

11.3.1 allowed suffix pattern

Associates a name-family rule with a suffix pattern that the rule permits.

ekg-file-system:allowedSuffixPattern	
https://ekgf.org/ontology/file-system#allowedSuffixPattern	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-file-system:NameFamilyRule
rdfs:label	allowed suffix pattern
rdfs:range	ekg-file-system:SuffixPattern

11.3.2 also allow final/draft suffix

Records whether a name-family rule also permits a final or draft suffix.

ekg-file-system:alsoAllowFinalDraftSuffix	
https://ekgf.org/ontology/file-system#alsoAllowFinalDraftSuffix	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:NameFamilyRule
rdfs:label	also allow final/draft suffix
rdfs:range	xsd:boolean

11.3.3 archive format

The decomposed container's format.

ekg-file-system:archiveFormat	
https://ekgf.org/ontology/file-system#archiveFormat	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-file-system:ArchiveVersion
rdfs:label	archive format
rdfs:range	ekg-file-system:ArchiveFormat

11.3.4 archive version

The decomposed container this member belongs to.

ekg-file-system:archiveVersion	
https://ekgf.org/ontology/file-system#archiveVersion	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-file-system:ArchiveMember
rdfs:label	archive version
rdfs:range	ekg-file-system:ArchiveVersion

11.3.5 attached to

The resource the attachment is asserted on.

ekg-file-system:attachedTo	
https://ekgf.org/ontology/file-system#attachedTo	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-file-system:Attachment
rdfs:label	attached to

11.3.6 attachment mode

Specifies the attachment mode selected for an attachment.

ekg-file-system:attachmentMode	
https://ekgf.org/ontology/file-system#attachmentMode	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty

ekg-file-system:attachmentMode	
https://ekgf.org/ontology/file-system#attachmentMode	
RDF predicate	**Value**
rdfs:domain	ekg-file-system:Attachment
rdfs:label	attachment mode
rdfs:range	ekg-file-system:AttachmentMode

11.3.7 attachment role

Specifies the role assigned to an attachment.

ekg-file-system:attachmentRole	
https://ekgf.org/ontology/file-system#attachmentRole	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-file-system:Attachment
rdfs:label	attachment role
rdfs:range	ekg-file-system:AttachmentRole

11.3.8 base name

Records the base-name component of a name-family rule, separately from its extension and folder.

ekg-file-system:baseName	
https://ekgf.org/ontology/file-system#baseName	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:NameFamilyRule
rdfs:label	base name
rdfs:range	xsd:string

11.3.9 byte size

The stored value's length in bytes.

ekg-file-system:byteSize	
https://ekgf.org/ontology/file-system#byteSize	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:FileVersion
rdfs:label	byte size
rdfs:range	xsd:integer

11.3.10 capability profile

The declared capabilities of the namespace a view projects.

ekg-file-system:capabilityProfile	
https://ekgf.org/ontology/file-system#capabilityProfile	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-file-system:FileSystemView
rdfs:label	capability profile
rdfs:range	ekg-file-system:CapabilityProfile

11.3.11 case sensitive names

Records whether a capability profile treats file and folder names as case-sensitive.

ekg-file-system:caseSensitiveNames	
https://ekgf.org/ontology/file-system#caseSensitiveNames	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:CapabilityProfile
rdfs:label	case sensitive names
rdfs:range	xsd:boolean

11.3.12 content

The version's complete value: a MIME-typed literal whose datatype is the mime: form of its media type. Large values are external in storage behind their literal manifest; many versions may reference the same literal term and share its chunks without sharing identity.

ekg-file-system:content	
https://ekgf.org/ontology/file-system#content	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:FileVersion
rdfs:label	content

11.3.13 current version

The series' current version on the branch being viewed. Time-travel reads resolve this pointer as of the commit.

ekg-file-system:currentVersion	
https://ekgf.org/ontology/file-system#currentVersion	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-file-system:FileSeries
rdfs:label	current version
rdfs:range	ekg-file-system:FileVersion

11.3.14 deleted at

Records the deletion time represented by a tombstone.

ekg-file-system:deletedAt	
https://ekgf.org/ontology/file-system#deletedAt	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:Tombstone
rdfs:label	deleted at
rdfs:range	xsd:dateTime

11.3.15 entry count

Non-member entries walked in this container.

ekg-file-system:entryCount	
https://ekgf.org/ontology/file-system#entryCount	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:ArchiveVersion
rdfs:label	entry count
rdfs:range	xsd:integer

11.3.16 entry kind

What this member entry is.

ekg-file-system:entryKind	
https://ekgf.org/ontology/file-system#entryKind	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-file-system:ArchiveMember
rdfs:label	entry kind

ekg-file-system:entryKind	
https://ekgf.org/ontology/file-system#entryKind	
RDF predicate	**Value**
rdfs:range	ekg-file-system:ArchiveEntryKind

11.3.17 extension

Records the filename-extension component of a name-family rule.

ekg-file-system:extension	
https://ekgf.org/ontology/file-system#extension	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:NameFamilyRule
rdfs:label	extension
rdfs:range	xsd:string

11.3.18 external id

The provider-stable object identifier. It identifies the external object, not its bytes.

ekg-file-system:externalId	
https://ekgf.org/ontology/file-system#externalId	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:ExternalObject
rdfs:label	external id
rdfs:range	xsd:string

11.3.19 external object

The provider object a shortcut or sync relationship refers to.

ekg-file-system:externalObject	
https://ekgf.org/ontology/file-system#externalObject	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:label	external object
rdfs:range	ekg-file-system:ExternalObject

11.3.20 external revision id

The provider revision a file version was fetched from, when the provider has stable revision IDs.

ekg-file-system:externalRevisionId	
https://ekgf.org/ontology/file-system#externalRevisionId	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:label	external revision id
rdfs:range	xsd:string

11.3.21 folder

Records the folder component of a name-family rule.

ekg-file-system:folder	
https://ekgf.org/ontology/file-system#folder	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:NameFamilyRule
rdfs:label	folder
rdfs:range	xsd:string

11.3.22 has attachment

The resource's attachment assertions.

ekg-file-system:hasAttachment	
https://ekgf.org/ontology/file-system#hasAttachment	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:label	has attachment
rdfs:range	ekg-file-system:Attachment

11.3.23 has version

Every observed version of the series.

ekg-file-system:hasVersion	
https://ekgf.org/ontology/file-system#hasVersion	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-file-system:FileSeries
rdfs:label	has version
rdfs:range	ekg-file-system:FileVersion

11.3.24 in view

The view a path rule resolves its path in.

ekg-file-system:inView	
https://ekgf.org/ontology/file-system#inView	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-file-system:PathRule
rdfs:label	in view
rdfs:range	ekg-file-system:FileSystemView

11.3.25 legal name pattern

A regular expression names must match to be legal in the namespace.

ekg-file-system:legalNamePattern	
https://ekgf.org/ontology/file-system#legalNamePattern	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:CapabilityProfile
rdfs:label	legal name pattern
rdfs:range	xsd:string

11.3.26 link target

A link entry's verbatim target.

ekg-file-system:linkTarget	
https://ekgf.org/ontology/file-system#linkTarget	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:ArchiveMember
rdfs:label	link target
rdfs:range	xsd:string

11.3.27 member count

Regular members walked in this container.

ekg-file-system:memberCount	
https://ekgf.org/ontology/file-system#memberCount	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:ArchiveVersion
rdfs:label	member count
rdfs:range	xsd:integer

11.3.28 member mode

POSIX-style mode bits, when the format carries them — reconstruction metadata.

ekg-file-system:memberMode	
https://ekgf.org/ontology/file-system#memberMode	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:ArchiveMember
rdfs:label	member mode
rdfs:range	xsd:integer

11.3.29 member mtime

Modification time in seconds since the epoch, when the format carries it — reconstruction metadata.

ekg-file-system:memberMtime	
https://ekgf.org/ontology/file-system#memberMtime	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:ArchiveMember
rdfs:label	member mtime
rdfs:range	xsd:long

11.3.30 member ordinal

The member's zero-based position in container order. Duplicate paths keep distinct ordinals; reconstruction replays this order.

ekg-file-system:memberOrdinal	
https://ekgf.org/ontology/file-system#memberOrdinal	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:ArchiveMember

ekg-file-system:memberOrdinal	
https://ekgf.org/ontology/file-system#memberOrdinal	
RDF predicate	**Value**
rdfs:label	member ordinal
rdfs:range	xsd:integer

11.3.31 member path

The member's safe relative path inside the container, verbatim: never absolute, never containing a parent traversal (such containers stay opaque).

ekg-file-system:memberPath	
https://ekgf.org/ontology/file-system#memberPath	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:ArchiveMember
rdfs:label	member path
rdfs:range	xsd:string

11.3.32 member target

What a member resolves to: a regular member's file version, or a nested container's own archive version. Directory, link, and metadata-only entries have no target.

ekg-file-system:memberTarget	
https://ekgf.org/ontology/file-system#memberTarget	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-file-system:ArchiveMember
rdfs:label	member target

11.3.33 name

The entry's name in its parent. Names belong to entries, never to files.

ekg-file-system:name	
https://ekgf.org/ontology/file-system#name	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:DirectoryEntry
rdfs:label	name
rdfs:range	xsd:string

11.3.34 normalized name

The entry name after the view's declared normalization, for duplicate detection.

ekg-file-system:normalizedName	
https://ekgf.org/ontology/file-system#normalizedName	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:DirectoryEntry
rdfs:label	normalized name
rdfs:range	xsd:string

11.3.35 normalized source file name

Records the normalized source filename retained for a file version, separately from the originally observed filename.

ekg-file-system:normalizedSourceFileName	
https://ekgf.org/ontology/file-system#normalizedSourceFileName	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:FileVersion
rdfs:label	normalized source file name
rdfs:range	xsd:string

11.3.36 observed at

Records when a file version was observed by the source integration.

ekg-file-system:observedAt	
https://ekgf.org/ontology/file-system#observedAt	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:FileVersion
rdfs:label	observed at
rdfs:range	xsd:dateTime

11.3.37 office package kind

The Office Open XML package kind (word, excel, powerpoint) when the container is a recognized Office package.

ekg-file-system:officePackageKind	
https://ekgf.org/ontology/file-system#officePackageKind	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:ArchiveVersion
rdfs:label	office package kind
rdfs:range	xsd:string

11.3.38 opaque reason

Why a recognized container could not be decomposed (encryption without a key, unsupported compression, malformed metadata, parser limits). Present only on an opaque archive's file version: its exact source bytes are stored, no member tree is published, and a later explicit reprocessing may retry decomposition.

ekg-file-system:opaqueReason	
https://ekgf.org/ontology/file-system#opaqueReason	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:FileVersion
rdfs:label	opaque reason
rdfs:range	xsd:string

11.3.39 parent

The directory an entry places its target in. Parents belong to entries, so a target may have many parents.

ekg-file-system:parent	
https://ekgf.org/ontology/file-system#parent	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-file-system:DirectoryEntry
rdfs:label	parent
rdfs:range	ekg-file-system:Directory

11.3.40 provider

The provider a connector profile resolves; vendor API fields and credentials stay in DRC connector profiles, never here.

ekg-file-system:provider	
https://ekgf.org/ontology/file-system#provider	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty

ekg-file-system:provider	
https://ekgf.org/ontology/file-system#provider	
RDF predicate	**Value**
rdfs:domain	ekg-file-system:ExternalObject
rdfs:label	provider
rdfs:range	xsd:string

11.3.41 root

The view's root directory.

ekg-file-system:root	
https://ekgf.org/ontology/file-system#root	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-file-system:FileSystemView
rdfs:label	root
rdfs:range	ekg-file-system:Directory

11.3.42 source file name

Records the filename observed for a file version in its source.

ekg-file-system:sourceFileName	
https://ekgf.org/ontology/file-system#sourceFileName	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:FileVersion
rdfs:label	source file name
rdfs:range	xsd:string

11.3.43 source kind

Classifies the source from which a file version was obtained using a SourceKind value.

ekg-file-system:sourceKind	
https://ekgf.org/ontology/file-system#sourceKind	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-file-system:FileVersion
rdfs:label	source kind
rdfs:range	ekg-file-system:SourceKind

11.3.44 supports atomic move

Records whether the source capability profile supports atomic move operations.

ekg-file-system:supportsAtomicMove	
https://ekgf.org/ontology/file-system#supportsAtomicMove	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:CapabilityProfile
rdfs:label	supports atomic move
rdfs:range	xsd:boolean

11.3.45 supports links

Records whether the source capability profile supports links.

ekg-file-system:supportsLinks	
https://ekgf.org/ontology/file-system#supportsLinks	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:CapabilityProfile
rdfs:label	supports links
rdfs:range	xsd:boolean

11.3.46 supports multiple roots

Records whether the source capability profile supports multiple roots.

ekg-file-system:supportsMultipleRoots	
https://ekgf.org/ontology/file-system#supportsMultipleRoots	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:CapabilityProfile
rdfs:label	supports multiple roots
rdfs:range	xsd:boolean

11.3.47 supports range reads

Records whether the source capability profile supports reading a selected byte range.

ekg-file-system:supportsRangeReads	
https://ekgf.org/ontology/file-system#supportsRangeReads	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:CapabilityProfile
rdfs:label	supports range reads
rdfs:range	xsd:boolean

11.3.48 supports server-side copy

Records whether the source capability profile supports copying within the source system without downloading and re-uploading the file.

ekg-file-system:supportsServerSideCopy	
https://ekgf.org/ontology/file-system#supportsServerSideCopy	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:CapabilityProfile
rdfs:label	supports server-side copy
rdfs:range	xsd:boolean

11.3.49 supports shortcuts

Records whether the source capability profile supports shortcut resources.

ekg-file-system:supportsShortcuts	
https://ekgf.org/ontology/file-system#supportsShortcuts	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:CapabilityProfile
rdfs:label	supports shortcuts
rdfs:range	xsd:boolean

11.3.50 supports stable revision ids

Records whether the source capability profile supplies stable revision identifiers.

ekg-file-system:supportsStableRevisionIds	
https://ekgf.org/ontology/file-system#supportsStableRevisionIds	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:CapabilityProfile

ekg-file-system:supportsStableRevisionIds	
https://ekgf.org/ontology/file-system#supportsStableRevisionIds	
RDF predicate	**Value**
rdfs:label	supports stable revision ids
rdfs:range	xsd:boolean

11.3.51 supports trash

Records whether the source capability profile supports a trash or recycle-bin facility.

ekg-file-system:supportsTrash	
https://ekgf.org/ontology/file-system#supportsTrash	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:CapabilityProfile
rdfs:label	supports trash
rdfs:range	xsd:boolean

11.3.52 supports versioning

Records whether the source capability profile supports file versions.

ekg-file-system:supportsVersioning	
https://ekgf.org/ontology/file-system#supportsVersioning	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:CapabilityProfile
rdfs:label	supports versioning
rdfs:range	xsd:boolean

11.3.53 symbolic target path

Records the target path named by a symbolic link.

ekg-file-system:symbolicTargetPath	
https://ekgf.org/ontology/file-system#symbolicTargetPath	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:SymbolicLink
rdfs:label	symbolic target path
rdfs:range	xsd:string

11.3.54 target

What the entry names: a directory, file series, symbolic link, or service shortcut.

ekg-file-system:target	
https://ekgf.org/ontology/file-system#target	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-file-system:DirectoryEntry
rdfs:label	target

11.3.55 target external object

The provider-stable object a track-external-id attachment follows.

ekg-file-system:targetExternalObject	
https://ekgf.org/ontology/file-system#targetExternalObject	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-file-system:Attachment
rdfs:label	target external object
rdfs:range	ekg-file-system:ExternalObject

11.3.56 target name family rule

Identifies the rule governing the filename family of an attachment in its target view.

ekg-file-system:targetNameFamilyRule	
https://ekgf.org/ontology/file-system#targetNameFamilyRule	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-file-system:Attachment
rdfs:label	target name family rule
rdfs:range	ekg-file-system:NameFamilyRule

11.3.57 target path rule

Identifies the rule governing the target-view path of an attachment.

ekg-file-system:targetPathRule	
https://ekgf.org/ontology/file-system#targetPathRule	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty

ekg-file-system:targetPathRule	
https://ekgf.org/ontology/file-system#targetPathRule	
RDF predicate	**Value**
rdfs:domain	ekg-file-system:Attachment
rdfs:label	target path rule
rdfs:range	ekg-file-system:PathRule

11.3.58 target series

The series a track-series attachment follows.

ekg-file-system:targetSeries	
https://ekgf.org/ontology/file-system#targetSeries	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-file-system:Attachment
rdfs:label	target series
rdfs:range	ekg-file-system:FileSeries

11.3.59 target version

The immutable version an exact-version attachment resolves to.

ekg-file-system:targetVersion	
https://ekgf.org/ontology/file-system#targetVersion	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-file-system:Attachment
rdfs:label	target version
rdfs:range	ekg-file-system:FileVersion

11.3.60 tombstone of

The entry, series, or directory the tombstone retires.

ekg-file-system:tombstoneOf	
https://ekgf.org/ontology/file-system#tombstoneOf	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-file-system:Tombstone
rdfs:label	tombstone of

11.3.61 unicode normalization

The normalization form the namespace applies to names (NFC, NFD, none).

ekg-file-system:unicodeNormalization	
https://ekgf.org/ontology/file-system#unicodeNormalization	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:CapabilityProfile
rdfs:label	unicode normalization
rdfs:range	xsd:string

11.3.62 view path

Records the path expression used by a path rule in a target view.

ekg-file-system:viewPath	
https://ekgf.org/ontology/file-system#viewPath	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:PathRule
rdfs:label	view path
rdfs:range	xsd:string

11.4 Other Terms

11.4.1 archive directory entry

An explicit directory entry inside the container's own tree — never a workspace directory.

ekg-file-system:ArchiveDirectoryEntry	
https://ekgf.org/ontology/file-system#ArchiveDirectoryEntry	
RDF predicate	**Value**
rdf:type	ekg-file-system:ArchiveEntryKind
rdfs:label	archive directory entry

11.4.2 exact version

Always resolves to one immutable file version. The default for evidence: compliance, contracts, audit, decisions, invoices, signed documents.

ekg-file-system:ExactVersion	
https://ekgf.org/ontology/file-system#ExactVersion	
RDF predicate	**Value**
rdf:type	ekg-file-system:AttachmentMode
rdfs:label	exact version

11.4.3 gzip

A gzip stream wrapping a single non-archive member.

ekg-file-system:Gzip	
https://ekgf.org/ontology/file-system#Gzip	
RDF predicate	**Value**
rdf:type	ekg-file-system:ArchiveFormat
rdfs:label	gzip

11.4.4 hardlink entry

A hard-link entry; its verbatim target is preserved for reconstruction.

ekg-file-system:HardlinkEntry	
https://ekgf.org/ontology/file-system#HardlinkEntry	
RDF predicate	**Value**
rdf:type	ekg-file-system:ArchiveEntryKind
rdfs:label	hardlink entry

11.4.5 native upload

The bytes arrived through a direct upload into the workspace.

ekg-file-system:NativeUpload	
https://ekgf.org/ontology/file-system#NativeUpload	
RDF predicate	**Value**
rdf:type	ekg-file-system:SourceKind
rdfs:label	native upload

11.4.6 nested archive

A member that is itself a recognized container, decomposed recursively into its own archive version.

ekg-file-system:NestedArchive	
https://ekgf.org/ontology/file-system#NestedArchive	
RDF predicate	**Value**
rdf:type	ekg-file-system:ArchiveEntryKind
rdfs:label	nested archive

11.4.7 other entry

A recognized entry carried as metadata only (device nodes, FIFOs), preserved by its format-specific tag.

ekg-file-system:OtherEntry	
https://ekgf.org/ontology/file-system#OtherEntry	
RDF predicate	**Value**
rdf:type	ekg-file-system:ArchiveEntryKind
rdfs:label	other entry

11.4.8 provider sync

The bytes were fetched from an external provider by a connector worker.

ekg-file-system:ProviderSync	
https://ekgf.org/ontology/file-system#ProviderSync	
RDF predicate	**Value**
rdf:type	ekg-file-system:SourceKind
rdfs:label	provider sync

11.4.9 regular file

A regular member whose decompressed bytes are an ordinary member file version.

ekg-file-system:RegularFile	
https://ekgf.org/ontology/file-system#RegularFile	
RDF predicate	**Value**
rdf:type	ekg-file-system:ArchiveEntryKind
rdfs:label	regular file

11.4.10 symlink entry

A symbolic-link entry; its verbatim target is preserved for reconstruction.

ekg-file-system:SymlinkEntry	
https://ekgf.org/ontology/file-system#SymlinkEntry	
RDF predicate	**Value**
rdf:type	ekg-file-system:ArchiveEntryKind
rdfs:label	symlink entry

11.4.11 tar

A TAR archive.

ekg-file-system:Tar	
https://ekgf.org/ontology/file-system#Tar	
RDF predicate	**Value**
rdf:type	ekg-file-system:ArchiveFormat
rdfs:label	tar

11.4.12 tar+gzip

A gzip stream wrapping a TAR archive.

ekg-file-system:TarGzip	
https://ekgf.org/ontology/file-system#TarGzip	
RDF predicate	**Value**
rdf:type	ekg-file-system:ArchiveFormat
rdfs:label	tar+gzip

11.4.13 track external id

Follows a provider-stable file ID; every fetched provider revision still becomes a new file version in one series.

ekg-file-system:TrackExternalId	
https://ekgf.org/ontology/file-system#TrackExternalId	
RDF predicate	**Value**
rdf:type	ekg-file-system:AttachmentMode
rdfs:label	track external id

11.4.14 track name family

Follows an explicit name-family rule treating copied names as one document stream.

ekg-file-system:TrackNameFamily	
https://ekgf.org/ontology/file-system#TrackNameFamily	
RDF predicate	**Value**
rdf:type	ekg-file-system:AttachmentMode
rdfs:label	track name family

11.4.15 track path

Follows whatever series currently occupies a path in a selected view. A view address, not identity; broken or ambiguous resolution fails hard.

ekg-file-system:TrackPath	
https://ekgf.org/ontology/file-system#TrackPath	
RDF predicate	**Value**
rdf:type	ekg-file-system:AttachmentMode
rdfs:label	track path

11.4.16 track series

Resolves the series' current version as of the branch/commit being viewed — the living document.

ekg-file-system:TrackSeries	
https://ekgf.org/ontology/file-system#TrackSeries	
RDF predicate	**Value**
rdf:type	ekg-file-system:AttachmentMode
rdfs:label	track series

11.4.17 windows copy suffix

The " (2)", " (3)" copy suffixes Windows appends to duplicated names.

ekg-file-system:WindowsCopySuffix	
https://ekgf.org/ontology/file-system#WindowsCopySuffix	
RDF predicate	**Value**
rdf:type	ekg-file-system:SuffixPattern
rdfs:label	windows copy suffix

11.4.18 zip

A ZIP archive, including Office Open XML packages.

ekg-file-system:Zip	
https://ekgf.org/ontology/file-system#Zip	
RDF predicate	**Value**
rdf:type	ekg-file-system:ArchiveFormat
rdfs:label	zip

12 Lifecycle Ontology

12.1 Overview

Lifecycle Ontology

Identifier	Value
Ontology IRI / Namespace	`https://ekgf.org/ontology/lifecycle#`
Prefix	`ekg-lifecycle`

Everything has a lifecycle: use cases, stories, concepts, personas, outcomes, data products and datasets. This ontology says what a lifecycle is, which lifecycle a thing follows and which state the thing is in. It depends on none of the ontologies that describe those things.

A lifecycle has a name, states and permitted transitions, optionally grouped into phases. Its states and transitions form a directed graph; it does not require a separate RDF named graph. Multiple lifecycles can share an RDF graph, with inLifecycle expressing membership independently of graph placement. State order controls presentation, not which moves are permitted. There can be any number of lifecycles. This ontology defines none: the well-known lifecycles are a dataset (dataset/well-known-lifecycles.ttl), and anyone can define more. The registry uses a separate namespace for each lifecycle to distinguish shared state names such as Draft and Deprecated. This is an IRI naming convention, not a graph-storage requirement; custom lifecycles can use another IRI naming convention.

Which lifecycles a thing follows is decided in this order:

1. The complete set of lifecycles the thing names itself, with followsLifecycle. 2. Otherwise, the union of lifecycles that apply to its classes (including superclasses), with appliesTo.

A thing has one inState history head per effective lifecycle. Each head is a StateTransition occurrence whose currentState belongs to that lifecycle. priorStateTransition links earlier occurrences; the lifecycle is derived from each state, and optional provenance is recorded on the occurrence.

12.2 Classes

12.2.1 Lifecycle

A lifecycle with a name, states and permitted transitions, optionally grouped into phases. Membership is expressed with inLifecycle and does not require a separate RDF named graph.

`ekg-lifecycle:Lifecycle`	
`https://ekgf.org/ontology/lifecycle#Lifecycle`	
RDF predicate	**Value**
`rdf:type`	`owl:Class`
`rdfs:label`	Lifecycle

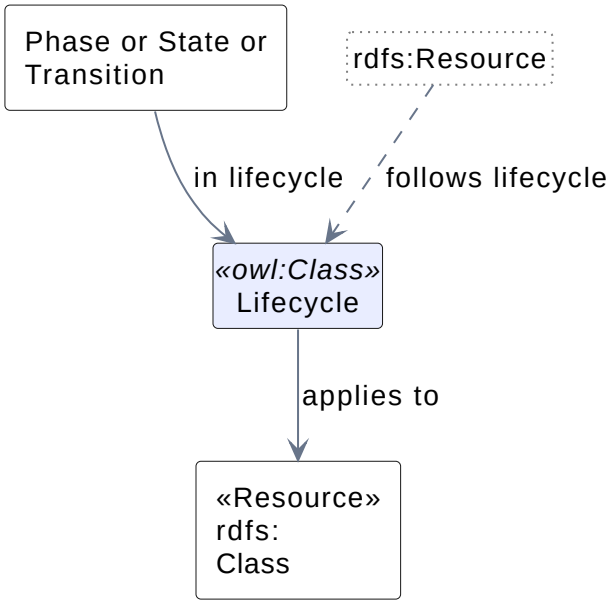


Diagram note

12.2.2 Phase

A high-level part of a lifecycle that groups states.

ekg-lifecycle:Phase	
https://ekgf.org/ontology/lifecycle#Phase	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Phase

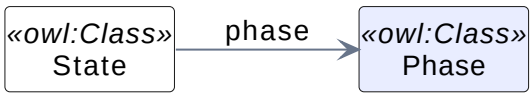


Diagram note

12.2.3 State

One state of a lifecycle.

ekg-lifecycle:State	
https://ekgf.org/ontology/lifecycle#State	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	State

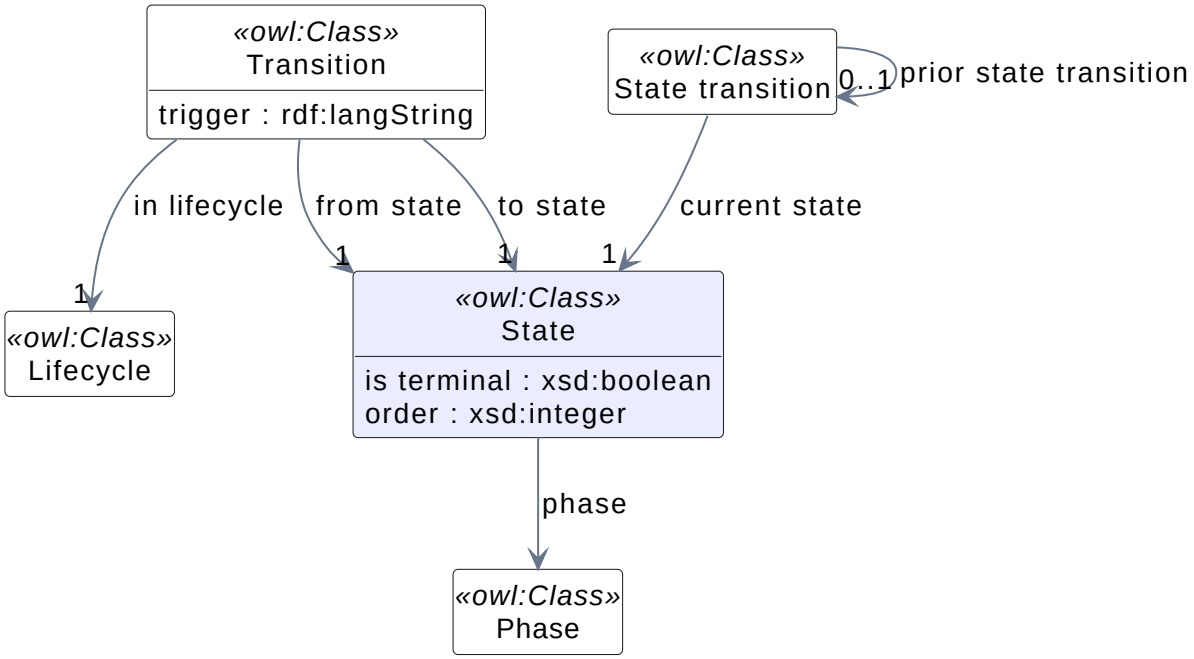


Diagram note

12.2.4 State transition

An actual state-entry occurrence for one thing in one lifecycle, distinct from a permitted Transition definition. It records exactly one currentState and optionally one priorStateTransition. The first occurrence records an initial state without a predecessor. Occurrences can carry optional PROV-O properties and can be identified by IRIs or blank nodes.

ekg-lifecycle:StateTransition	
https://ekgf.org/ontology/lifecycle#StateTransition	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	State transition
owl:disjointWith	ekg-lifecycle:Transition

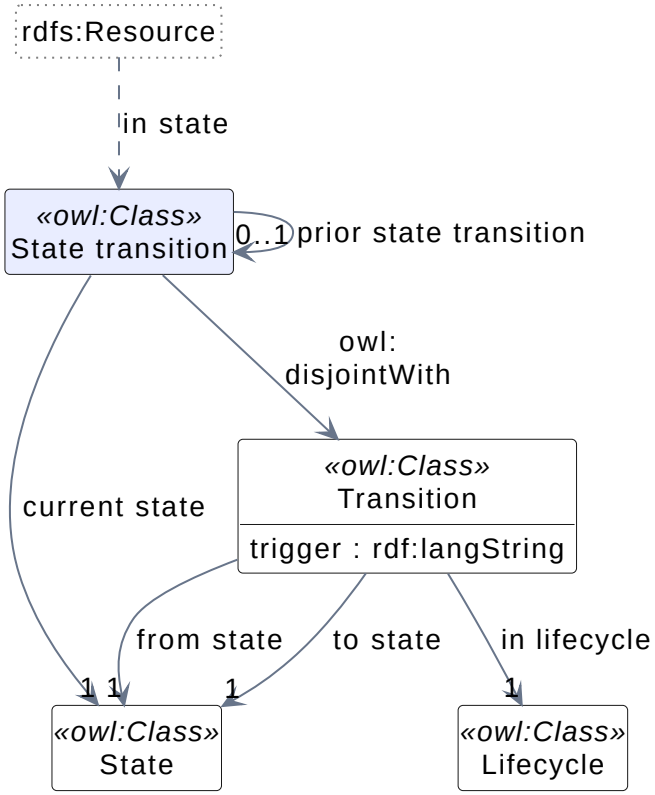


Diagram note

12.2.5 Transition

A permitted directed move between two states of one lifecycle. This describes a possible move, not an occurrence of a state change. Multiple transitions may have the same endpoints and different triggers; loops and branches are explicit.

ekg-lifecycle:Transition	
https://ekgf.org/ontology/lifecycle#Transition	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Transition

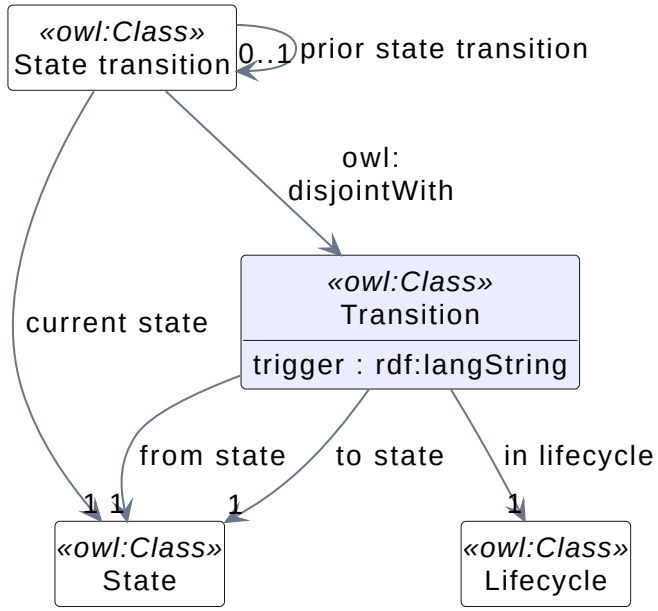


Diagram note

12.3 Properties

12.3.1 applies to

A class whose members follow this lifecycle unless they name another one. This is policy rather than vocabulary: an organization states it in its own data, and can state it differently. It is an annotation property because its values are classes, which are not individuals.

ekg-lifecycle:appliesTo	
https://ekgf.org/ontology/lifecycle#appliesTo	
RDF predicate	**Value**
rdf:type	owl:AnnotationProperty
rdfs:domain	ekg-lifecycle:Lifecycle
rdfs:label	applies to
rdfs:range	rdfs:Class

12.3.2 current state

The state entered by this occurrence. Its inLifecycle identifies the lifecycle; the occurrence does not duplicate that assignment.

ekg-lifecycle:currentState	
https://ekgf.org/ontology/lifecycle#currentState	
RDF predicate	**Value**
rdf:type	owl:FunctionalProperty
rdf:type	owl:ObjectProperty

ekg-lifecycle:currentState	
https://ekgf.org/ontology/lifecycle#currentState	
RDF predicate	**Value**
rdfs:domain	ekg-lifecycle:StateTransition
rdfs:label	current state
rdfs:range	ekg-lifecycle:State

12.3.3 follows lifecycle

A lifecycle this thing follows. When explicit assignments exist, their complete set replaces the class defaults from appliesTo. A thing can follow several lifecycles.

ekg-lifecycle:followsLifecycle	
https://ekgf.org/ontology/lifecycle#followsLifecycle	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:label	follows lifecycle
rdfs:range	ekg-lifecycle:Lifecycle

12.3.4 from state

The source state of a permitted transition.

ekg-lifecycle:fromState	
https://ekgf.org/ontology/lifecycle#fromState	
RDF predicate	**Value**
rdf:type	owl:FunctionalProperty
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-lifecycle:Transition
rdfs:label	from state
rdfs:range	ekg-lifecycle:State

12.3.5 in lifecycle

The lifecycle that a phase, state or transition belongs to.

ekg-lifecycle:inLifecycle	
https://ekgf.org/ontology/lifecycle#inLifecycle	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	(rdf:type: owl:Class; owl:unionOf: (ekg-lifecycle:Phase, ekg-lifecycle:State, ekg-lifecycle:Transition))
rdfs:label	in lifecycle

ekg-lifecycle:inLifecycle	
https://ekgf.org/ontology/lifecycle#inLifecycle	
RDF predicate	**Value**
rdfs:range	ekg-lifecycle:Lifecycle

12.3.6 in state

Links a thing to its current state-entry occurrence, one per effective lifecycle. Earlier occurrences are reached through priorStateTransition. It has no domain because anything can follow lifecycles.

ekg-lifecycle:inState	
https://ekgf.org/ontology/lifecycle#inState	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:label	in state
rdfs:range	ekg-lifecycle:StateTransition

12.3.7 is terminal

True for a state that nothing follows. A lifecycle can end in several ways, such as done, failed or cancelled; each of those is a terminal state, and their order is only the order in which they are listed.

ekg-lifecycle:isTerminal	
https://ekgf.org/ontology/lifecycle#isTerminal	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-lifecycle:State
rdfs:label	is terminal
rdfs:range	xsd:boolean

12.3.8 order

Display position of a state within its lifecycle. Order does not imply a permitted transition.

ekg-lifecycle:order	
https://ekgf.org/ontology/lifecycle#order	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-lifecycle:State
rdfs:label	order

ekg-lifecycle:order	
https://ekgf.org/ontology/lifecycle#order	
RDF predicate	**Value**
rdfs:range	xsd:integer

12.3.9 phase

The phase that a state belongs to.

ekg-lifecycle:phase	
https://ekgf.org/ontology/lifecycle#phase	
RDF predicate	**Value**
rdf:type	owl:FunctionalProperty
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-lifecycle:State
rdfs:label	phase
rdfs:range	ekg-lifecycle:Phase

12.3.10 prior state transition

The immediately preceding occurrence for the same thing and lifecycle. Absent for an initial recorded state. History chains must be acyclic even when permitted state routes contain loops.

ekg-lifecycle:priorStateTransition	
https://ekgf.org/ontology/lifecycle#priorStateTransition	
RDF predicate	**Value**
rdf:type	owl:FunctionalProperty
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-lifecycle:StateTransition
rdfs:label	prior state transition
rdfs:range	ekg-lifecycle:StateTransition

12.3.11 to state

The target state of a permitted transition.

ekg-lifecycle:toState	
https://ekgf.org/ontology/lifecycle#toState	
RDF predicate	**Value**
rdf:type	owl:FunctionalProperty
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-lifecycle:Transition

ekg-lifecycle:toState	
https://ekgf.org/ontology/lifecycle#toState	
RDF predicate	**Value**
rdfs:label	to state
rdfs:range	ekg-lifecycle:State

12.3.12 trigger

An optional human-readable description of the event that enables a transition. It is not an executable guard or an authorization rule. Omitting it leaves the trigger unspecified.

ekg-lifecycle:trigger	
https://ekgf.org/ontology/lifecycle#trigger	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-lifecycle:Transition
rdfs:label	trigger
rdfs:range	rdf:langString

12.4 Shapes

12.4.1 Current State Shape

Validates effective lifecycle selection and current history-head assignments, including explicitly selected lifecycles and class defaults.

ekg-lifecycle:CurrentStateShape	
https://ekgf.org/ontology/lifecycle#CurrentStateShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Current State Shape
sh:property	(sh:class: ekg-lifecycle:Lifecycle; sh:path: ekg-lifecycle:followsLifecycle)
sh:property	(sh:class: ekg-lifecycle:StateTransition; sh:path: ekg-lifecycle:inState)
sh:sparql	(sh:message: A current history head cannot have a recorded successor.; sh:select: SPARQL SELECT query 1)
sh:sparql	(sh:message: A current history head must belong to an effective lifecycle.; sh:select: SPARQL SELECT query 2)
sh:sparql	(sh:message: An object must have only one current history head per lifecycle.; sh:select: SPARQL SELECT query 3)
sh:sparql	(sh:message: Each effective lifecycle requires a current history head.; sh:select: SPARQL SELECT query 4)
sh:targetSubjectsOf	rdf:type
sh:targetSubjectsOf	ekg-lifecycle:followsLifecycle

ekg-lifecycle:CurrentStateShape	
https://ekgf.org/ontology/lifecycle#CurrentStateShape	
RDF predicate	**Value**
sh:targetSubjectsOf	ekg-lifecycle:inState

12.4.1.1 SPARQL query 1

A current history head cannot have a recorded successor.

```
PREFIX lifecycle: <https://ekgf.org/ontology/lifecycle#>
SELECT $this WHERE {
    $this lifecycle:inState ?head .
    ?successor lifecycle:priorStateTransition ?head .
}
```

12.4.1.2 SPARQL query 2

A current history head must belong to an effective lifecycle.

```
PREFIX lifecycle: <https://ekgf.org/ontology/lifecycle#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT $this WHERE {
    $this lifecycle:inState/lifecycle:currentState/lifecycle:inLifecycle ?lc .
    FILTER NOT EXISTS { $this lifecycle:followsLifecycle ?lc }
    FILTER (
        EXISTS { $this lifecycle:followsLifecycle ?explicit }
        || NOT EXISTS {
            $this rdf:type/rdfs:subClassOf* ?class .
            ?lc lifecycle:appliesTo ?class .
        }
    )
}
```

12.4.1.3 SPARQL query 3

An object must have only one current history head per lifecycle.

```
PREFIX lifecycle: <https://ekgf.org/ontology/lifecycle#>
SELECT $this WHERE {
    $this lifecycle:inState ?head , ?other .
    ?head lifecycle:currentState/lifecycle:inLifecycle ?lc .
    ?other lifecycle:currentState/lifecycle:inLifecycle ?lc .
    FILTER (?head != ?other)
}
```

12.4.1.4 SPARQL query 4

Each effective lifecycle requires a current history head.

```
PREFIX lifecycle: <https://ekgf.org/ontology/lifecycle#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT $this WHERE {
    {
```

```
        $this lifecycle:followsLifecycle ?lc .
    } UNION {
        FILTER NOT EXISTS { $this lifecycle:followsLifecycle ?explicit }
        $this rdf:type/rdfs:subClassOf* ?class .
        ?lc lifecycle:appliesTo ?class .
    }
    FILTER NOT EXISTS {
        $this lifecycle:inState/lifecycle:currentState/lifecycle:inLifecycle ?lc .
    }
}
```

12.4.2 History State Shape

Validates the State resources referenced by history occurrences, requiring each state to identify exactly one lifecycle.

ekg-lifecycle:HistoryStateShape	
https://ekgf.org/ontology/lifecycle#HistoryStateShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	History State Shape
sh:class	ekg-lifecycle:State
sh:property	(sh:class: ekg-lifecycle:Lifecycle; sh:maxCount: 1; sh:minCount: 1; sh:path: ekg-lifecycle:inLifecycle)
sh:targetObjectsOf	ekg-lifecycle:currentState

12.4.3 Legacy State Shape

Rejects use of the removed hasState predicate. Current assignments use inState with StateTransition occurrences instead.

ekg-lifecycle:LegacyStateShape	
https://ekgf.org/ontology/lifecycle#LegacyStateShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Legacy State Shape
sh:property	(sh:maxCount: 0; sh:message: hasState was removed. Use inState with a State-Transition occurrence.; sh:path: ekg-lifecycle:hasState)
sh:targetSubjectsOf	ekg-lifecycle:hasState

12.4.4 State Transition Shape

Validates a recorded state-transition event, including its current state and predecessor transition where supplied.

ekg-lifecycle:StateTransitionShape	
https://ekgf.org/ontology/lifecycle#StateTransitionShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	State Transition Shape
sh:class	ekg-lifecycle:StateTransition
sh:not	(sh:class: ekg-lifecycle:Transition)
sh:property	(sh:class: ekg-lifecycle:State; sh:maxCount: 1; sh:minCount: 1; sh:path: ekg-lifecycle:currentState)
sh:property	(sh:class: ekg-lifecycle:StateTransition; sh:maxCount: 1; sh:path: ekg-lifecycle:priorStateTransition)
sh:sparql	(sh:message: A state-entry occurrence cannot belong to histories of different objects.; sh:select: SPARQL SELECT query 1)
sh:sparql	(sh:message: State history must not contain predecessor cycles.; sh:select: SPARQL SELECT query 2)
sh:sparql	(sh:message: Successive occurrences must follow a permitted transition within the same lifecycle.; sh:select: SPARQL SELECT query 3)
sh:targetClass	ekg-lifecycle:StateTransition
sh:targetObjectsOf	ekg-lifecycle:inState
sh:targetObjectsOf	ekg-lifecycle:priorStateTransition
sh:targetSubjectsOf	ekg-lifecycle:currentState
sh:targetSubjectsOf	ekg-lifecycle:priorStateTransition

12.4.4.1 SPARQL query 1

A state-entry occurrence cannot belong to histories of different objects.

```
PREFIX lifecycle: <https://ekgf.org/ontology/lifecycle#>
SELECT $this WHERE {
    ?object lifecycle:inState/lifecycle:priorStateTransition* $this .
    ?other lifecycle:inState/lifecycle:priorStateTransition* $this .
    FILTER (?object != ?other)
}
```

12.4.4.2 SPARQL query 2

State history must not contain predecessor cycles.

```
PREFIX lifecycle: <https://ekgf.org/ontology/lifecycle#>
SELECT $this WHERE {
    $this lifecycle:priorStateTransition+ $this .
}
```

12.4.4.3 SPARQL query 3

Successive occurrences must follow a permitted transition within the same lifecycle.

```
PREFIX lifecycle: <https://ekgf.org/ontology/lifecycle#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
```

```
SELECT $this WHERE {
    $this lifecycle:currentState ?to ;
          lifecycle:priorStateTransition/lifecycle:currentState ?from .
    FILTER NOT EXISTS {
        ?from lifecycle:inLifecycle ?lc .
        ?to lifecycle:inLifecycle ?lc .
        ?route rdf:type lifecycle:Transition ;
               lifecycle:inLifecycle ?lc ;
               lifecycle:fromState ?from ;
               lifecycle:toState ?to .
    }
}
```

12.4.5 Transition Shape

Validates a lifecycle transition's lifecycle, source state, destination state, and trigger.

ekg-lifecycle:TransitionShape	
https://ekgf.org/ontology/lifecycle#TransitionShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Transition Shape
sh:class	ekg-lifecycle:Transition
sh:property	(sh:class: ekg-lifecycle:State; sh:maxCount: 1; sh:minCount: 1; sh:path: ekg-lifecycle:toState)
sh:property	(sh:datatype: rdf:langString; sh:path: ekg-lifecycle:trigger)
sh:property	(sh:class: ekg-lifecycle:State; sh:maxCount: 1; sh:minCount: 1; sh:path: ekg-lifecycle:fromState)
sh:property	(sh:class: ekg-lifecycle:Lifecycle; sh:maxCount: 1; sh:minCount: 1; sh:path: ekg-lifecycle:inLifecycle)
sh:sparql	(sh:message: A terminal state cannot be the source of a transition.; sh:select: SPARQL SELECT query 1)
sh:sparql	(sh:message: Both endpoint states must belong only to the transition's lifecycle.; sh:select: SPARQL SELECT query 2)
sh:targetClass	ekg-lifecycle:Transition
sh:targetSubjectsOf	ekg-lifecycle:fromState
sh:targetSubjectsOf	ekg-lifecycle:toState
sh:targetSubjectsOf	ekg-lifecycle:trigger

12.4.5.1 SPARQL query 1

A terminal state cannot be the source of a transition.

```
PREFIX lifecycle: <https://ekgf.org/ontology/lifecycle#>
SELECT $this WHERE {
    $this lifecycle:fromState ?state .
    ?state lifecycle:isTerminal true .
}
```

12.4.5.2 SPARQL query 2

Both endpoint states must belong only to the transition's lifecycle.

```
PREFIX lifecycle: <https://ekgf.org/ontology/lifecycle#>
SELECT $this WHERE {
    $this lifecycle:inLifecycle ?lifecycle ;
          (lifecycle:fromState|lifecycle:toState) ?state .
    FILTER (
        NOT EXISTS { ?state lifecycle:inLifecycle ?lifecycle }
        || EXISTS {
            ?state lifecycle:inLifecycle ?other .
            FILTER (?other != ?lifecycle)
        }
    )
}
```

13 Maturity Model Ontology

13.1 Overview

Maturity Model Ontology

Identifier	Value
Ontology IRI	`https://ekgf.org/ontology/ekgf-maturity-model`
Namespace	`https://ekgf.org/ontology/ekgf-maturity-model#`
Prefix	`ekg-mm`

A vocabulary for organizing maturity models into pillars, capability areas, capabilities, and levels, with an example EKGF model containing four pillars and five numbered levels.

13.2 Classes

13.2.1 Capability

An ability assessed within the maturity model and associated with a capability area.

`ekg-mm:Capability`	
`https://ekgf.org/ontology/ekgf-maturity-model#Capability`	
RDF predicate	**Value**
`rdf:type`	`owl:Class`
`rdfs:label`	Capability
`ekg-mm:iriLocalTypeName`	capability

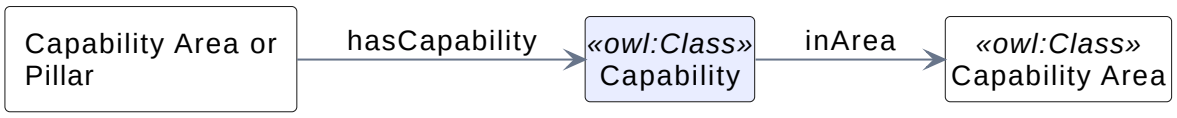


Diagram note

13.2.2 Capability Area

A subject area within a maturity-model pillar that organizes related capabilities.

`ekg-mm:CapabilityArea`	
`https://ekgf.org/ontology/ekgf-maturity-model#CapabilityArea`	
RDF predicate	**Value**
`rdf:type`	`owl:Class`
`rdfs:label`	Capability Area
`ekg-mm:iriLocalTypeName`	capability-area

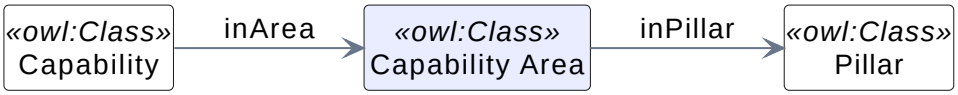


Diagram note

13.2.3 Level

A maturity level defined within a particular maturity model.

ekg-mm:Level	
https://ekgf.org/ontology/ekgf-maturity-model#Level	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Level
ekg-mm:iriLocalTypeName	level

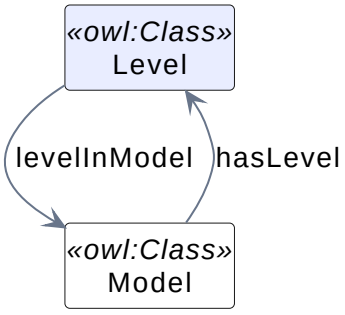


Diagram note

13.2.4 Model

An identified maturity assessment model organizing pillars and levels.

ekg-mm:Model	
https://ekgf.org/ontology/ekgf-maturity-model#Model	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Model
ekg-mm:iriLocalTypeName	model

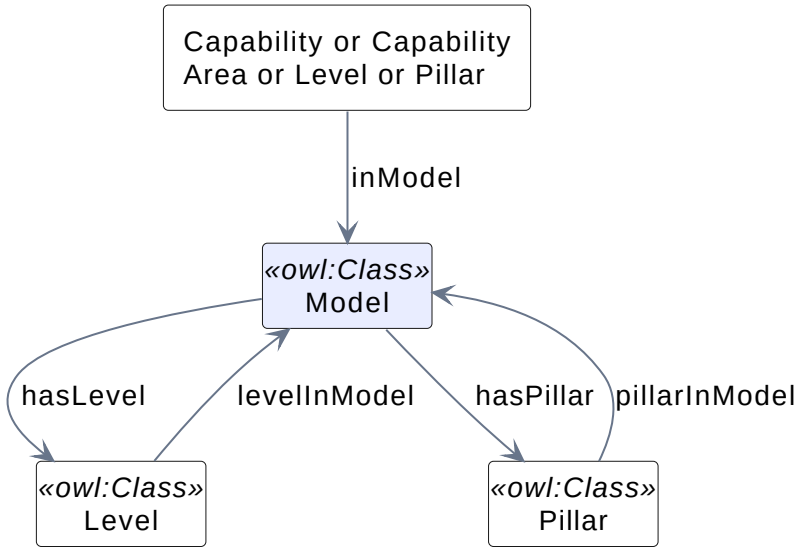


Diagram note

13.2.5 Pillar

A dimension of a maturity model, such as business, data, organization, or technology.

ekg-mm:Pillar	
https://ekgf.org/ontology/ekgf-maturity-model#Pillar	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Pillar
ekg-mm:iriLocalTypeName	pillar

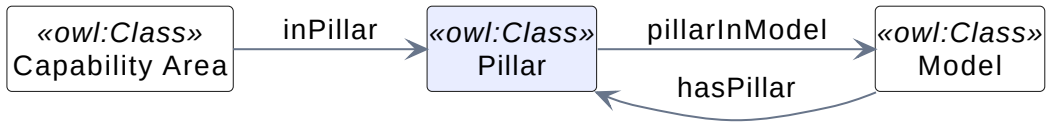


Diagram note

13.3 Properties

13.3.1 Capability Number

A capability number in the format `<pillar>.<area>.<capability>` where `<pillar>` is an uppercase letter and `<area>` and `<capability>` are integer numbers.

ekg-mm:capabilityNumber	
https://ekgf.org/ontology/ekgf-maturity-model#capabilityNumber	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty

ekg-mm:capabilityNumber	
https://ekgf.org/ontology/ekgf-maturity-model#capabilityNumber	
RDF predicate	**Value**
rdf:type	owl:FunctionalProperty
rdfs:label	Capability Number
rdfs:range	xsd:string

13.3.2 hasCapability

Associates a pillar or capability area with a capability it includes.

ekg-mm:hasCapability	
https://ekgf.org/ontology/ekgf-maturity-model#hasCapability	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	(rdf:type: owl:Class; owl:unionOf: (ekg-mm:CapabilityArea, ekg-mm:Pillar))
rdfs:label	hasCapability
rdfs:range	ekg-mm:Capability

13.3.3 hasLevel

Associates a maturity model with a level defined in that model; it is the inverse of levelInModel.

ekg-mm:hasLevel	
https://ekgf.org/ontology/ekgf-maturity-model#hasLevel	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-mm:Model
rdfs:label	hasLevel
rdfs:range	ekg-mm:Level
owl:inverseOf	ekg-mm:levelInModel

13.3.4 hasPillar

Associates a maturity model with one of its pillars; it is the inverse of pillarInModel.

ekg-mm:hasPillar	
https://ekgf.org/ontology/ekgf-maturity-model#hasPillar	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-mm:Model
rdfs:label	hasPillar

ekg-mm:hasPillar	
https://ekgf.org/ontology/ekgf-maturity-model#hasPillar	
RDF predicate	**Value**
rdfs:range	ekg-mm:Pillar
owl:inverseOf	ekg-mm:pillarInModel

13.3.5 inArea

Identifies the capability area containing a maturity-model capability.

ekg-mm:inArea	
https://ekgf.org/ontology/ekgf-maturity-model#inArea	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-mm:Capability
rdfs:label	inArea
rdfs:range	ekg-mm:CapabilityArea

13.3.6 inModel

Relates a capability, capability area, level, or pillar to the maturity model in which it participates. The property is declared transitive.

ekg-mm:inModel	
https://ekgf.org/ontology/ekgf-maturity-model#inModel	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdf:type	owl:TransitiveProperty
rdfs:domain	(rdf:type: owl:Class; owl:unionOf: (ekg-mm:Capability, ekg-mm:CapabilityArea, ekg-mm:Level, ekg-mm:Pillar))
rdfs:label	inModel
rdfs:range	ekg-mm:Model

13.3.7 inPillar

Identifies the pillar containing a capability area.

ekg-mm:inPillar	
https://ekgf.org/ontology/ekgf-maturity-model#inPillar	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-mm:CapabilityArea
rdfs:label	inPillar

ekg-mm:inPillar	
https://ekgf.org/ontology/ekgf-maturity-model#inPillar	
RDF predicate	**Value**
rdfs:range	ekg-mm:Pillar

13.3.8 IRI local name

The "local name" of the given object as it is used in IRIs or URIs

ekg-mm:iriLocalName	
https://ekgf.org/ontology/ekgf-maturity-model#iriLocalName	
RDF predicate	**Value**
rdf:type	owl:AnnotationProperty
rdfs:label	IRI local name

13.3.9 IRI local type name

The "local type name" of the given object as it is used in IRIs or URIs

ekg-mm:iriLocalTypeName	
https://ekgf.org/ontology/ekgf-maturity-model#iriLocalTypeName	
RDF predicate	**Value**
rdf:type	owl:AnnotationProperty
rdfs:label	IRI local type name

13.3.10 levelInModel

Identifies the maturity model defining a level. The functional-property axiom limits each level resource to one model.

ekg-mm:levelInModel	
https://ekgf.org/ontology/ekgf-maturity-model#levelInModel	
RDF predicate	**Value**
rdf:type	owl:FunctionalProperty
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-mm:Level
rdfs:label	levelInModel
rdfs:range	ekg-mm:Model

13.3.11 pillarInModel

Identifies the maturity model containing a pillar. The functional-property axiom limits each pillar resource to one model.

ekg-mm:pillarInModel	
https://ekgf.org/ontology/ekgf-maturity-model#pillarInModel	
RDF predicate	**Value**
rdf:type	owl:FunctionalProperty
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-mm:Pillar
rdfs:label	pillarInModel
rdfs:range	ekg-mm:Model

13.4 Named Individuals

13.4.1 Business Pillar

Instance of: ekg-mm:Pillar

The business dimension of the example EKGF maturity model.

ekg-mm:BusinessPillar	
https://ekgf.org/ontology/ekgf-maturity-model#BusinessPillar	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-mm:Pillar
rdfs:label	Business Pillar
ekg-mm:inModel	ekg-mm:EKGFMaturityModel
ekg-mm:iriLocalName	business

13.4.2 Data Pillar

Instance of: ekg-mm:Pillar

The data dimension of the example EKGF maturity model.

ekg-mm:DataPillar	
https://ekgf.org/ontology/ekgf-maturity-model#DataPillar	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-mm:Pillar
rdfs:label	Data Pillar
ekg-mm:iriLocalName	data

13.4.3 Test EKG/MM

Instance of: ekg-mm:Model

The example EKGF maturity-model instance in this vocabulary, containing five numbered levels and four pillars. This instance does not provide the assessment criteria for those levels.

ekg-mm:EKGFMaturityModel	
https://ekgf.org/ontology/ekgf-maturity-model#EKGFMaturityModel	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-mm:Model
rdfs:label	Test EKG/MM
ekg-mm:hasLevel	ekg-mm:Level1
ekg-mm:hasLevel	ekg-mm:Level2
ekg-mm:hasLevel	ekg-mm:Level3
ekg-mm:hasLevel	ekg-mm:Level4
ekg-mm:hasLevel	ekg-mm:Level5
ekg-mm:hasPillar	ekg-mm:BusinessPillar
ekg-mm:hasPillar	ekg-mm:DataPillar
ekg-mm:hasPillar	ekg-mm:OrganizationPillar
ekg-mm:hasPillar	ekg-mm:TechnologyPillar
ekg-mm:iriLocalName	ekg-mm

13.4.4 Level 1

Instance of: ekg-mm:Level

Numbered level 1 of the example EKGF maturity model. The vocabulary supplies the level identifier without defining its assessment criteria.

ekg-mm:Level1	
https://ekgf.org/ontology/ekgf-maturity-model#Level1	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-mm:Level
rdfs:label	Level 1
ekg-mm:iriLocalName	1

13.4.5 Level 2

Instance of: ekg-mm:Level

Numbered level 2 of the example EKGF maturity model. The vocabulary supplies the level identifier without defining its assessment criteria.

ekg-mm:Level2	
https://ekgf.org/ontology/ekgf-maturity-model#Level2	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-mm:Level
rdfs:label	Level 2
ekg-mm:iriLocalName	2

13.4.6 Level 3

Instance of: ekg-mm:Level

Numbered level 3 of the example EKGF maturity model. The vocabulary supplies the level identifier without defining its assessment criteria.

ekg-mm:Level3	
https://ekgf.org/ontology/ekgf-maturity-model#Level3	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-mm:Level
rdfs:label	Level 3
ekg-mm:iriLocalName	3

13.4.7 Level 4

Instance of: ekg-mm:Level

Numbered level 4 of the example EKGF maturity model. The vocabulary supplies the level identifier without defining its assessment criteria.

ekg-mm:Level4	
https://ekgf.org/ontology/ekgf-maturity-model#Level4	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-mm:Level
rdfs:label	Level 4
ekg-mm:iriLocalName	4

13.4.8 Level 5

Instance of: ekg-mm:Level

Numbered level 5 of the example EKGF maturity model. The vocabulary supplies the level identifier without defining its assessment criteria.

ekg-mm:Level5	
https://ekgf.org/ontology/ekgf-maturity-model#Level5	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-mm:Level
rdfs:label	Level 5
ekg-mm:iriLocalName	5

13.4.9 Organization Pillar

Instance of: ekg-mm:Pillar

The organizational dimension of the example EKGF maturity model.

ekg-mm:OrganizationPillar	
https://ekgf.org/ontology/ekgf-maturity-model#OrganizationPillar	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-mm:Pillar
rdfs:label	Organization Pillar
ekg-mm:iriLocalName	organization

13.4.10 Technology Pillar

Instance of: ekg-mm:Pillar

The technology dimension of the example EKGF maturity model.

ekg-mm:TechnologyPillar	
https://ekgf.org/ontology/ekgf-maturity-model#TechnologyPillar	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-mm:Pillar
rdfs:label	Technology Pillar
ekg-mm:iriLocalName	technology

14 Outcome Ontology

14.1 Overview

Outcome Ontology

Identifier	Value
Ontology IRI / Namespace	`https://ekgf.org/ontology/outcome#`
Prefix	`ekg-outcome`

An Outcome records a shared business result. An OutcomeRelationship records one use case's contribution to that result, with its own UUID URN identity, one `useCase`, one `targetOutcome`, and optional `contributionDescription`. The inverse `use-case:outcomeLink` supports navigation from the use case. Several use cases can contribute to the same outcome with separate descriptions.

The Method's Outcome Relationship provides the source model. OWL declares endpoint types, exact cardinalities, and separation of relationship and endpoint roles. SHACL checks the supplied resource terms, identifiers and text; validate before OWL identity reasoning.

`use-case:hasOutcome` is an unqualified summary that OWL can derive from `use-case:outcomeLink` followed by `targetOutcome`. The summary does not identify a relationship, record its contribution, or establish ownership. Direct summary assertions do not manufacture relationship objects. `hasOutcome` in this namespace and `story:contributesTo` retain their direct-outcome meanings.

Relationships referencing an ancestor's outcome must be stated explicitly. Use-case parentage and reuse create no automatic outcome links, contributions or ownership assignments. Outcome priority concerns the outcome itself. The Method does not define confidence or contribution-priority scales; these properties and an explicit outcome-ownership model remain separate decisions.

All entry and nested validators belong to this ontology's shapes graph, the ontology IRI typed `sh:ShapesGraph`. See the relationship guide for a complete example, query, validation command and snapshot semantics.

14.2 Classes

14.2.1 Outcome

A result, goal, benefit, or theme to which an EKG use case or story contributes, classified by an outcome stereotype and optionally prioritized.

`ekg-outcome:Outcome`	
`https://ekgf.org/ontology/outcome#Outcome`	
RDF predicate	**Value**
`dc:creator`	`http://orcid.org/0000-0003-2309-5068`
`dc:date`	1 April 2020, 18:12:06 UTC
`rdf:type`	`owl:Class`
`rdfs:label`	Outcome

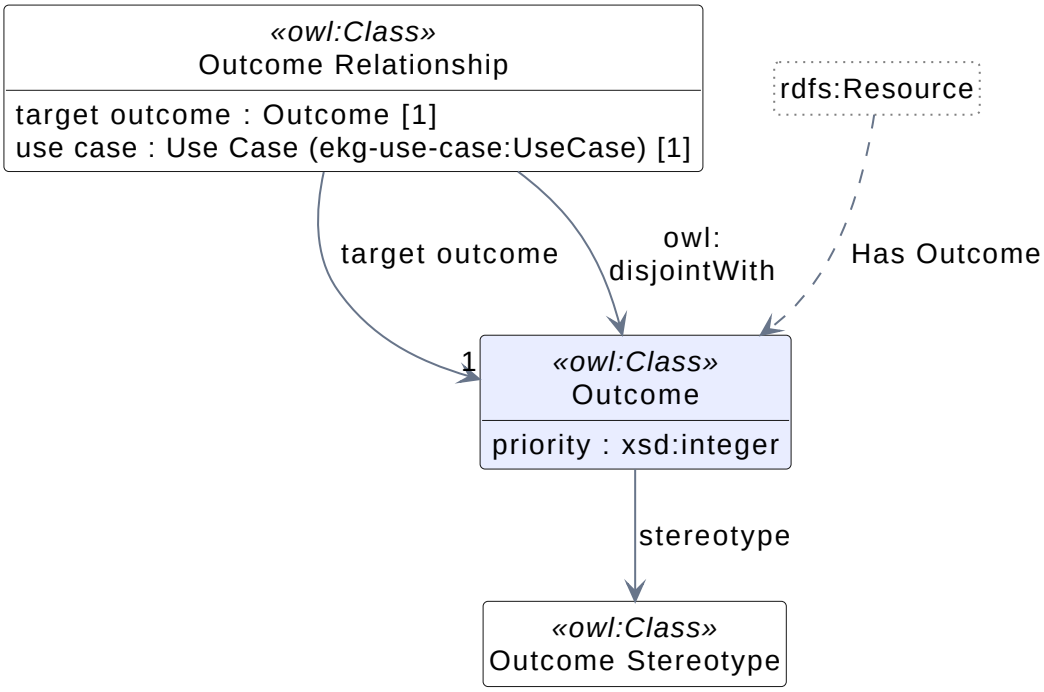


Diagram note

14.2.2 Outcome Relationship

Each relationship connects exactly one UseCase to exactly one Outcome. Use a stable UUID URN such as `urn:uuid:fa48a552-6f45-4f01-872e-0146622295d6` for its identity; generate a fresh random UUIDv4 when creating a relationship and keep it when labels or descriptions change. The validation profile checks the hyphenated UUID URN syntax. It does not prove random generation or global uniqueness, and does not constrain the UUID version.

The relationship is disjoint from UseCase and Outcome. It is not the outcome itself. Its endpoints are functional, but neither endpoint identifies the relationship: sharing a source, a target, or both never merges relationship resources. Multiple relationships for the same pair are permitted and can carry separate contribution descriptions. No pair-based identity key is set.

The relationship describes contribution. It does not make the source use case the owner of the outcome. Parent and child use cases can explicitly refer to the same shared outcome; this does not copy that outcome or automatically create relationships for other descendants.

`ekg-outcome:OutcomeRelationship`	
`https://ekgf.org/ontology/outcome#OutcomeRelationship`	
RDF predicate	**Value**
`rdf:type`	`owl:Class`
`rdfs:label`	Outcome Relationship
`rdfs:subClassOf`	(`rdf:type`: `owl:Restriction`; `owl:cardinality`: 1; `owl:onProperty`: ekg-outcome:targetOutcome)
`rdfs:subClassOf`	(`rdf:type`: `owl:Restriction`; `owl:cardinality`: 1; `owl:onProperty`: ekg-outcome:useCase)
`owl:disjointWith`	ekg-outcome:Outcome

ekg-outcome:OutcomeRelationship	
https://ekgf.org/ontology/outcome#OutcomeRelationship	
RDF predicate	**Value**
owl:disjointWith	ekg-use-case:UseCase

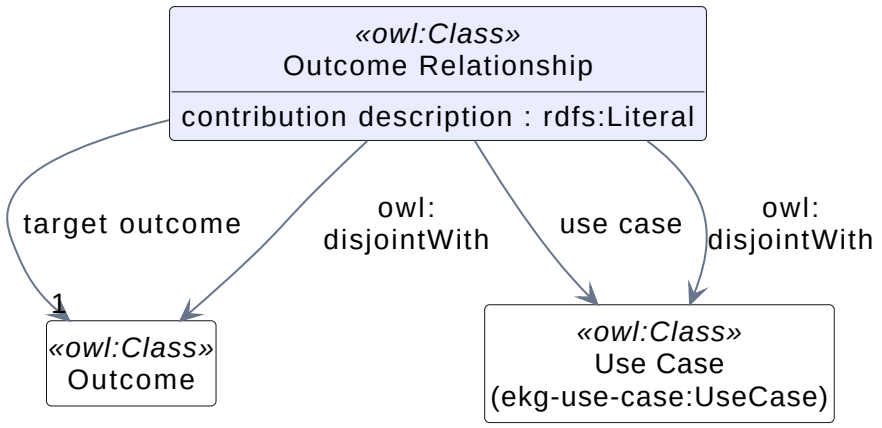


Diagram note

14.2.3 Outcome Stereotype

The kinds of outcome.

ekg-outcome:OutcomeStereoType	
https://ekgf.org/ontology/outcome#OutcomeStereoType	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Outcome Stereotype

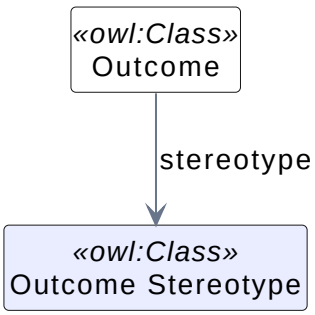


Diagram note

14.3 Properties

14.3.1 contribution description

Optional text describing the source use case's contribution to the target outcome. Use plain strings or language-tagged literals. Multiple translations are supported. Every supplied value must contain a non-whitespace character; numbers, resource IRIs, empty strings and whitespace-only text fail validation.

This description belongs to the relationship. Different use cases can share one outcome while explaining different contributions, and an absent description does not invalidate an otherwise complete relationship. The text does not assert that an outcome has been achieved, assign ownership or execute metrics.

ekg-outcome:contributionDescription	
https://ekgf.org/ontology/outcome#contributionDescription	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-outcome:OutcomeRelationship
rdfs:label	contribution description
rdfs:range	rdfs:Literal

14.3.2 Has Outcome

Relates a resource, such as a use case or story, to an outcome it contributes to. No domain restriction is declared.

ekg-outcome:hasOutcome	
https://ekgf.org/ontology/outcome#hasOutcome	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:label	Has Outcome
rdfs:range	ekg-outcome:Outcome

14.3.3 priority

The priority of the outcome, as an integer.

ekg-outcome:priority	
https://ekgf.org/ontology/outcome#priority	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-outcome:Outcome
rdfs:label	priority
rdfs:range	xsd:integer

14.3.4 stereotype

The kind of outcome: theme, goal, objective or benefit.

ekg-outcome:stereotype	
https://ekgf.org/ontology/outcome#stereotype	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-outcome:Outcome
rdfs:label	stereotype
rdfs:range	ekg-outcome:OutcomeStereoType

14.3.5 target outcome

Identifies exactly one Outcome for this relationship. Many relationships can target that same Outcome, including contributions from different use cases. The outcome's definition, classification, lifecycle and outcome-level priority remain on the shared Outcome; the contribution description stays on each OutcomeRelationship.

This property is functional, not inverse-functional. It does not limit an outcome to one contributing use case or equate relationships sharing a target. The validator requires one supplied Outcome IRI and rejects multiple target terms before OWL functionality could equate those names.

ekg-outcome:targetOutcome	
https://ekgf.org/ontology/outcome#targetOutcome	
RDF predicate	**Value**
rdf:type	owl:FunctionalProperty
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-outcome:OutcomeRelationship
rdfs:label	target outcome
rdfs:range	ekg-outcome:Outcome

14.3.6 updated at

When the outcome was last changed.

ekg-outcome:updatedAt	
https://ekgf.org/ontology/outcome#updatedAt	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-outcome:Outcome
rdfs:label	updated at
rdfs:range	xsd:dateTime

14.3.7 use case

Identifies the single use case whose contribution this OutcomeRelationship describes. `use-case:outcomeLink` is its inverse, declared in the Use Case Ontology. Load both ontology modules for their complete OWL relationships. A use case can participate in several relationship objects.

The validator counts `useCase` objects and inverse `use-case:outcomeLink` subjects together without requiring inverse inference. Repeating the same source in both directions counts once; two different source terms fail. Functional OWL properties can equate endpoint names, so validate the supplied terms before applying OWL identity reasoning.

ekg-outcome:useCase	
https://ekgf.org/ontology/outcome#useCase	
RDF predicate	**Value**
rdf:type	owl:FunctionalProperty
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-outcome:OutcomeRelationship
rdfs:label	use case
rdfs:range	ekg-use-case:UseCase

14.4 Named Individuals

14.4.1 Benefit

Instance of: ekg-outcome:OutcomeStereoType

The outcome stereotype used to classify an outcome as a benefit.

ekg-outcome:OutcomeStereoTypeBenefit	
https://ekgf.org/ontology/outcome#OutcomeStereoTypeBenefit	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-outcome:OutcomeStereoType
rdfs:label	Benefit

14.4.2 Goal

Instance of: ekg-outcome:OutcomeStereoType

The outcome stereotype used to classify an outcome as a goal.

ekg-outcome:OutcomeStereoTypeGoal	
https://ekgf.org/ontology/outcome#OutcomeStereoTypeGoal	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-outcome:OutcomeStereoType
rdfs:label	Doel
rdfs:label	Goal

14.4.3 Objective

Instance of: ekg-outcome:OutcomeStereoType

The outcome stereotype used to classify an outcome as an objective.

ekg-outcome:OutcomeStereoTypeObjective	
https://ekgf.org/ontology/outcome#OutcomeStereoTypeObjective	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-outcome:OutcomeStereoType
rdfs:label	Doelstelling
rdfs:label	Objective

14.4.4 Theme

Instance of: ekg-outcome:OutcomeStereoType

The outcome stereotype used to classify an outcome as a guiding theme.

ekg-outcome:OutcomeStereoTypeTheme	
https://ekgf.org/ontology/outcome#OutcomeStereoTypeTheme	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-outcome:OutcomeStereoType
rdfs:label	Thema
rdfs:label	Theme

14.4.5 Accountability

Instance of: ekg-outcome:Outcome

A theme outcome for recording goals and benefits concerning accountability, linked to the corresponding EKG principle for further explanation.

ekg-outcome:ThemeAccountability	
https://ekgf.org/ontology/outcome#ThemeAccountability	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-outcome:Outcome
rdfs:label	Accountability
rdfs:label	Verantwoordelijkheid
rdfs:seeAlso	https://principles.ekgf.org/theme/accountability/
ekg-outcome:stereotype	ekg-outcome:OutcomeStereoTypeTheme

14.4.6 Composable Business

Instance of: ekg-outcome:Outcome

A theme outcome for recording goals and benefits concerning composable business, linked to the corresponding EKG principle for further explanation.

ekg-outcome:ThemeComposableBusiness	
https://ekgf.org/ontology/outcome#ThemeComposableBusiness	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-outcome:Outcome
rdfs:label	Composable Business
rdfs:label	Samenstelbaar Bedrijf
rdfs:seeAlso	https://principles.ekgf.org/theme/composable-business/
ekg-outcome:stereotype	ekg-outcome:OutcomeStereoTypeTheme

14.4.7 Contextual Truth

Instance of: ekg-outcome:Outcome

A theme outcome for recording goals and benefits concerning contextual truth, linked to the corresponding EKG principle for further explanation.

ekg-outcome:ThemeContextualTruth	
https://ekgf.org/ontology/outcome#ThemeContextualTruth	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-outcome:Outcome
rdfs:label	Contextual Truth
rdfs:label	Contextuele Waarheid
rdfs:seeAlso	https://principles.ekgf.org/theme/contextual-truth/
ekg-outcome:stereotype	ekg-outcome:OutcomeStereoTypeTheme

14.4.8 Digital Assets

Instance of: ekg-outcome:Outcome

A theme outcome for recording goals and benefits concerning digital assets, linked to the corresponding EKG principle for further explanation.

ekg-outcome:ThemeDigitalAssets	
https://ekgf.org/ontology/outcome#ThemeDigitalAssets	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-outcome:Outcome
rdfs:label	Digital Assets

ekg-outcome:ThemeDigitalAssets	
https://ekgf.org/ontology/outcome#ThemeDigitalAssets	
RDF predicate	**Value**
rdfs:label	Digitale Middelen
rdfs:seeAlso	https://principles.ekgf.org/theme/digital-assets/
ekg-outcome:stereotype	ekg-outcome:OutcomeStereoTypeTheme

14.4.9 Environmental Sustainability

Instance of: ekg-outcome:Outcome

A theme outcome for recording goals and benefits concerning environmental sustainability, linked to the corresponding EKG principle for further explanation.

ekg-outcome:ThemeEnvironmentalSustainability	
https://ekgf.org/ontology/outcome#ThemeEnvironmentalSustainability	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-outcome:Outcome
rdfs:label	Environmental Sustainability
rdfs:label	Milieuduurzaamheid
rdfs:seeAlso	https://principles.ekgf.org/theme/environmental-sustainability/
ekg-outcome:stereotype	ekg-outcome:OutcomeStereoTypeTheme

14.4.10 Fairness

Instance of: ekg-outcome:Outcome

A theme outcome for recording goals and benefits concerning fairness, linked to the corresponding EKG principle for further explanation.

ekg-outcome:ThemeFairness	
https://ekgf.org/ontology/outcome#ThemeFairness	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-outcome:Outcome
rdfs:label	Eerlijkheid
rdfs:label	Fairness
rdfs:seeAlso	https://principles.ekgf.org/theme/fairness/
ekg-outcome:stereotype	ekg-outcome:OutcomeStereoTypeTheme

14.4.11 Manage Profitability

Instance of: ekg-outcome:Outcome

A theme outcome for recording goals and benefits concerning profitability management, linked to the corresponding EKG principle for further explanation.

ekg-outcome:ThemeManageProfitability	
https://ekgf.org/ontology/outcome#ThemeManageProfitability	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-outcome:Outcome
rdfs:label	Beheer Winstgevendheid
rdfs:label	Manage Profitability
rdfs:seeAlso	https://principles.ekgf.org/theme/manage-profitability/
ekg-outcome:stereotype	ekg-outcome:OutcomeStereoTypeTheme

14.4.12 Manage Risk & Compliance

Instance of: ekg-outcome:Outcome

A theme outcome for recording goals and benefits concerning risk and compliance management, linked to the corresponding EKG principle for further explanation.

ekg-outcome:ThemeManageRiskAndCompliance	
https://ekgf.org/ontology/outcome#ThemeManageRiskAndCompliance	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-outcome:Outcome
rdfs:label	Beheer Risico & Compliance
rdfs:label	Manage Risk & Compliance
rdfs:seeAlso	https://principles.ekgf.org/theme/manage-risk-and-compliance/
ekg-outcome:stereotype	ekg-outcome:OutcomeStereoTypeTheme

14.4.13 Openness

Instance of: ekg-outcome:Outcome

A theme outcome for recording goals and benefits concerning openness, linked to the corresponding EKG principle for further explanation.

ekg-outcome:ThemeOpenness	
https://ekgf.org/ontology/outcome#ThemeOpenness	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-outcome:Outcome
rdfs:label	Openheid

ekg-outcome:ThemeOpenness	
https://ekgf.org/ontology/outcome#ThemeOpenness	
RDF predicate	**Value**
rdfs:label	Openness
rdfs:seeAlso	https://principles.ekgf.org/theme/openness/
ekg-outcome:stereotype	ekg-outcome:OutcomeStereoTypeTheme

14.4.14 Power of Abstraction

Instance of: ekg-outcome:Outcome

A theme outcome for recording goals and benefits concerning the use of abstraction, linked to the corresponding EKG principle for further explanation.

ekg-outcome:ThemePowerOfAbstraction	
https://ekgf.org/ontology/outcome#ThemePowerOfAbstraction	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-outcome:Outcome
rdfs:label	Kracht van Abstractie
rdfs:label	Power of Abstraction
rdfs:seeAlso	https://principles.ekgf.org/theme/power-of-abstraction/
ekg-outcome:stereotype	ekg-outcome:OutcomeStereoTypeTheme

14.4.15 Resilience

Instance of: ekg-outcome:Outcome

A theme outcome for recording goals and benefits concerning resilience, linked to the corresponding EKG principle for further explanation.

ekg-outcome:ThemeResilience	
https://ekgf.org/ontology/outcome#ThemeResilience	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-outcome:Outcome
rdfs:label	Resilience
rdfs:label	Weerbaarheid
rdfs:seeAlso	https://principles.ekgf.org/theme/resilience/
ekg-outcome:stereotype	ekg-outcome:OutcomeStereoTypeTheme

14.4.16 Reusability

Instance of: ekg-outcome:Outcome

A theme outcome for recording goals and benefits concerning reusability, linked to the corresponding EKG principle for further explanation.

ekg-outcome:ThemeReusability	
https://ekgf.org/ontology/outcome#ThemeReusability	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-outcome:Outcome
rdfs:label	Herbruikbaarheid
rdfs:label	Reusability
rdfs:seeAlso	https://principles.ekgf.org/theme/reusability/
ekg-outcome:stereotype	ekg-outcome:OutcomeStereoTypeTheme

14.4.17 Sustainability

Instance of: ekg-outcome:Outcome

A theme outcome for recording goals and benefits concerning sustainability, linked to the corresponding EKG principle for further explanation.

ekg-outcome:ThemeSustainability	
https://ekgf.org/ontology/outcome#ThemeSustainability	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-outcome:Outcome
rdfs:label	Duurzaamheid
rdfs:label	Sustainability
rdfs:seeAlso	https://principles.ekgf.org/theme/sustainability/
ekg-outcome:stereotype	ekg-outcome:OutcomeStereoTypeTheme

14.4.18 Transparency

Instance of: ekg-outcome:Outcome

A theme outcome for recording goals and benefits concerning transparency, linked to the corresponding EKG principle for further explanation.

ekg-outcome:ThemeTransparency	
https://ekgf.org/ontology/outcome#ThemeTransparency	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-outcome:Outcome
rdfs:label	Transparantie

ekg-outcome:ThemeTransparency	
https://ekgf.org/ontology/outcome#ThemeTransparency	
RDF predicate	**Value**
rdfs:label	Transparency
rdfs:seeAlso	https://principles.ekgf.org/theme/transparency/
ekg-outcome:stereotype	ekg-outcome:OutcomeStereoTypeTheme

14.4.19 Unknown Outcome

Instance of: ekg-outcome:Outcome

The default outcome. Think about what your use case or story accomplishes.

ekg-outcome:UnknownOutcome	
https://ekgf.org/ontology/outcome#UnknownOutcome	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-outcome:Outcome
rdfs:label	Onbekende uitkomst
rdfs:label	Unknown Outcome
ekg-outcome:stereotype	ekg-outcome:OutcomeStereoTypeTheme

14.5 Shapes

14.5.1 Outcome Relationship Shape

Targets OutcomeRelationship instances and subclasses, subjects of `useCase`, `targetOutcome` and `contributionDescription`, and objects of `use-case:outcomeLink`. Predicate targets prevent incomplete records from escaping validation merely by omitting an explicit relationship type.

Requires a UUID URN identifier, exactly one source UseCase IRI counted in both directions, exactly one target Outcome IRI, and nonblank text for every supplied description. Relationship resources cannot also be UseCases or Outcomes. With no inference, endpoint classes must be supplied; RDFS inference with both ontology modules can derive endpoint types from property domains/ranges.

Validate one coherent version context before OWL identity reasoning. Different endpoint-version IRIs count as different supplied terms in a merged graph. This shape validates relationship instances; it does not require every UseCase to have one, determine business ownership, infer inheritance or assess whether the contribution description is accurate.

ekg-outcome:OutcomeRelationshipShape	
https://ekgf.org/ontology/outcome#OutcomeRelationshipShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Outcome Relationship Shape
sh:nodeKind	sh:IRI

ekg-outcome:OutcomeRelationshipShape	
https://ekgf.org/ontology/outcome#OutcomeRelationshipShape	
RDF predicate	**Value**
sh:not	(sh:class: ekg-outcome:Outcome)
sh:not	(sh:class: ekg-use-case:UseCase)
sh:pattern	^urn:uuid:[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12}$
sh:property	(sh:message: A supplied contribution description must be nonblank plain or language-tagged text.; sh:or: ((sh:datatype: xsd:string), (sh:datatype: rdf:langString)); sh:path: ekg-outcome:contributionDescription; sh:pattern: \S)
sh:property	(sh:class: ekg-use-case:UseCase; sh:maxCount: 1; sh:message: An outcome relationship must identify exactly one UseCase IRI, counting useCase and inverse outcomeLink assertions together.; sh:minCount: 1; sh:nodeKind: sh:IRI; sh:path: (sh:alternativePath: (ekg-outcome:useCase, (sh:inversePath: ekg-use-case:outcomeLink))))
sh:property	(sh:class: ekg-outcome:Outcome; sh:maxCount: 1; sh:message: An outcome relationship must identify exactly one target Outcome IRI.; sh:minCount: 1; sh:nodeKind: sh:IRI; sh:path: ekg-outcome:targetOutcome)
sh:targetClass	ekg-outcome:OutcomeRelationship
sh:targetObjectsOf	ekg-use-case:outcomeLink
sh:targetSubjectsOf	ekg-outcome:contributionDescription
sh:targetSubjectsOf	ekg-outcome:targetOutcome
sh:targetSubjectsOf	ekg-outcome:useCase

15 Persona Ontology

15.1 Overview

Persona Ontology

Identifier	Value
Ontology IRI / Namespace	`https://ekgf.org/ontology/persona#`
Prefix	`ekg-persona`

A vocabulary for personas participating in EKG stories, their accountabilities, and sub-persona relationships. Persona identities are distinct from individual people.

15.2 Classes

15.2.1 Accountability

A responsibility or expected area of ownership assigned to a persona.

`ekg-persona:Accountability`	
`https://ekgf.org/ontology/persona#Accountability`	
RDF predicate	**Value**
`dc:date`	18 August 2023, 17:12:00 UTC
`rdf:type`	`owl:Class`
`rdfs:label`	Accountability

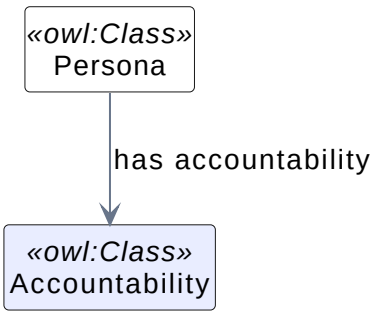


Diagram note

15.2.2 Persona

A role-oriented description of the actor who participates in a story or carries an accountability in the EKG method. It is not an identity record for an individual person.

`ekg-persona:Persona`	
`https://ekgf.org/ontology/persona#Persona`	
RDF predicate	**Value**
`dc:creator`	`http://orcid.org/0000-0003-2309-5068`

ekg-persona:Persona	
https://ekgf.org/ontology/persona#Persona	
RDF predicate	**Value**
dc:date	1 April 2020, 18:10:23 UTC
rdf:type	owl:Class
rdfs:label	Persona

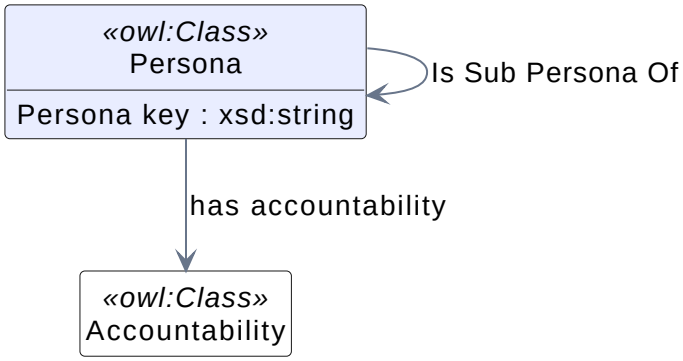


Diagram note

15.3 Properties

15.3.1 has accountability

Relates a persona to an accountability assigned to it.

ekg-persona:accountability	
https://ekgf.org/ontology/persona#accountability	
RDF predicate	**Value**
dc:date	18 August 2023, 17:12:22 UTC
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-persona:Persona
rdfs:label	has accountability
rdfs:range	ekg-persona:Accountability

15.3.2 Is Sub Persona Of

Relates a more specific persona to a broader persona. Its irreflexive axiom excludes a persona being its own sub-persona.

ekg-persona:isSubPersonaOf	
https://ekgf.org/ontology/persona#isSubPersonaOf	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068

ekg-persona:isSubPersonaOf	
https://ekgf.org/ontology/persona#isSubPersonaOf	
RDF predicate	**Value**
dc:date	20 June 2020, 14:34:46 UTC
rdf:type	owl:IrreflexiveProperty
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-persona:Persona
rdfs:label	Is Sub Persona Of
rdfs:range	ekg-persona:Persona

15.3.3 Persona key

Records the string key used to identify a persona within an application or configuration. Functionality limits a persona to one value; it does not assert global uniqueness.

ekg-persona:key	
https://ekgf.org/ontology/persona#key	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	1 May 2020, 15:41:28 UTC
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-persona:Persona
rdfs:label	Persona key
rdfs:range	xsd:string

16 Raw Data Ontology

16.1 Overview

Raw Data Ontology

Identifier	Value
Ontology IRI / Namespace	`https://ekgf.org/ontology/raw#`
Prefix	`ekg-raw`

The Raw Data Ontology is used by a "capture step" in an EKG DataOps pipeline to capture any data from any data source as RDF triples without mapping it to higher level ontologies. It is assumed that subsequent steps in a given EKG DataOps pipeline will transform the raw data to match with target ontologies.

16.2 Classes

16.2.1 Column

A physical field position in exactly one captured raw-data view, linked through isColumnInView or inverse hasColumn. Its identity is independent of its mapped predicate; source metadata and optional statistics describe the column.

`ekg-raw:Column`	
`https://ekgf.org/ontology/raw#Column`	
RDF predicate	**Value**
`dc:creator`	`http://orcid.org/0000-0003-2309-5068`
`dc:date`	13 June 2020, 09:09:19 UTC
`rdf:type`	`owl:Class`
`rdfs:label`	Column
`rdfs:subClassOf`	(`rdf:type`: `owl:Restriction`; `owl:minCardinality`: 1; `owl:onProperty`: ekg-raw:isColumnInView)

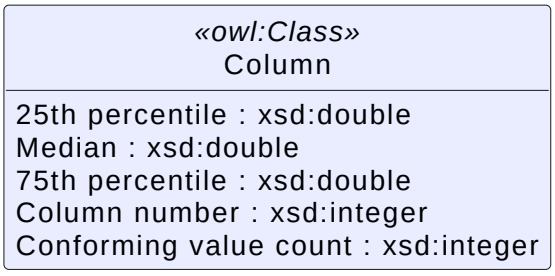


Diagram note

16.2.2 CSV file

A raw-data file source in comma-separated values format (RFC 4180). Every field is captured as text; the capture step does not infer datatypes.

ekg-raw:CsvFile	
https://ekgf.org/ontology/raw#CsvFile	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	CSV file
rdfs:subClassOf	ekg-raw:File

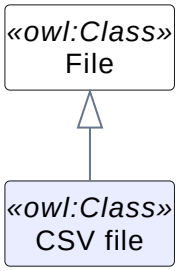


Diagram note

16.2.3 File

A file used as a raw-data source during the capture step of a DataOps pipeline.

ekg-raw:File	
https://ekgf.org/ontology/raw#File	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	13 June 2020, 09:26:36 UTC
rdf:type	owl:Class
rdfs:label	File
rdfs:subClassOf	ekg-raw:Source

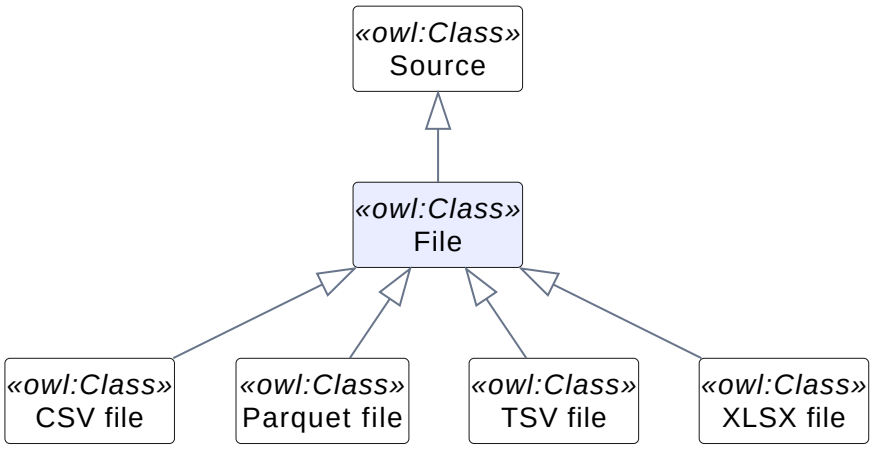


Diagram note

16.2.4 Null value

A captured value occurrence whose source explicitly reports null. It has no literalValue. Missing content metadata and empty strings do not imply this classification; the source reader must identify the null according to the source format or an explicit capture policy.

ekg-raw:NullValue	
https://ekgf.org/ontology/raw#NullValue	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Null value
rdfs:subClassOf	ekg-raw:Value
rdfs:subClassOf	(rdf:type: owl:Restriction; owl:maxCardinality: 0; owl:onProperty: ekg-raw:literalValue)

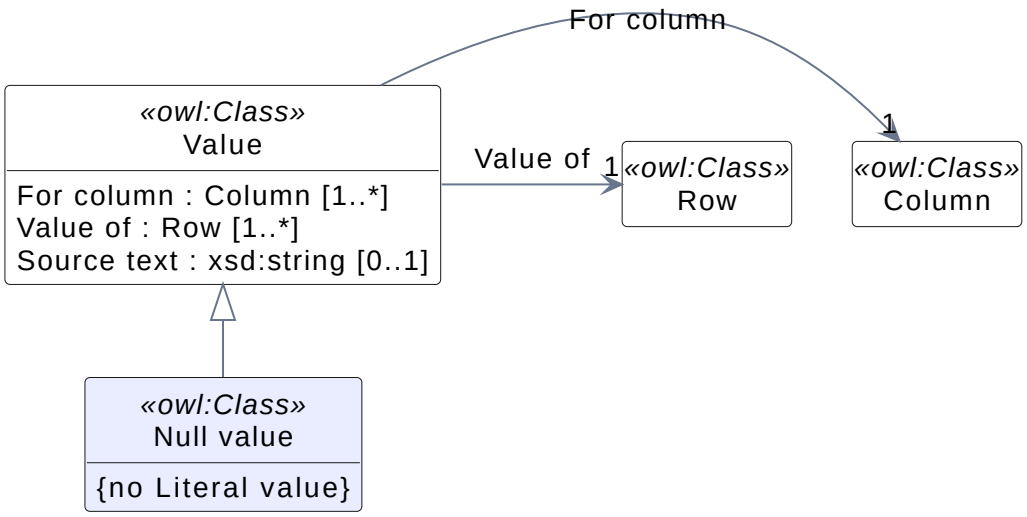


Diagram note

16.2.5 Parquet file

A raw-data file source in the Apache Parquet columnar format. One file is one view whose columns are the schema's top-level fields; values keep the literal datatypes that the file's physical and logical types define.

ekg-raw:ParquetFile	
https://ekgf.org/ontology/raw#ParquetFile	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Parquet file
rdfs:subClassOf	ekg-raw:File

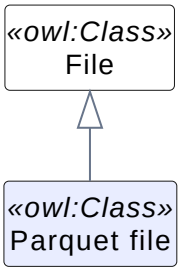


Diagram note

16.2.6 Row

A record encountered in exactly one raw-data view, retaining its source identifier, sequence position, and captured value occurrences. Each occurrence identifies this row through valueOf and its physical source column through forColumn.

ekg-raw:Row	
https://ekgf.org/ontology/raw#Row	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	13 June 2020, 09:09:08 UTC
rdf:type	owl:Class
rdfs:label	Row
rdfs:subClassOf	(rdf:type: owl:Restriction; owl:minCardinality: 1; owl:onProperty: ekg-raw:isRowInView)

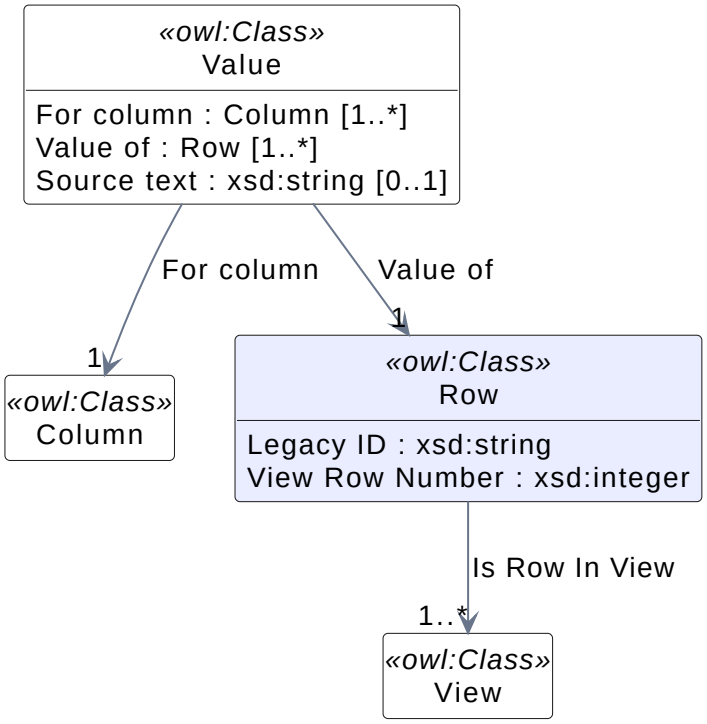


Diagram note

16.2.7 Source

An input data source captured as RDF before transformation into higher-level domain ontologies. A source may expose several views.

ekg-raw:Source	
https://ekgf.org/ontology/raw#Source	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	13 June 2020, 09:26:09 UTC
rdf:type	owl:Class
rdfs:label	Source

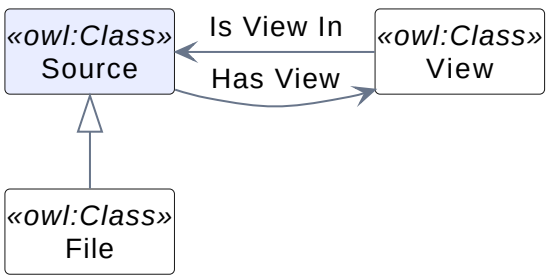


Diagram note

16.2.8 TSV file

A raw-data file source in tab-separated values format. Every field is captured as text; the capture step does not infer datatypes.

ekg-raw:TsvFile	
https://ekgf.org/ontology/raw#TsvFile	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	TSV file
rdfs:subClassOf	ekg-raw:File

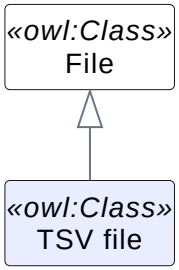


Diagram note

16.2.9 Value

`Value` identifies an occurrence of source data. Its `literalValue` contains the source-supplied content as an RDF literal, retaining the available datatype or language. Text stays text when the source supplies no type; the capture step must not guess that a string such as `"00123"` denotes a number. Domain-specific interpretation belongs to later transformation steps.

Use `sourceText` when the reader exposes an original textual representation. For example, literal content `7` and source text `"0007"` preserve both the source-supplied numeric value and its textual spelling. Do not fabricate source text by formatting a parsed value. This field records reader-exposed text, not the source file's bytes, quoting, or complete spreadsheet formatting.

An empty string, whitespace, zero, and false are values. Explicit source nulls use `NullValue` and have no `literalValue`. A null may retain its source marker in `sourceText`. Missing payload metadata is invalid, not an implicit null. No string, including `"NULL"`, becomes null without the source format or an explicit capture policy establishing that interpretation. An omitted source field is not automatically an explicit null occurrence.

`ValueContentShape`, in this ontology's shapes graph, checks this contract. OWL models the alternatives and cardinalities; it does not supply missing content or classify an incomplete occurrence as null.

Migration and scope

Existing `Value` resources need their captured content or an explicit `NullValue` classification. Preserve their identities and references. Recover content from the original source or the capture application's documented representation; do not infer content from resource IRIs or reinterpret a mapped row property as an occurrence without establishing the correspondence.

Each occurrence also requires one `valueOf` row and one `forColumn` column. Rows identify their one view through `isRowInView`; columns identify theirs through `isColumnInView` or inverse `hasColumn`. The

two views must match. `ValueLocationShape`, `RowLocationShape`, and `ColumnLocationShape` validate these relationships and belong to the same shapes graph.

Recover column links from the source's physical structure or documented capture metadata. Do not join occurrences to columns by mapped predicate alone: different columns can share a mapping, including columns in different views. Validate the supplied capture context before OWL identity reasoning; functional relationships can equate conflicting owner IRIs rather than rejecting them.

Original headers and view names, view-to-source ownership validation, coordinate constraints, mapping-property semantics, and source-key identity remain separate modeling work.

See the raw-data guide for executable examples, validation, and the remaining work.

ekg-raw:Value	
https://ekgf.org/ontology/raw#Value	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	13 June 2020, 09:14:07 UTC
rdf:type	owl:Class
rdfs:label	Value
rdfs:subClassOf	(rdf:type: owl:Class; owl:unionOf: (ekg-raw:NullValue, (rdf:type: owl:Restriction; owl:minCardinality: 1; owl:onProperty: ekg-raw:literalValue)))
rdfs:subClassOf	(rdf:type: owl:Restriction; owl:minCardinality: 1; owl:onProperty: ekg-raw:forColumn)
rdfs:subClassOf	(rdf:type: owl:Restriction; owl:minCardinality: 1; owl:onProperty: ekg-raw:valueOf)

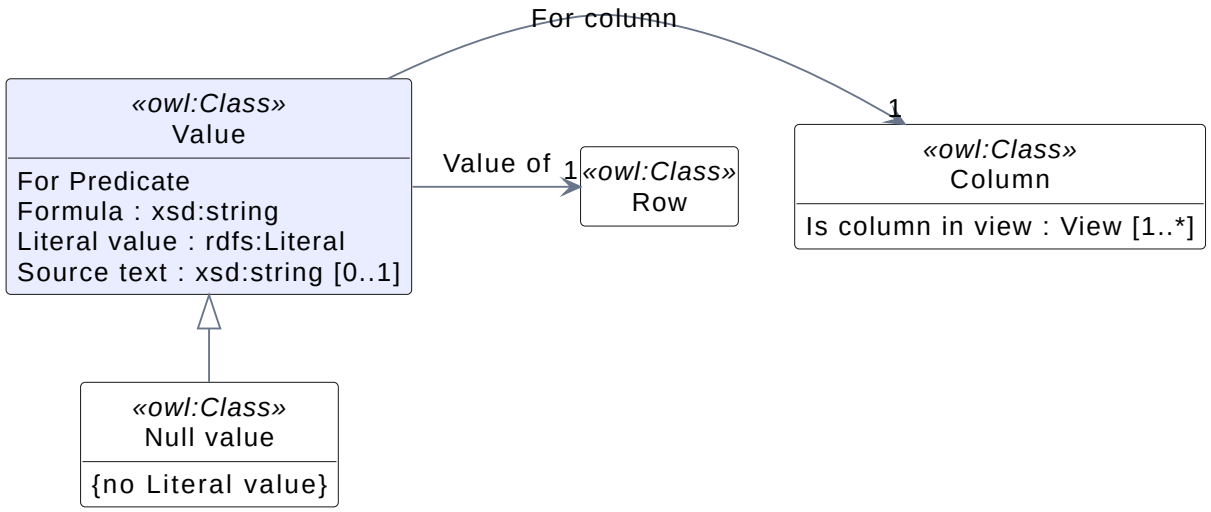


Diagram note

16.2.10 View

A tabular or record-oriented view within a captured source, organizing the rows and columns exposed by that source.

ekg-raw:View	
https://ekgf.org/ontology/raw#View	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	13 June 2020, 09:25:24 UTC
rdf:type	owl:Class
rdfs:label	View

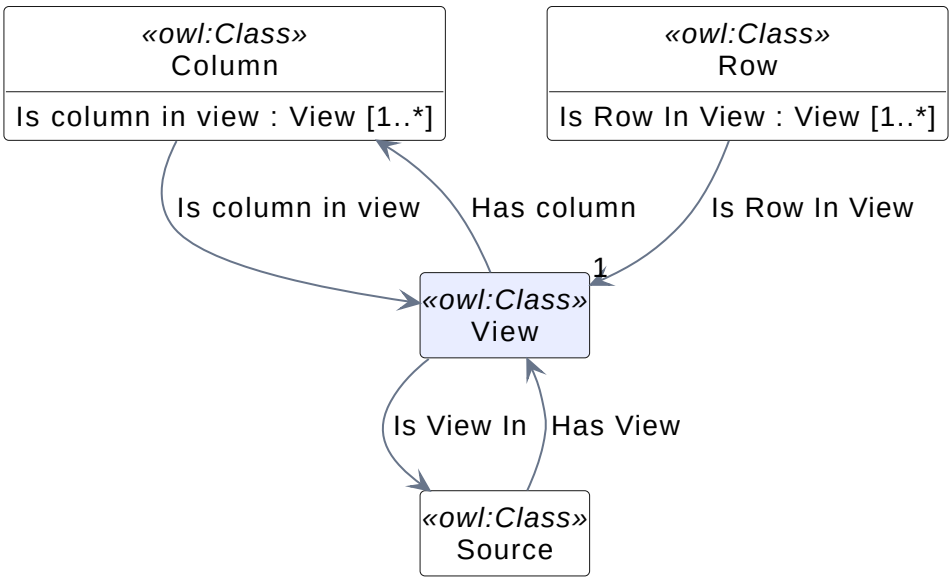


Diagram note

16.2.11 XLSX file

A raw-data file source in the Excel XLSX workbook format.

ekg-raw:XlsxFile	
https://ekgf.org/ontology/raw#XlsxFile	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	13 June 2020, 09:26:15 UTC
rdf:type	owl:Class
rdfs:label	XLSX file
rdfs:subClassOf	ekg-raw:File

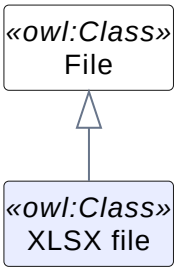


Diagram note

16.3 Properties

16.3.1 25th percentile

The 25th percentile of the numeric values of occurrences conforming to a numeric inferred datatype, interpolating linearly between the closest ranks (Hyndman and Fan type 7).

ekg-raw:25pct	
https://ekgf.org/ontology/raw#25pct	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-raw:Column
rdfs:label	25th percentile
rdfs:range	xsd:double

16.3.2 Median

The 50th percentile (median) of the numeric values of occurrences conforming to a numeric inferred datatype, interpolating linearly between the closest ranks (Hyndman and Fan type 7).

ekg-raw:50pct	
https://ekgf.org/ontology/raw#50pct	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-raw:Column
rdfs:label	Median
rdfs:range	xsd:double

16.3.3 75th percentile

The 75th percentile of the numeric values of occurrences conforming to a numeric inferred datatype, interpolating linearly between the closest ranks (Hyndman and Fan type 7).

ekg-raw:75pct	
https://ekgf.org/ontology/raw#75pct	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-raw:Column
rdfs:label	75th percentile
rdfs:range	xsd:double

16.3.4 Column number

1-based column index

ekg-raw:columnNumber	
https://ekgf.org/ontology/raw#columnNumber	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	13 June 2020, 09:31:29 UTC
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-raw:Column
rdfs:label	Column number
rdfs:range	xsd:integer

16.3.5 Conforming value count

The number of counted occurrences of this column conforming to its inferred datatype. Together with count it gives the share of the column that a later step can cast to that datatype, and identifies how many occurrences need attention.

ekg-raw:conformingCount	
https://ekgf.org/ontology/raw#conformingCount	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-raw:Column
rdfs:label	Conforming value count
rdfs:range	xsd:integer

16.3.6 Value count

The number of captured value occurrences for this column (raw:forColumn) that are not NullValue. Header cells are not occurrences and are not counted.

ekg-raw:count	
https://ekgf.org/ontology/raw#count	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-raw:Column
rdfs:label	Value count
rdfs:range	xsd:integer

16.3.7 File name

The name of the file as it was passed to the capture step

ekg-raw:fileName	
https://ekgf.org/ontology/raw#fileName	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	13 June 2020, 09:26:52 UTC
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-raw:File
rdfs:label	File name
rdfs:range	xsd:string

16.3.8 For column

Identifies the physical source column of a captured value occurrence. Each occurrence has exactly one column, belonging to the same view as its row. A shared predicate mapping does not establish column identity.

ekg-raw:forColumn	
https://ekgf.org/ontology/raw#forColumn	
RDF predicate	**Value**
rdf:type	owl:FunctionalProperty
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-raw:Value
rdfs:label	For column
rdfs:range	ekg-raw:Column

16.3.9 For Predicate

The predicate for which the given value was encountered

ekg-raw:forPredicate	
https://ekgf.org/ontology/raw#forPredicate	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	13 June 2020, 09:14:28 UTC
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-raw:Value
rdfs:label	For Predicate

16.3.10 Formula

The spreadsheet formula that computes a captured value occurrence, verbatim in the source's A1 notation and without the leading '='. The value's literalValue is the result the source cached; an occurrence without a cached result is a NullValue that still records its formula.

ekg-raw:formula	
https://ekgf.org/ontology/raw#formula	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-raw:Value
rdfs:label	Formula
rdfs:range	xsd:string

16.3.11 Formula (R1C1)

A spreadsheet formula in R1C1 notation, with references relative to the cell holding it. On a Value it is that occurrence's formula. On a Column it is the single relative formula shared by every formula occurrence in the column, a derivation rule for the column that later steps can translate into SPARQL or SHACL rules.

ekg-raw:formulaR1C1	
https://ekgf.org/ontology/raw#formulaR1C1	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	(rdf:type: owl:Class; owl:unionOf: (ekg-raw:Value, ekg-raw:Column))
rdfs:label	Formula (R1C1)
rdfs:range	xsd:string

16.3.12 Most frequent value count

The number of counted occurrences whose literalValue is the column's top value.

ekg-raw:freq	
https://ekgf.org/ontology/raw#freq	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-raw:Column
rdfs:label	Most frequent value count
rdfs:range	xsd:integer

16.3.13 Git blob

The git blob object id, as lowercase hexadecimal, of the captured bytes. It is computed from the content alone, so it is present whether or not the file is in a repository, and equals the id git would assign to the same bytes.

ekg-raw:gitBlob	
https://ekgf.org/ontology/raw#gitBlob	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-raw:File
rdfs:label	Git blob
rdfs:range	xsd:string

16.3.14 Git commit

The object id, as lowercase hexadecimal, of the HEAD commit of the repository whose work tree contained the file at capture time. Together with gitPath it locates the file in history; gitModified states whether the captured bytes are the bytes of that commit.

ekg-raw:gitCommit	
https://ekgf.org/ontology/raw#gitCommit	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-raw:File
rdfs:label	Git commit
rdfs:range	xsd:string

16.3.15 Git modified

Whether the captured bytes differ from the file at gitPath in gitCommit. True also when the file is untracked in that commit. False means the capture is reproducible from the repository history.

ekg-raw:gitModified	
https://ekgf.org/ontology/raw#gitModified	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-raw:File
rdfs:label	Git modified
rdfs:range	xsd:boolean

16.3.16 Git path

The path of the file relative to the root of the repository work tree, using '/' as separator, as git records it.

ekg-raw:gitPath	
https://ekgf.org/ontology/raw#gitPath	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-raw:File
rdfs:label	Git path
rdfs:range	xsd:string

16.3.17 Git repository

The fetch URL of the default remote of the git repository whose work tree contained the file at capture time, recorded verbatim. Scp-style remotes such as git@host:org/repo.git are not IRIs, so the URL is a string.

ekg-raw:gitRepository	
https://ekgf.org/ontology/raw#gitRepository	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-raw:File
rdfs:label	Git repository
rdfs:range	xsd:string

16.3.18 Has column

Lists a view's physical source columns. A view can contain multiple columns; each column belongs to exactly one view. Columns sharing a mapped predicate retain separate identities.

ekg-raw:hasColumn	
https://ekgf.org/ontology/raw#hasColumn	
RDF predicate	**Value**
rdf:type	owl:InverseFunctionalProperty
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-raw:View
rdfs:label	Has column
rdfs:range	ekg-raw:Column
owl:inverseOf	ekg-raw:isColumnInView

16.3.19 Has View

One Source can have multiple Views

ekg-raw:hasView	
https://ekgf.org/ontology/raw#hasView	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	13 June 2020, 09:28:29 UTC
rdf:type	owl:InverseFunctionalProperty
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-raw:Source
rdfs:label	Has View
rdfs:range	ekg-raw:View
owl:inverseOf	ekg-raw:isViewIn

16.3.20 Inferred datatype

A datatype to which conformingCount of the column's counted occurrences conform: their literalValue has this datatype, or its lexical form is valid for it. Which datatypes are considered and what share of conforming occurrences is required are capture policy; conformingCount records the evidence. Captured values keep their source literals.

ekg-raw:inferredDatatype	
https://ekgf.org/ontology/raw#inferredDatatype	
RDF predicate	**Value**
rdf:type	owl:FunctionalProperty
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-raw:Column
rdfs:label	Inferred datatype
rdfs:range	rdfs:Datatype

16.3.21 Is column in view

Identifies the one view containing a physical source column; inverse of hasColumn. Assert either direction or both consistently. Reusing a mapped property does not reuse the column across views.

ekg-raw:isColumnInView	
https://ekgf.org/ontology/raw#isColumnInView	
RDF predicate	**Value**
rdf:type	owl:FunctionalProperty
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-raw:Column
rdfs:label	Is column in view
rdfs:range	ekg-raw:View

16.3.22 Is Row In View

The current row occurs in the given view.

ekg-raw:isRowInView	
https://ekgf.org/ontology/raw#isRowInView	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	13 June 2020, 09:36:41 UTC
rdf:type	owl:FunctionalProperty
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-raw:Row
rdfs:label	Is Row In View
rdfs:range	ekg-raw:View

16.3.23 Is View In

This View occurs in the given Source

ekg-raw:isViewIn	
https://ekgf.org/ontology/raw#isViewIn	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	13 June 2020, 09:30:00 UTC
rdf:type	owl:FunctionalProperty
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-raw:View
rdfs:label	Is View In

ekg-raw:isViewIn	
https://ekgf.org/ontology/raw#isViewIn	
RDF predicate	**Value**
rdfs:range	ekg-raw:Source

16.3.24 Legacy ID

The values in the key-column in the given view will always be mapped to raw:legacyId

ekg-raw:legacyId	
https://ekgf.org/ontology/raw#legacyId	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	13 June 2020, 09:38:33 UTC
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-raw:Row
rdfs:label	Legacy ID
rdfs:range	xsd:string

16.3.25 Legacy Id In Iri

The value of legacyId translated to an IRI-compliant lower kebab-case value

ekg-raw:legacyIdInIri	
https://ekgf.org/ontology/raw#legacyIdInIri	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	13 June 2020, 09:40:31 UTC
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:label	Legacy Id In Iri

16.3.26 Literal value

The source-supplied content of a captured value occurrence, represented as one RDF literal with its datatype or language retained. An empty string is content; a source null is represented by NullValue without this property.

ekg-raw:literalValue	
https://ekgf.org/ontology/raw#literalValue	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-raw:Value
rdfs:label	Literal value
rdfs:range	rdfs:Literal

16.3.27 Local name

The local name of the property that this Column is mapped to

ekg-raw:localName	
https://ekgf.org/ontology/raw#localName	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	13 June 2020, 09:32:26 UTC
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-raw:Column
rdfs:label	Local name
rdfs:range	xsd:string

16.3.28 Maximum

The greatest value, in the value space of the inferred datatype, among occurrences conforming to it, written as a literal of that datatype.

ekg-raw:max	
https://ekgf.org/ontology/raw#max	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-raw:Column
rdfs:label	Maximum
rdfs:range	rdfs:Literal

16.3.29 Mean

The arithmetic mean of the numeric values of occurrences conforming to a numeric inferred datatype.

ekg-raw:mean	
https://ekgf.org/ontology/raw#mean	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-raw:Column
rdfs:label	Mean
rdfs:range	xsd:double

16.3.30 Minimum

The least value, in the value space of the inferred datatype, among occurrences conforming to it, written as a literal of that datatype.

ekg-raw:min	
https://ekgf.org/ontology/raw#min	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-raw:Column
rdfs:label	Minimum
rdfs:range	rdfs:Literal

16.3.31 Null count

The number of captured value occurrences for this column that are NullValue.

ekg-raw:nullCount	
https://ekgf.org/ontology/raw#nullCount	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-raw:Column
rdfs:label	Null count
rdfs:range	xsd:integer

16.3.32 Predicate

The property that this Column is mapped to. A column is mapped to one property. It is an annotation property because its values are properties, which are not individuals.

ekg-raw:predicate	
https://ekgf.org/ontology/raw#predicate	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	13 June 2020, 09:33:37 UTC
rdf:type	owl:AnnotationProperty
rdfs:domain	ekg-raw:Column
rdfs:label	Predicate
rdfs:range	rdf:Property

16.3.33 Source text

The original text of a captured value as exposed by the source reader before value normalization, when such text is available. This optional string preserves lexical details independently of literalValue and may record a source's explicit null marker.

ekg-raw:sourceText	
https://ekgf.org/ontology/raw#sourceText	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-raw:Value
rdfs:label	Source text
rdfs:range	xsd:string

16.3.34 Standard deviation

The sample standard deviation (divisor n - 1) of the numeric values of occurrences conforming to a numeric inferred datatype.

ekg-raw:std	
https://ekgf.org/ontology/raw#std	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-raw:Column
rdfs:label	Standard deviation
rdfs:range	xsd:double

16.3.35 Most frequent value

A literalValue occurring most often among the counted occurrences. When several occur equally often, the one whose N-Triples form sorts first by code point.

ekg-raw:top	
https://ekgf.org/ontology/raw#top	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-raw:Column
rdfs:label	Most frequent value
rdfs:range	rdfs:Literal

16.3.36 Distinct value count

The number of distinct RDF terms among the literalValue of the counted occurrences; literals with different datatypes or language tags are distinct. Absence means the measure was not computed, not that there are no values.

ekg-raw:unique	
https://ekgf.org/ontology/raw#unique	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-raw:Column
rdfs:label	Distinct value count
rdfs:range	xsd:integer

16.3.37 Value of

Identifies the one source row at which a value occurrence was captured. The row and the occurrence's forColumn must belong to the same view.

ekg-raw:valueOf	
https://ekgf.org/ontology/raw#valueOf	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	13 June 2020, 09:17:59 UTC
rdf:type	owl:FunctionalProperty
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-raw:Value
rdfs:label	Value of
rdfs:range	ekg-raw:Row

16.3.38 View number

The 1-based position of a view within its source in the order the source exposes its views, such as the worksheet order of a workbook. A view name is not a stable substitute: names can repeat across sources and change between captures.

ekg-raw:viewNumber	
https://ekgf.org/ontology/raw#viewNumber	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-raw:View
rdfs:label	View number
rdfs:range	xsd:integer

16.3.39 View Row Number

The 0-based sequence number of the Row in the given View as it was encountered in the Source

ekg-raw:viewRowNumber	
https://ekgf.org/ontology/raw#viewRowNumber	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	13 June 2020, 09:41:42 UTC
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-raw:Row
rdfs:label	View Row Number
rdfs:range	xsd:integer

16.4 Shapes

16.4.1 Column Location Shape

Requires each physical source column to belong to one resource view. Counts both membership directions without requiring OWL inverse inference, and covers subclasses and untyped link endpoints.

ekg-raw:ColumnLocationShape	
https://ekgf.org/ontology/raw#ColumnLocationShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Column Location Shape
sh:nodeKind	sh:BlankNodeOrIRI

ekg-raw:ColumnLocationShape	
https://ekgf.org/ontology/raw#ColumnLocationShape	
RDF predicate	**Value**
sh:property	(sh:class: ekg-raw:View; sh:maxCount: 1; sh:message: A physical source column must belong to exactly one view, counting isColumnInView and inverse hasColumn together.; sh:minCount: 1; sh:nodeKind: sh:BlankNodeOrIRI; sh:path: (sh:alternativePath: (ekg-raw:isColumnInView, (sh:inversePath: ekg-raw:hasColumn))))
sh:targetClass	ekg-raw:Column
sh:targetObjectsOf	ekg-raw:forColumn
sh:targetObjectsOf	ekg-raw:hasColumn
sh:targetSubjectsOf	ekg-raw:isColumnInView

16.4.2 Column Profile Shape

Validates a column profile: single-valued measures with precise datatypes, counts bounded by the value count, an inferred datatype together with its evidence, statistics only with an inferred datatype, and ordered extremes and percentiles.

ekg-raw:ColumnProfileShape	
https://ekgf.org/ontology/raw#ColumnProfileShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Column Profile Shape
sh:class	ekg-raw:Column
sh:property	(sh:datatype: xsd:integer; sh:maxCount: 1; sh:minInclusive: 0; sh:path: ekg-raw:freq)
sh:property	(sh:maxCount: 1; sh:nodeKind: sh:IRI; sh:path: ekg-raw:inferredDatatype)
sh:property	(sh:datatype: xsd:integer; sh:maxCount: 1; sh:minInclusive: 0; sh:path: ekg-raw:unique)
sh:property	(sh:maxCount: 1; sh:nodeKind: sh:Literal; sh:path: ekg-raw:max)
sh:property	(sh:maxCount: 1; sh:nodeKind: sh:Literal; sh:path: ekg-raw:min)
sh:property	(sh:datatype: xsd:double; sh:maxCount: 1; sh:path: ekg-raw:75pct)
sh:property	(sh:datatype: xsd:double; sh:maxCount: 1; sh:path: ekg-raw:std)
sh:property	(sh:datatype: xsd:double; sh:maxCount: 1; sh:path: ekg-raw:mean)
sh:property	(sh:datatype: xsd:integer; sh:maxCount: 1; sh:minInclusive: 0; sh:path: ekg-raw:conformingCount)
sh:property	(sh:maxCount: 1; sh:nodeKind: sh:Literal; sh:path: ekg-raw:top)
sh:property	(sh:datatype: xsd:double; sh:maxCount: 1; sh:path: ekg-raw:25pct)
sh:property	(sh:datatype: xsd:integer; sh:maxCount: 1; sh:minInclusive: 0; sh:path: ekg-raw:nullCount)
sh:property	(sh:datatype: xsd:integer; sh:maxCount: 1; sh:minInclusive: 0; sh:path: ekg-raw:count)

ekg-raw:ColumnProfileShape	
https://ekgf.org/ontology/raw#ColumnProfileShape	
RDF predicate	**Value**
sh:property	(sh:datatype: xsd:double; sh:maxCount: 1; sh:path: ekg-raw:50pct)
sh:sparql	(sh:message: inferredDatatype and conformingCount are stated together.; sh:select: SPARQL SELECT query 1)
sh:sparql	(sh:message: min, 25pct, 50pct, 75pct and max must be in non-decreasing order.; sh:select: SPARQL SELECT query 2)
sh:sparql	(sh:message: min, max, mean, std and percentiles describe conforming values and require an inferredDatatype.; sh:select: SPARQL SELECT query 3)
sh:sparql	(sh:message: top and freq are stated together.; sh:select: SPARQL SELECT query 4)
sh:sparql	(sh:message: unique, freq and conformingCount cannot exceed count.; sh:select: SPARQL SELECT query 5)
sh:targetSubjectsOf	ekg-raw:25pct
sh:targetSubjectsOf	ekg-raw:50pct
sh:targetSubjectsOf	ekg-raw:75pct
sh:targetSubjectsOf	ekg-raw:conformingCount
sh:targetSubjectsOf	ekg-raw:count
sh:targetSubjectsOf	ekg-raw:freq
sh:targetSubjectsOf	ekg-raw:inferredDatatype
sh:targetSubjectsOf	ekg-raw:max
sh:targetSubjectsOf	ekg-raw:mean
sh:targetSubjectsOf	ekg-raw:min
sh:targetSubjectsOf	ekg-raw:nullCount
sh:targetSubjectsOf	ekg-raw:std
sh:targetSubjectsOf	ekg-raw:top
sh:targetSubjectsOf	ekg-raw:unique

16.4.2.1 SPARQL query 1

inferredDatatype and conformingCount are stated together.

```
SELECT $this WHERE {
    { $this <https://ekgf.org/ontology/raw#inferredDatatype> ?t FILTER NOT EXISTS { $this
<https://ekgf.org/ontology/raw#conformingCount> ?c } }
    UNION
    { $this <https://ekgf.org/ontology/raw#conformingCount> ?c FILTER NOT EXISTS { $this <
https://ekgf.org/ontology/raw#inferredDatatype> ?t } }
}
```

16.4.2.2 SPARQL query 2

min, 25pct, 50pct, 75pct and max must be in non-decreasing order.

```
SELECT $this WHERE {
    { $this <https://ekgf.org/ontology/raw#min> ?a ; <https://ekgf.org/ontology/raw#25pct>
 ?b }
    UNION
    { $this <https://ekgf.org/ontology/raw#25pct> ?a ; <https://ekgf.org/ontology/raw#50pc
t> ?b }
    UNION
    { $this <https://ekgf.org/ontology/raw#50pct> ?a ; <https://ekgf.org/ontology/raw#75pc
t> ?b }
    UNION
    { $this <https://ekgf.org/ontology/raw#75pct> ?a ; <https://ekgf.org/ontology/raw#max>
 ?b }
    UNION
    { $this <https://ekgf.org/ontology/raw#min> ?a ; <https://ekgf.org/ontology/raw#max> ?
b }
    FILTER (?b < ?a)
}
```

16.4.2.3 SPARQL query 3

min, max, mean, std and percentiles describe conforming values and require an inferredDatatype.

```
SELECT $this WHERE {
    $this <https://ekgf.org/ontology/raw#min>|<https://ekgf.org/ontology/raw#max>|<https:/
/ekgf.org/ontology/raw#mean>|<https://ekgf.org/ontology/raw#std>|<https://ekgf.org/ontolog
y/raw#25pct>|<https://ekgf.org/ontology/raw#50pct>|<https://ekgf.org/ontology/raw#75pct> ?
stat .
    FILTER NOT EXISTS { $this <https://ekgf.org/ontology/raw#inferredDatatype> ?t }
}
```

16.4.2.4 SPARQL query 4

top and freq are stated together.

```
SELECT $this WHERE {
    { $this <https://ekgf.org/ontology/raw#top> ?t FILTER NOT EXISTS { $this <https://ekgf
.org/ontology/raw#freq> ?f } }
    UNION
    { $this <https://ekgf.org/ontology/raw#freq> ?f FILTER NOT EXISTS { $this <https://ekg
f.org/ontology/raw#top> ?t } }
}
```

16.4.2.5 SPARQL query 5

unique, freq and conformingCount cannot exceed count.

```
SELECT $this WHERE {
    $this <https://ekgf.org/ontology/raw#count> ?count .
    $this <https://ekgf.org/ontology/raw#unique>|<https://ekgf.org/ontology/raw#freq>|<htt
ps://ekgf.org/ontology/raw#conformingCount> ?part .
    FILTER (?part > ?count)
}
```

16.4.3 Formula R1C1 Shape

Validates relative R1C1 formulas on value occurrences and the single shared derivation rule of a column.

ekg-raw:FormulaR1C1Shape	
https://ekgf.org/ontology/raw#FormulaR1C1Shape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Formula R1C1 Shape
sh:message	formulaR1C1 belongs to a captured value occurrence or to a column whose formulas share one relative form.
sh:or	((sh:class: ekg-raw:Value), (sh:class: ekg-raw:Column))
sh:property	(sh:datatype: xsd:string; sh:maxCount: 1; sh:message: A value or column has at most one relative R1C1 formula, a string.; sh:path: ekg-raw:formulaR1C1)
sh:targetSubjectsOf	ekg-raw:formulaR1C1

16.4.4 Formula Shape

Validates verbatim spreadsheet formulas on captured value occurrences.

ekg-raw:FormulaShape	
https://ekgf.org/ontology/raw#FormulaShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Formula Shape
sh:class	ekg-raw:Value
sh:property	(sh:datatype: xsd:string; sh:maxCount: 1; sh:message: A value has at most one formula: a non-empty string in A1 notation without the leading '='.; sh:path: ekg-raw:formula; sh:pattern: ^[^=])
sh:targetSubjectsOf	ekg-raw:formula

16.4.5 Git Origin Shape

Validates the git provenance of a captured file: object id syntax, a relative path, and that a commit is always accompanied by the path and modification state needed to reproduce the capture.

ekg-raw:GitOriginShape	
https://ekgf.org/ontology/raw#GitOriginShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Git Origin Shape
sh:message	A file in a repository records gitCommit, gitPath and gitModified together; gitRepository requires them. gitBlob alone is valid outside a repository.
sh:or	((sh:property: (sh:maxCount: 0; sh:path: ekg-raw:gitRepository); sh:property: (sh:maxCount: 0; sh:path: ekg-raw:gitPath); sh:property: (sh:maxCount: 0; sh:path: ekg-raw:gitModified); sh:property: (sh:maxCount: 0; sh:path: ekg-raw:gitCommit)), (sh:property: (sh:minCount: 1; sh:path: ekg-raw:gitPath); sh:property: (sh:minCount: 1; sh:path: ekg-

ekg-raw:GitOriginShape	
https://ekgf.org/ontology/raw#GitOriginShape	
RDF predicate	**Value**
	raw:gitCommit); sh:property: (sh:minCount: 1; sh:path: ekg-raw:gitModified)))
sh:property	(sh:datatype: xsd:string; sh:maxCount: 1; sh:message: gitBlob is one lowercase hexadecimal SHA-1 or SHA-256 object id.; sh:path: ekg-raw:gitBlob; sh:pattern: ^([0-9a-f]{40}|[0-9a-f]{64})$)
sh:property	(sh:datatype: xsd:string; sh:maxCount: 1; sh:message: gitCommit is one lowercase hexadecimal SHA-1 or SHA-256 object id.; sh:path: ekg-raw:gitCommit; sh:pattern: ^([0-9a-f]{40}|[0-9a-f]{64})$)
sh:property	(sh:datatype: xsd:string; sh:maxCount: 1; sh:message: gitPath is one work-tree-relative path using '/' separators.; sh:path: ekg-raw:gitPath; sh:pattern: ^[^/\\][^\\]*$)
sh:property	(sh:datatype: xsd:string; sh:maxCount: 1; sh:path: ekg-raw:gitRepository)
sh:property	(sh:datatype: xsd:boolean; sh:maxCount: 1; sh:path: ekg-raw:gitModified)
sh:targetSubjectsOf	ekg-raw:gitBlob
sh:targetSubjectsOf	ekg-raw:gitCommit
sh:targetSubjectsOf	ekg-raw:gitModified
sh:targetSubjectsOf	ekg-raw:gitPath
sh:targetSubjectsOf	ekg-raw:gitRepository

16.4.6 Row Location Shape

Requires each supplied source row to identify one resource view, including subclasses and untyped row-link endpoints. Coordinate bounds and view-to-source ownership are separate constraints.

ekg-raw:RowLocationShape	
https://ekgf.org/ontology/raw#RowLocationShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Row Location Shape
sh:nodeKind	sh:BlankNodeOrIRI
sh:property	(sh:class: ekg-raw:View; sh:maxCount: 1; sh:message: A source row must belong to exactly one view through isRowInView.; sh:minCount: 1; sh:nodeKind: sh:BlankNodeOrIRI; sh:path: ekg-raw:isRowInView)
sh:targetClass	ekg-raw:Row
sh:targetObjectsOf	ekg-raw:valueOf
sh:targetSubjectsOf	ekg-raw:isRowInView

16.4.7 Value Content Shape

Validates the content of raw value occurrences, including subclasses and untyped resources using capture properties. Accepts typed and language-tagged literals and empty strings, distinguishes explicit source nulls, and checks optional original source text. Location constraints are enforced separately by ValueLocationShape, RowLocationShape, and ColumnLocationShape.

ekg-raw:ValueContentShape	
https://ekgf.org/ontology/raw#ValueContentShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Value Content Shape
sh:message	A captured value must have exactly one literalValue or be explicitly typed Null-Value without a literalValue. Source text alone does not supply the content.
sh:nodeKind	sh:BlankNodeOrIRI
sh:property	(sh:datatype: xsd:string; sh:maxCount: 1; sh:path: ekg-raw:sourceText)
sh:targetClass	ekg-raw:Value
sh:targetSubjectsOf	ekg-raw:forColumn
sh:targetSubjectsOf	ekg-raw:forPredicate
sh:targetSubjectsOf	ekg-raw:literalValue
sh:targetSubjectsOf	ekg-raw:sourceText
sh:targetSubjectsOf	ekg-raw:valueOf
sh:xone	((sh:class: ekg-raw:NullValue; sh:property: (sh:maxCount: 0; sh:path: ekg-raw:literalValue)), (sh:not: (sh:class: ekg-raw:NullValue); sh:property: (sh:maxCount: 1; sh:minCount: 1; sh:nodeKind: sh:Literal; sh:path: ekg-raw:literalValue)))

16.4.8 Value Location Shape

Requires one resource row and one resource column for each captured value, including nulls, and rejects row/column view mismatches. Predicate mappings cannot substitute for physical column links.

ekg-raw:ValueLocationShape	
https://ekgf.org/ontology/raw#ValueLocationShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Value Location Shape
sh:nodeKind	sh:BlankNodeOrIRI
sh:property	(sh:class: ekg-raw:Column; sh:maxCount: 1; sh:message: A captured value must identify exactly one physical source column through forColumn.; sh:minCount: 1; sh:nodeKind: sh:BlankNodeOrIRI; sh:path: ekg-raw:forColumn)
sh:property	(sh:class: ekg-raw:Row; sh:maxCount: 1; sh:message: A captured value must identify exactly one source row through valueOf.; sh:minCount: 1; sh:nodeKind: sh:BlankNodeOrIRI; sh:path: ekg-raw:valueOf)
sh:sparql	(sh:message: A captured value's row and column must belong to the same view.; sh:select: SPARQL SELECT query 1)
sh:targetClass	ekg-raw:Value
sh:targetSubjectsOf	ekg-raw:forColumn

ekg-raw:ValueLocationShape	
https://ekgf.org/ontology/raw#ValueLocationShape	
RDF predicate	**Value**
sh:targetSubjectsOf	ekg-raw:forPredicate
sh:targetSubjectsOf	ekg-raw:literalValue
sh:targetSubjectsOf	ekg-raw:sourceText
sh:targetSubjectsOf	ekg-raw:valueOf

16.4.8.1 SPARQL query 1

A captured value's row and column must belong to the same view.

```
SELECT $this WHERE {
    $this <https://ekgf.org/ontology/raw#valueOf>/<https://ekgf.org/ontology/raw#isRowInView> ?rowView .
    $this <https://ekgf.org/ontology/raw#forColumn>/(<https://ekgf.org/ontology/raw#isColumnInView>|^<https://ekgf.org/ontology/raw#hasColumn>) ?columnView .
    FILTER (?rowView != ?columnView)
}
```

16.4.9 View Number Shape

Validates the 1-based position of a view within its source.

ekg-raw:ViewNumberShape	
https://ekgf.org/ontology/raw#ViewNumberShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	View Number Shape
sh:property	(sh:datatype: xsd:integer; sh:maxCount: 1; sh:message: A view has at most one viewNumber, a positive xsd:integer.; sh:minInclusive: 1; sh:path: ekg-raw:viewNumber)
sh:targetSubjectsOf	ekg-raw:viewNumber

17 Specification by Example Ontology

17.1 Overview

Specification by Example Ontology

Identifier	Value
Ontology IRI / Namespace	`https://ekgf.org/ontology/specification-by-example#`
Prefix	`ekg-sbe`

A vocabulary for executable story scenarios, including initial graph and dataset states, story requests, expected results, chained scenarios, and captured run-time values.

17.2 Classes

17.2.1 Concept Value

This is the value for a test concept

`ekg-sbe:ConceptValue`	
`https://ekgf.org/ontology/specification-by-example#ConceptValue`	
RDF predicate	**Value**
`rdf:type`	`owl:Class`
`rdfs:label`	Concept Value
`rdfs:label`	Valor do Conceito

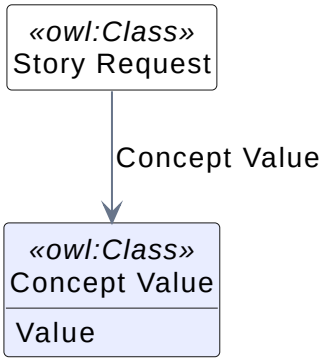


Diagram note

17.2.2 Graph

A Graph in a Scenario State, every Scenario State (either the Given-state or the Then-state) can have one or more Graphs.

ekg-sbe:Graph	
https://ekgf.org/ontology/specification-by-example#Graph	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Graph

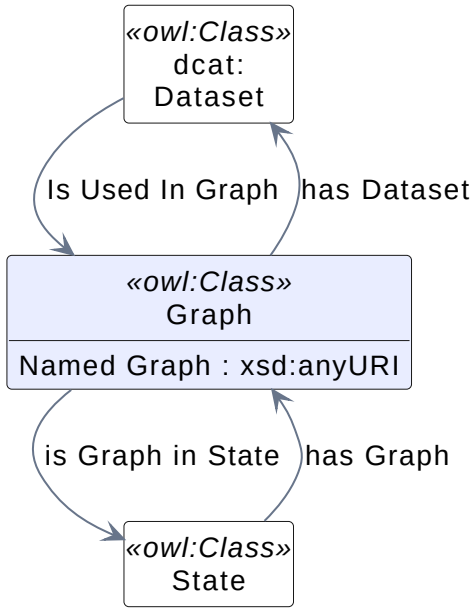


Diagram note

17.2.3 Scenario

An executable example specifying the request, initial state, and expected state or results for a story. Scenarios can be chained through follows.

ekg-sbe:Scenario	
https://ekgf.org/ontology/specification-by-example#Scenario	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Scenario

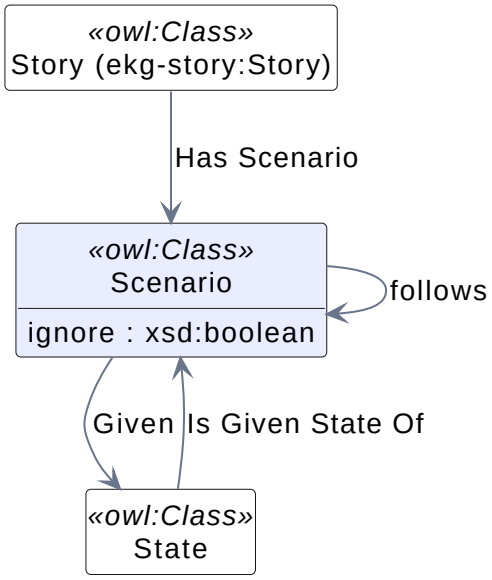


Diagram note

17.2.4 State

A set of graph and dataset conditions used to describe an initial or expected situation in a scenario.

ekg-sbe:State	
https://ekgf.org/ontology/specification-by-example#State	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	State

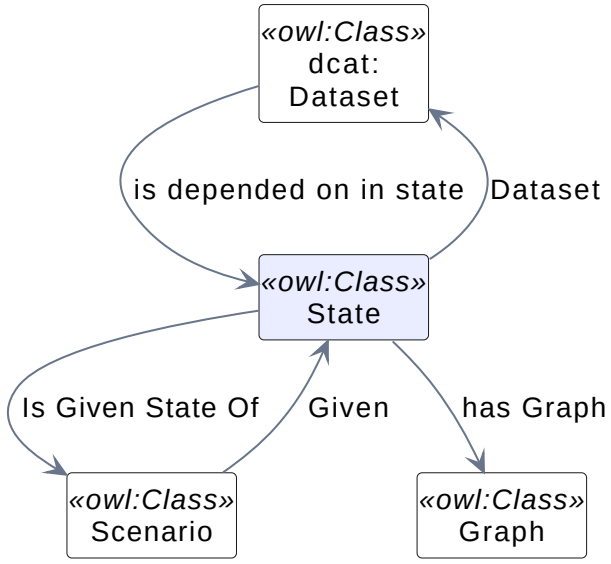


Diagram note

17.2.5 Story Request

The inputs to the story that is being tested in the given Scenario

ekg-sbe:StoryRequest	
https://ekgf.org/ontology/specification-by-example#StoryRequest	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Story Request

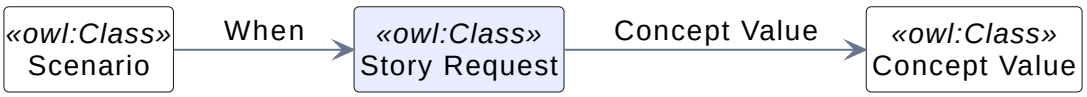


Diagram note

17.2.6 capture

A literal of this datatype in an expected cell names a capture: "id"^^sbe:capture binds the actual value on first sight and requires equality everywhere the same name appears in the scenario. For asserting that two cells, or two rows, carry the same run-time value without pinning what it is.

ekg-sbe:capture	
https://ekgf.org/ontology/specification-by-example#capture	
RDF predicate	**Value**
rdf:type	rdfs:Datatype
rdfs:label	capture

17.3 Properties

17.3.1 Concept Value

Relates a story request in a scenario to a value supplied for one of the concepts involved in that request.

ekg-sbe:conceptValue	
https://ekgf.org/ontology/specification-by-example#conceptValue	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-sbe:StoryRequest
rdfs:label	Concept Value
rdfs:range	ekg-sbe:ConceptValue

17.3.2 Dataset

Identifies a dataset on which a scenario state depends; it is the inverse of isDependedOnInState.

ekg-sbe:dataset	
https://ekgf.org/ontology/specification-by-example#dataset	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-sbe:State
rdfs:label	Dataset
rdfs:range	dcat:Dataset
owl:inverseOf	ekg-sbe:isDependedOnInState

17.3.3 follows

This scenario runs after the named scenario, on the same branch, with the predecessor's captures available: a value "id"^^sbe:capture in this scenario's `when` is substituted with what the predecessor's `then` bound to "id". Scenarios linked by `follows` form a chain — one branch prepared once, steps in dependency order — and a red step leaves the remaining steps unrun. A scenario without `follows` is a chain of one.

ekg-sbe:follows	
https://ekgf.org/ontology/specification-by-example#follows	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-sbe:Scenario
rdfs:label	follows
rdfs:range	ekg-sbe:Scenario

17.3.4 Given

Relates a scenario to its initial state, before the story request is executed.

ekg-sbe:given	
https://ekgf.org/ontology/specification-by-example#given	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-sbe:Scenario
rdfs:label	Given
rdfs:range	ekg-sbe:State
owl:inverseOf	ekg-sbe:isGivenStateOf

17.3.5 has Dataset

a State can have one or more Graphs where each Graph can have one or more Datasets

ekg-sbe:hasDataset	
https://ekgf.org/ontology/specification-by-example#hasDataset	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-sbe:Graph
rdfs:label	has Dataset
rdfs:range	dcat:Dataset
owl:inverseOf	ekg-sbe:isUsedInGraph

17.3.6 has Graph

Any Scenario State can have one or more Graphs

ekg-sbe:hasGraph	
https://ekgf.org/ontology/specification-by-example#hasGraph	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-sbe:State
rdfs:label	has Graph
rdfs:range	ekg-sbe:Graph
owl:inverseOf	ekg-sbe:isGraphInState

17.3.7 Has Scenario

Relates a story to an example scenario that validates its behavior; it is the inverse of validates.

ekg-sbe:hasScenario	
https://ekgf.org/ontology/specification-by-example#hasScenario	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-story:Story
rdfs:label	Has Scenario
rdfs:range	ekg-sbe:Scenario
owl:inverseOf	ekg-sbe:validates

17.3.8 ignore

Skip this scenario. Kept in the model rather than as a runner flag so a scenario can be parked with its reason alongside it, and so the decision travels with the data.

ekg-sbe:ignore	
https://ekgf.org/ontology/specification-by-example#ignore	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-sbe:Scenario
rdfs:label	ignore
rdfs:range	xsd:boolean

17.3.9 is depended on in state

Usually inferred from the inverse `sbe:dataset`

ekg-sbe:isDependedOnInState	
https://ekgf.org/ontology/specification-by-example#isDependedOnInState	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	dcat:Dataset
rdfs:label	is depended on in state
rdfs:range	ekg-sbe:State

17.3.10 Is Given State Of

Usually inferred from inverse `sbe:given`

ekg-sbe:isGivenStateOf	
https://ekgf.org/ontology/specification-by-example#isGivenStateOf	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-sbe:State
rdfs:label	Is Given State Of
rdfs:range	ekg-sbe:Scenario

17.3.11 is Graph in State

Usually inferred, a Graph can be part of one or more States

ekg-sbe:isGraphInState	
https://ekgf.org/ontology/specification-by-example#isGraphInState	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-sbe:Graph
rdfs:label	is Graph in State

ekg-sbe:isGraphInState	
https://ekgf.org/ontology/specification-by-example#isGraphInState	
RDF predicate	**Value**
rdfs:range	ekg-sbe:State

17.3.12 Is Used In Graph

Relates a dataset to a scenario graph that uses it; it is the inverse of hasDataset.

ekg-sbe:isUsedInGraph	
https://ekgf.org/ontology/specification-by-example#isUsedInGraph	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	dcat:Dataset
rdfs:label	Is Used In Graph
rdfs:range	ekg-sbe:Graph

17.3.13 Named Graph

Records the IRI naming a graph in a scenario state as an xsd:anyURI literal.

ekg-sbe:namedGraph	
https://ekgf.org/ontology/specification-by-example#namedGraph	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-sbe:Graph
rdfs:label	Named Graph
rdfs:range	xsd:anyURI

17.3.14 Story Parameters For Scenario

Usually inferred from inverse `sbe:when`

ekg-sbe:storyParametersForScenario	
https://ekgf.org/ontology/specification-by-example#storyParametersForScenario	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-sbe:State
rdfs:label	Story Parameters For Scenario
rdfs:range	ekg-sbe:Scenario
owl:inverseOf	ekg-sbe:when

17.3.15 Then

Relates a scenario to the state or results expected after its story request is executed.

ekg-sbe:then	
https://ekgf.org/ontology/specification-by-example#then	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-sbe:Scenario
rdfs:label	Then
rdfs:range	ekg-sbe:State

17.3.16 Validates

Identifies the story whose behavior a scenario validates. The functional-property axiom assigns at most one story to a scenario.

ekg-sbe:validates	
https://ekgf.org/ontology/specification-by-example#validates	
RDF predicate	**Value**
rdf:type	owl:FunctionalProperty
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-sbe:Scenario
rdfs:label	Validates
rdfs:range	ekg-story:Story

17.3.17 Value

Contains the literal value represented by a scenario concept-value resource. No literal datatype restriction is declared.

ekg-sbe:value	
https://ekgf.org/ontology/specification-by-example#value	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-sbe:ConceptValue
rdfs:label	Value

17.3.18 When

Relates a scenario to the story request that is executed between the initial and expected states.

ekg-sbe:when	
https://ekgf.org/ontology/specification-by-example#when	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-sbe:Scenario
rdfs:label	When
rdfs:range	ekg-sbe:StoryRequest

17.4 Named Individuals

17.4.1 any

As an expected cell's value: any bound value satisfies it. For results a story derives at run time — an IRI minted with UUID(), a NOW() stamp — where the scenario asserts presence, not identity.

ekg-sbe:any	
https://ekgf.org/ontology/specification-by-example#any	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdfs:label	any

18 Story Ontology

18.1 Overview

Story Ontology

Identifier	Value
Ontology IRI / Namespace	`https://ekgf.org/ontology/story#`
Prefix	`ekg-story`

A story describes an interaction within a use case, including the persona playing it, the outcome it contributes to, and the concepts it uses. Describe these as **input concepts**, **output concepts**, and **dependent concepts**. An input concept is part of the business requirement; its technical manifestations can include API names, query variables, or source-code identifiers.

18.1.1 Implementations and ownership

`implementedBy` links a story to its implementations. Its inverse, `implementationOf`, identifies the single story an implementation belongs to. A story can have zero or more implementations. Each implementation belongs to exactly one story.

The OWL class restriction records exactly one owner, and `implementationOf` is functional. OWL can equate different owner names when their distinctness has not been asserted. `StoryImplementationShape` validates the supplied data and rejects missing or multiple owner terms, counting both link directions. An owner asserted in both directions is counted once. The shape also covers implementation subclasses and relationship endpoints whose types are inferred.

The number of implementations does not select which one executes. A Story Service resolves an appropriate runnable implementation in its execution context and rejects invocation when that selection is absent or ambiguous.

18.1.1.1 Migration and validation

Replace every `hasImplementation` assertion with `implementedBy`, or use `implementationOf` in the other direction. `LegacyImplementationShape` rejects the removed predicate, including when a canonical link is also present. Replace previously loaded ontology graphs so the old declaration and inverse axiom are removed.

For a Turtle data file, validate with the Story Ontology as both the shapes graph and the ontology graph. RDFS inference supplies endpoint types and subclass relationships:

```
pyshacl -s ekgf-story.ttl -e ekgf-story.ttl -i rdfs implementation-data.ttl
```

Either direction can be supplied:

```
@prefix ekg-story: <https://ekgf.org/ontology/story#> .
@prefix ex: <https://example.org/> .

ex:findCustomer ekg-story:implementedBy ex:findCustomerSPARQL .
# Equivalently:
# ex:findCustomerSPARQL ekg-story:implementationOf ex:findCustomer .
```

Validate the graph or deliberately selected graph set representing one coherent version context. Historical snapshots can repeat the same story and implementation IRIs without adding owners. If snapshots link

the same implementation IRI to different story-version IRIs, each snapshot must be validated separately; combining them produces a multiple-owner violation. Named graphs alone do not define this context or exempt ownership constraints.

18.1.2 Ownership and reuse

A story owns its concept-usage objects. Each usage belongs to exactly one story and references exactly one reusable `ekg-concept:DataElementConcept` through `ekg-concept:hasConcept`. Different stories reference the same concept through separate usage objects. Deleting a story removes its usage objects; the referenced concepts retain their identity.

Concepts are members of `ekg-concept:ConceptSet` resources. A use case owns its concept vocabularies through `ekg-use-case:ownsConceptSet`; stories can draw from their use case's vocabularies and those inherited from ancestor use cases. Using a concept does not transfer its ownership to the story.

18.1.3 Concept-usage model

The OWL classes, property ranges, ownership restrictions, input flags, and SHACL constraints below are declared in this ontology.

Story predicate	Range of the predicate
`hasInputConcept`	`StoryInput`
`hasOutputConcept`	`StoryOutput`
`hasDependentConcept`	`StoryDependent`

The three range classes specialize a common `StoryConceptUsage` class. The three predicates specialize `hasConceptUsage`. A specific predicate lets the reasoner infer both the specific usage class and its common superclass, and the general relationship. A separate usage-type value is unnecessary. These usage classes describe relationships to concepts.

For example:

```
@prefix ekg-story: <https://ekgf.org/ontology/story#> .
@prefix ekg-concept: <https://ekgf.org/ontology/concept#> .
@prefix ex: <https://example.org/> .

ex:findCustomer ekg-story:hasInputConcept [
    ekg-concept:hasConcept ex:customerIdentifier ;
    ekg-story:isMandatory true
] .

ex:customerIdentifier a ekg-concept:DataElementConcept .
```

The blank node is the story-owned input-concept usage. Its type is inferred from the predicate's range. The customer-identifier concept can be reused by other stories.

18.1.4 Input-concept behavior

An input usage can carry `isMandatory` and `isServerProvided`. `isMandatory` means that the input is required for execution; `isServerProvided` means that the Story Service supplies it. The flags are independent: an input can be mandatory and server-provided.

A Story Service interprets each absent flag as **false**. Missing flags remain valid data. `StoryInputShape` property shapes record `sh:defaultValue false`, accept at most one value per flag, and require supplied values to be `xsd:boolean`. Invalid values are validation errors.

`sh:defaultValue` records the application default; SHACL Core validation does not insert it. OWL reasoning does not infer false from absence. A service applies these defaults during execution; materializing explicit false triples is an application choice.

`StoryConceptUsageShape` requires exactly one owning story, one referenced concept, and one of the three usage roles. The role classes are disjoint: use separate usage objects if a concept is both an input and an output. The general `hasConceptUsage` predicate requires an explicit or inferred role class. Either input flag infers `StoryInput` through its domain, even when false; specialized predicates infer their roles through their ranges. Blank nodes and IRIs are accepted for usage objects. Shared concepts are never usage objects. Consumers must retain usage identity rather than merge usages by concept IRI: distinct usages of one concept can have different flags.

`concept:hasConcept` retains its general meaning and has no global domain. Its exactly-one restriction applies to `StoryConceptUsage`, so other resources can reference concepts without becoming story usages.

18.1.5 Migration from earlier generations

For an old direct story-to-concept link, create a story-owned usage object and move the concept reference to its `concept:hasConcept` property. The `hasInputConcept`, `hasOutputConcept`, and `hasDependentConcept` predicates now range over usages, not directly over concepts. Never turn the reusable concept itself into the usage object.

Replace `hasInput` and `hasOutput` with `hasInputConcept` and `hasOutputConcept`, retaining their usage objects and concept references. Replace `MandatoryParameter` with `StoryInput` plus `isMandatory true`. Replace `OptionalParameter` with `StoryInput`; `isMandatory false` can be explicit or absent. Retired `serverProvidedIRI` assertions must be resolved to the actual input usage and replaced with `isServerProvided` there; a story-level flag does not identify that input automatically.

`OptionalStoryOutput` is retired. Resolve the intended output contract when migrating those records; input flags do not express output optionality. `LegacyConceptUsageShape` rejects all retired predicates and classes rather than silently translating them. Replace previously loaded ontology graphs to remove old ranges, parameter classes, and the former global domain of `concept:hasConcept`.

All named validation shapes belong to this ontology's shapes graph; nested shapes are included through their entry definitions. Validate with the Story Ontology as both shapes and ontology graph and enable RDFS inference, as in the validation command above. See the repository's concept-usage guide for a complete executable example.

See the Method's Story/Concept Relationship, its concept-vocabulary ownership rules, and SHACL default-value semantics.

18.2 Classes

18.2.1 Agent Job Implementation

A story implementation expressed as a job performed by an agent.

ekg-story:AgentJobImplementation	
https://ekgf.org/ontology/story#AgentJobImplementation	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Agent Job Implementation
rdfs:subClassOf	ekg-story:StoryImplementation

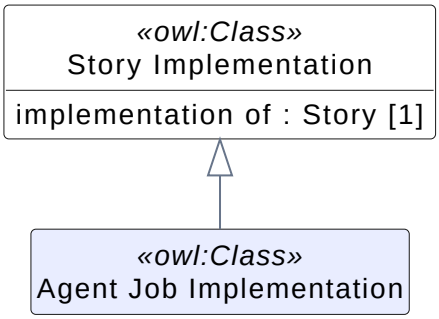


Diagram note

18.2.2 API Proxy Implementation

A story implementation that delegates the interaction to an API through a proxy.

ekg-story:ApiProxyImplementation	
https://ekgf.org/ontology/story#ApiProxyImplementation	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	API Proxy Implementation
rdfs:subClassOf	ekg-story:StoryImplementation

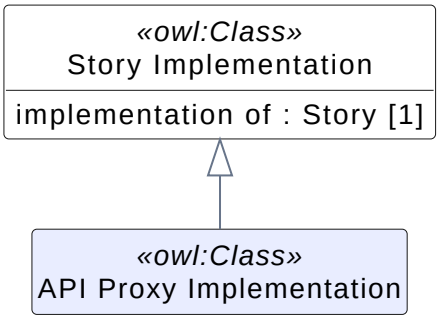


Diagram note

18.2.3 Named Statement Implementation

A story implementation identified through a named executable statement.

ekg-story:NamedStatementImplementation	
https://ekgf.org/ontology/story#NamedStatementImplementation	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Named Statement Implementation
rdfs:subClassOf	ekg-story:StoryImplementation

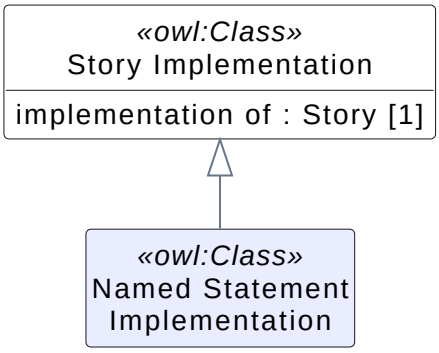


Diagram note

18.2.4 Story

A description of an interaction in a use case, identifying its participating persona, inputs, outputs, implementations, and the outcomes it helps achieve.

ekg-story:Story	
https://ekgf.org/ontology/story#Story	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Story
owl:disjointWith	ekg-story:StoryImplementation

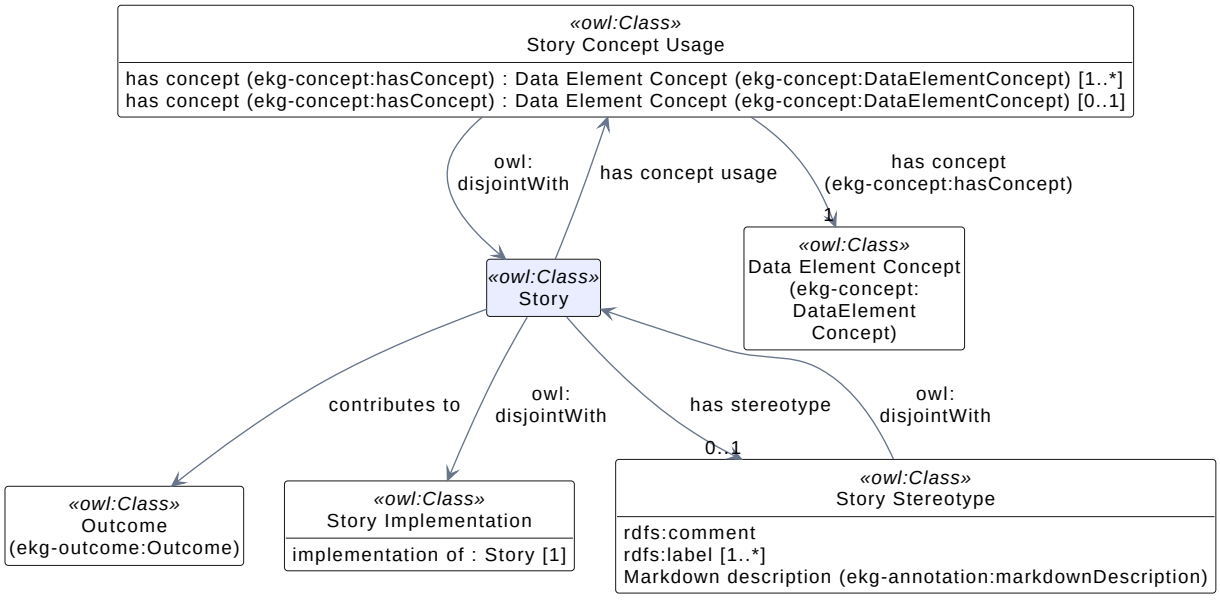


Diagram note

18.2.5 Story Concept Usage

A story-owned relationship object with exactly one owning story and exactly one referenced data element concept. Its input, output, or dependent role describes the usage, not the reusable concept.

ekg-story:StoryConceptUsage	
https://ekgf.org/ontology/story#StoryConceptUsage	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Story Concept Usage
rdfs:subClassOf	(rdf:type: owl:Restriction; owl:minCardinality: 1; owl:onProperty: (owl:inverseOf: ekg-story:hasConceptUsage))
rdfs:subClassOf	(rdf:type: owl:Restriction; owl:minCardinality: 1; owl:onProperty: ekg-concept:hasConcept)
rdfs:subClassOf	(rdf:type: owl:Restriction; owl:maxCardinality: 1; owl:onProperty: ekg-concept:hasConcept)
rdfs:subClassOf	(rdf:type: owl:Restriction; owl:maxCardinality: 1; owl:onProperty: (owl:inverseOf: ekg-story:hasConceptUsage))
owl:disjointWith	ekg-concept:ConceptManifestation
owl:disjointWith	ekg-concept:DataElementConcept
owl:disjointWith	ekg-story:Story

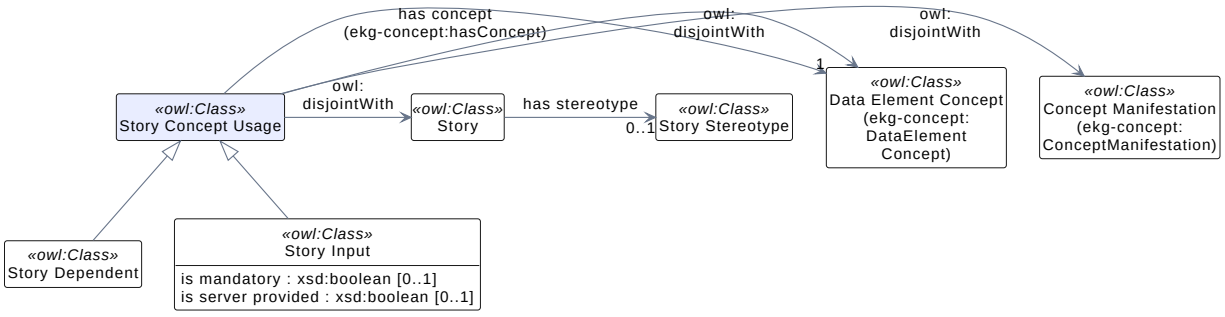


Diagram note

18.2.6 Story Dependent

This usage records a concept used inside the story's implementation without exposing it as an input or output, such as a join or filter concept. It is a subclass of `StoryConceptUsage` and the range of `hasDependentConcept`. The story owns the usage; `concept:hasConcept` points to the shared meaning. Input flags apply only to input usages.

ekg-story:StoryDependent	
https://ekgf.org/ontology/story#StoryDependent	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Story Dependent
rdfs:subClassOf	ekg-story:StoryConceptUsage

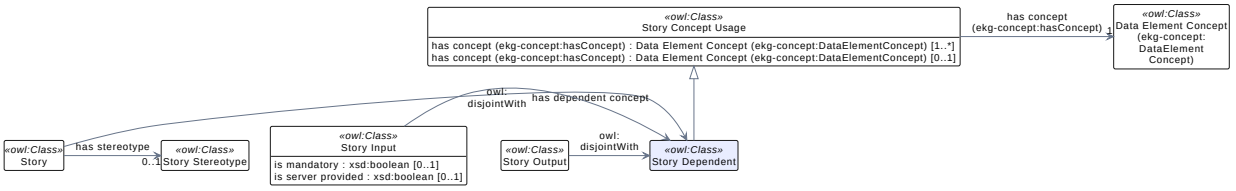


Diagram note

18.2.7 Story Implementation

Each implementation belongs to exactly one story through `implementationOf`. The inverse `implementedBy` lets a story refer to zero or more implementations. An implementation can evolve independently of its story and coexist with other implementations of that story.

The OWL cardinality restriction expresses ownership. SHACL validates actual owner assertions in either direction, rejecting missing owners and multiple owner terms. This validation is required because OWL's open-world semantics does not report a missing assertion as a data error, and functionality can equate owner names.

Validate within one coherent version context. Repeated assertions about the same story identity across snapshots still identify one owner. Different story-version IRIs count as different owner terms when their graphs are combined; historical snapshots must be validated in their intended contexts.

The ontology's Markdown description includes the validation command and migration from `hasImplementation` to the canonical inverse pair.

ekg-story:StoryImplementation	
https://ekgf.org/ontology/story#StoryImplementation	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Story Implementation
rdfs:subClassOf	(rdf:type: owl:Restriction; owl:cardinality: 1; owl:onProperty: ekg-story:implementationOf)

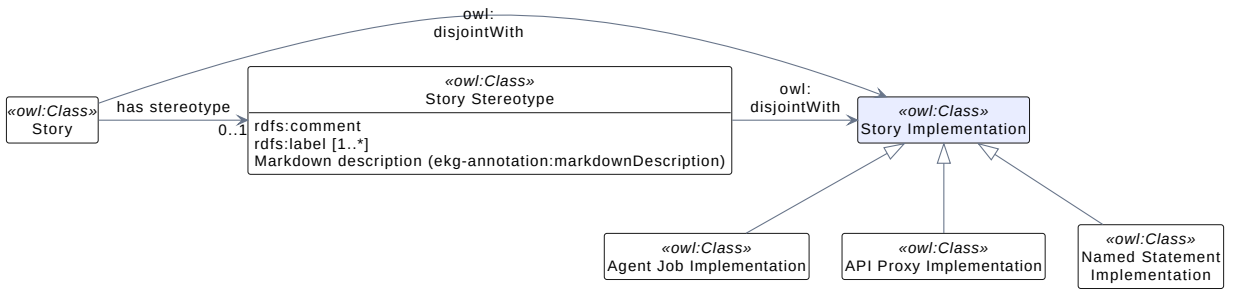


Diagram note

18.2.8 Story Input

This resource describes how one story uses a reusable concept as an input. The story owns this usage object; `ekg-concept:hasConcept` references the shared data element concept. Separate stories use separate usage objects even when they reference the same concept.

This class is a subclass of `StoryConceptUsage` and the range of `hasInputConcept`. Input characteristics such as requiredness and server provision belong on the usage. Their false defaults are applied by the Story Service.

`StoryInputShape` validates the flags, and `StoryConceptUsageShape` validates the owner, concept, and usage role. See the ontology's Markdown description for validation and migration instructions.

ekg-story:StoryInput	
https://ekgf.org/ontology/story#StoryInput	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	1 April 2020, 17:24:36 UTC
rdf:type	owl:Class
rdfs:label	Story Input
rdfs:subClassOf	ekg-story:StoryConceptUsage
owl:disjointWith	ekg-story:StoryDependent
owl:disjointWith	ekg-story:StoryOutput

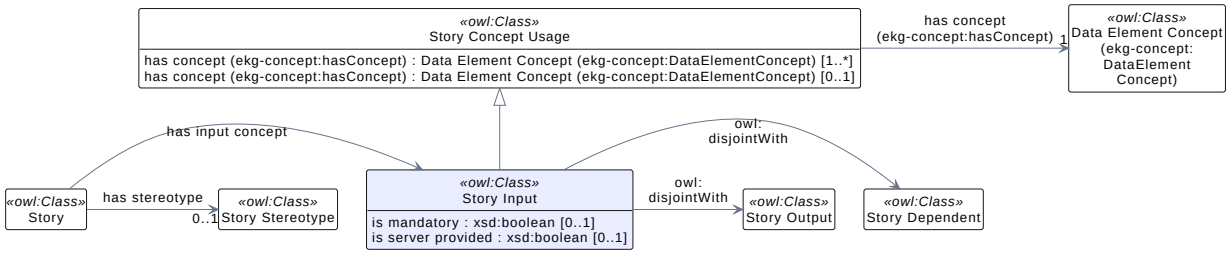


Diagram note

18.2.9 Story Output

This resource describes how one story uses a reusable concept as an output. The story owns the usage object; `ekg-concept:hasConcept` references the shared data element concept. A concept can describe a compound output, including the meaning represented by a whole class or shape.

This class is a subclass of `StoryConceptUsage` and the range of `hasOutputConcept`. The input flags do not apply to outputs. Use a separate input usage if the same concept is also an input to this story.

ekg-story:StoryOutput	
https://ekgf.org/ontology/story#StoryOutput	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	7 April 2020, 20:40:12 UTC
rdf:type	owl:Class
rdfs:label	Story Output
rdfs:subClassOf	ekg-story:StoryConceptUsage

ekg-story:StoryOutput	
https://ekgf.org/ontology/story#StoryOutput	
RDF predicate	**Value**
owl:disjointWith	ekg-story:StoryDependent

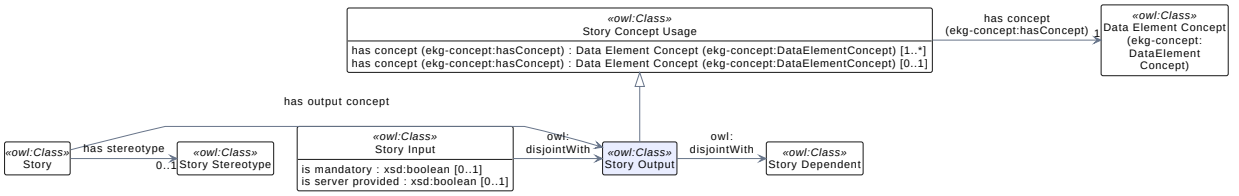


Diagram note

18.2.10 Story Stereotype

`hasStereoType` assigns zero or one primary-purpose classification to a story. The built-in individuals are `StoryStereoTypeQuery`, `StoryStereoTypeUpdate`, `StoryStereoTypeTransformationRule`, and `StoryStereoTypeValidationRule`. They are distinct individuals of this class. Applications can define more stereotypes using their own IRIs with labels and meaningful definitions. No stereotype is inferred when the relationship is absent.

Classify the story's primary purpose. A validation story may be implemented by a query or an API; a transformation story may use a query, pipeline, or rule engine. A stereotype does not select an implementation, infer executable source, or assign `isUpdate` or `isDeletion`. Those properties retain their separately documented meanings. The word "Rule" does not imply SHACL.

A stereotype is not a Story subclass. Multiple stories can share one stereotype without acquiring the same identity. `hasStereoType` is functional, and the four built-ins are asserted distinct. OWL can equate otherwise unconstrained custom stereotype IRIs to satisfy functionality; SHACL instead rejects multiple supplied terms. Validate the intended version context before identity reasoning.

`StoryClassificationShape` checks assignments, and `StoryStereoTypeShape` checks definitions. Both belong to the Story Ontology's shapes graph. Existing unclassified stories remain valid. When migrating application-defined classifications, map terms by confirmed meaning and replace the relationship and object IRIs explicitly. Do not infer purpose from labels, flags, or the language of an implementation.

SHACL-based implementation selection and rule execution are separate modeling work. The stereotype guide contains examples, validation, and migration guidance.

ekg-story:StoryStereoType	
https://ekgf.org/ontology/story#StoryStereoType	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Story Stereotype
owl:disjointWith	ekg-story:Story
owl:disjointWith	ekg-story:StoryConceptUsage
owl:disjointWith	ekg-story:StoryImplementation

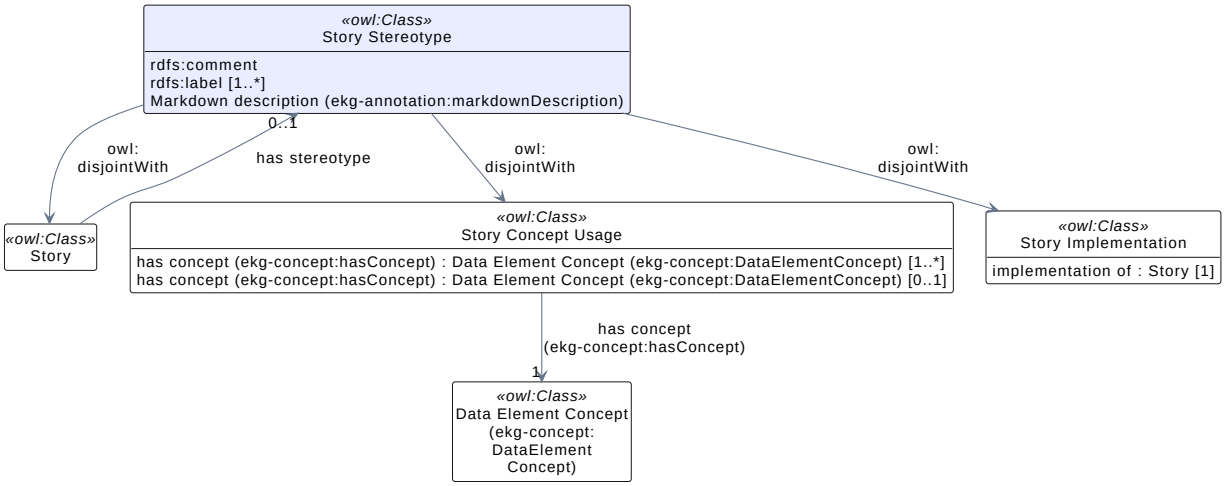


Diagram note

18.2.11 Workflow Implementation

A story implementation expressed as a workflow of execution steps.

ekg-story:WorkflowImplementation	
https://ekgf.org/ontology/story#WorkflowImplementation	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Workflow Implementation
rdfs:subClassOf	ekg-story:StoryImplementation

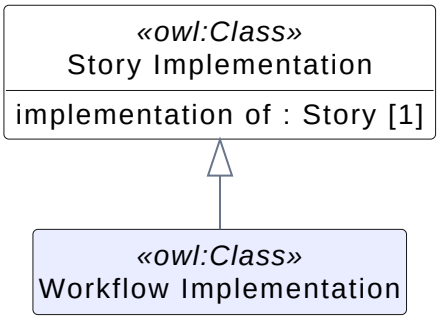


Diagram note

18.3 Properties

18.3.1 contributes to

The business outcome this story contributes to.

ekg-story:contributesTo	
https://ekgf.org/ontology/story#contributesTo	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-story:Story
rdfs:label	contributes to
rdfs:range	ekg-outcome:Outcome

18.3.2 expected usage role

Maps a `concept:StatementPosition` to the Story usage role that a sighting in that position implies for the sighted concept:

Position	Expected usage	Reason
concept:ParameterPosition	StoryInput	The invocation supplies the value.
concept:ProjectionPosition	StoryOutput	The story returns it as a result.
concept:GraphTemplate Position	StoryOutput	The story constructs or adds it.
concept:GraphPatternPosition	StoryDependent	The story only matches it against data.

The value is one of the three usage classes. The property is an annotation, so OWL does not treat those classes as individuals. An application-defined position compared with Story declarations must declare its expected role; `StatementPositionUsageRoleShape` rejects positions without exactly one.

The mapping classifies evidence. It never creates a usage or changes a declared one. `UndeclaredConceptUsageShape` compares the expectations with the story's declared usages.

ekg-story:expectedUsageRole	
https://ekgf.org/ontology/story#expectedUsageRole	
RDF predicate	**Value**
rdf:type	owl:AnnotationProperty
rdfs:label	expected usage role

18.3.3 has concept usage

Links a story to a concept-usage object owned by that story. Different stories use separate usage objects when referencing the same reusable concept.

ekg-story:hasConceptUsage	
https://ekgf.org/ontology/story#hasConceptUsage	
RDF predicate	**Value**
rdf:type	owl:InverseFunctionalProperty
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-story:Story

ekg-story:hasConceptUsage	
https://ekgf.org/ontology/story#hasConceptUsage	
RDF predicate	**Value**
rdfs:label	has concept usage
rdfs:range	ekg-story:StoryConceptUsage

18.3.4 has dependent concept

Links a story to a dependent-concept usage referencing a shared data element concept used internally rather than exposed as an input or output.

ekg-story:hasDependentConcept	
https://ekgf.org/ontology/story#hasDependentConcept	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	23 February 2026, 00:00:00 UTC
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-story:Story
rdfs:label	has dependent concept
rdfs:range	ekg-story:StoryDependent
rdfs:subPropertyOf	ekg-story:hasConceptUsage

18.3.5 has input concept

Links a story to its input-concept usage. The usage references a shared data element concept through concept:hasConcept and carries the story's input requirements.

ekg-story:hasInputConcept	
https://ekgf.org/ontology/story#hasInputConcept	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	23 February 2026, 00:00:00 UTC
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-story:Story
rdfs:label	has input concept
rdfs:range	ekg-story:StoryInput
rdfs:subPropertyOf	ekg-story:hasConceptUsage

18.3.6 has output concept

Links a story to an output-concept usage referencing the shared data element concept produced by the story.

ekg-story:hasOutputConcept	
https://ekgf.org/ontology/story#hasOutputConcept	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	23 February 2026, 00:00:00 UTC
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-story:Story
rdfs:label	has output concept
rdfs:range	ekg-story:StoryOutput
rdfs:subPropertyOf	ekg-story:hasConceptUsage

18.3.7 has stereotype

Assigns a story's optional primary-purpose classification. A story has at most one stereotype; no default is inferred. Stereotypes classify the story's intent independently of implementation language, mechanism, and mutation flags.

ekg-story:hasStereoType	
https://ekgf.org/ontology/story#hasStereoType	
RDF predicate	**Value**
rdf:type	owl:FunctionalProperty
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-story:Story
rdfs:label	has stereotype
rdfs:range	ekg-story:StoryStereoType

18.3.8 implementation of

Identifies the one story implemented by this resource. It is functional and the inverse of `implementedBy`. `StoryImplementationShape` counts owners from both directions, so duplicating an assertion in its inverse form still identifies one owner, while conflicting links are rejected.

ekg-story:implementationOf	
https://ekgf.org/ontology/story#implementationOf	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	23 February 2026, 00:00:00 UTC
rdf:type	owl:FunctionalProperty
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-story:StoryImplementation
rdfs:label	implementation of

ekg-story:implementationOf	
https://ekgf.org/ontology/story#implementationOf	
RDF predicate	**Value**
rdfs:range	ekg-story:Story
owl:inverseOf	ekg-story:implementedBy

18.3.9 Implementation Type

Coarse implementation type. Supported values: NamedStatement, ApiProxy, Workflow, AgentJob, Pipeline, UiAction.

ekg-story:implementationType	
https://ekgf.org/ontology/story#implementationType	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	23 February 2026, 00:00:00 UTC
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-story:StoryImplementation
rdfs:label	Implementation Type
rdfs:range	xsd:token

18.3.10 implemented by

Links a story to each of its implementations. Its inverse is `implementationOf`; asserting either direction lets an OWL reasoner derive the other. Several implementations can belong to the same story.

Execution selection belongs to the Story Service's execution context. This relationship records ownership of implementations. The retired `hasImplementation` predicate must be migrated to this predicate.

ekg-story:implementedBy	
https://ekgf.org/ontology/story#implementedBy	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-story:Story
rdfs:label	implemented by
rdfs:range	ekg-story:StoryImplementation

18.3.11 is deletion

The story deletes. It is invoked by DELETE, and refuses a GET; how a story mutates is a property of the story, never a subclass of it.

ekg-story:isDeletion	
https://ekgf.org/ontology/story#isDeletion	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-story:Story
rdfs:label	is deletion
rdfs:range	xsd:boolean

18.3.12 is mandatory

Whether this story's input-concept usage is required for execution. A Story Service interprets an absent value as false; OWL and SHACL validation do not insert a value.

ekg-story:isMandatory	
https://ekgf.org/ontology/story#isMandatory	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-story:StoryInput
rdfs:label	is mandatory
rdfs:range	xsd:boolean

18.3.13 Is Plural

Do not use, use isSingular

ekg-story:isPlural	
https://ekgf.org/ontology/story#isPlural	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	6 May 2020, 09:38:46 UTC
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-story:Story
rdfs:label	Is Plural
rdfs:range	xsd:boolean
owl:propertyDisjointWith	ekg-story:isSingular

18.3.14 is server provided

Whether the Story Service supplies this story's input concept. Independent of requiredness. A Story Service interprets an absent value as false; the flag does not specify how the value is generated or injected.

ekg-story:isServerProvided	
https://ekgf.org/ontology/story#isServerProvided	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-story:StoryInput
rdfs:label	is server provided
rdfs:range	xsd:boolean

18.3.15 Is Singular

Means that the user story results in a singular Thing as its output

ekg-story:isSingular	
https://ekgf.org/ontology/story#isSingular	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	6 May 2020, 09:37:47 UTC
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-story:Story
rdfs:label	Is Singular
rdfs:range	xsd:boolean

18.3.16 Is Update

Means that the user story mutates the database by insert or delete statements

ekg-story:isUpdate	
https://ekgf.org/ontology/story#isUpdate	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	6 May 2020, 09:37:47 UTC
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-story:Story
rdfs:label	Is Update
rdfs:range	xsd:boolean

18.3.17 not found on empty

A singular story that returns no rows answers 404 rather than an empty object.

ekg-story:notFoundOnEmpty	
https://ekgf.org/ontology/story#notFoundOnEmpty	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-story:Story
rdfs:label	not found on empty
rdfs:range	xsd:boolean

18.3.18 Played By

Identifies the persona playing the actor role in a story. The functional-property axiom limits a story to one persona through this property.

ekg-story:playedBy	
https://ekgf.org/ontology/story#playedBy	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	1 April 2020, 18:12:45 UTC
rdf:type	owl:FunctionalProperty
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-story:Story
rdfs:label	Played By
rdfs:range	ekg-persona:Persona

18.3.19 provided by

The provider a story is supplied by. No range is asserted: every assertion in the corpus points at a resource the corpus does not define, so the intended class is not evidenced.

ekg-story:providedBy	
https://ekgf.org/ontology/story#providedBy	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-story:Story
rdfs:label	provided by

18.4 Named Individuals

18.4.1 Query

Instance of: ekg-story:StoryStereoType

A story whose primary purpose is to retrieve or present information. This classification does not select a query language, implementation mechanism, or mutation-flag value.

ekg-story:StoryStereoTypeQuery	
https://ekgf.org/ontology/story#StoryStereoTypeQuery	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-story:StoryStereoType
rdfs:label	Query
owl:differentFrom	ekg-story:StoryStereoTypeTransformationRule
owl:differentFrom	ekg-story:StoryStereoTypeUpdate
owl:differentFrom	ekg-story:StoryStereoTypeValidationRule

18.4.2 Transformation Rule

Instance of: ekg-story:StoryStereoType

A story whose primary purpose is to derive or transform data according to a rule. This does not require SHACL, SPARQL, or any particular rule engine, and does not specify whether derived data is persisted.

ekg-story:StoryStereoTypeTransformationRule	
https://ekgf.org/ontology/story#StoryStereoTypeTransformationRule	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-story:StoryStereoType
rdfs:label	Transformation Rule
owl:differentFrom	ekg-story:StoryStereoTypeValidationRule

18.4.3 Update

Instance of: ekg-story:StoryStereoType

A story whose primary purpose is to apply an explicitly requested change to data. This describes the intended action, independently of the implementation technology and separately recorded mutation flags.

ekg-story:StoryStereoTypeUpdate	
https://ekgf.org/ontology/story#StoryStereoTypeUpdate	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-story:StoryStereoType
rdfs:label	Update
owl:differentFrom	ekg-story:StoryStereoTypeTransformationRule
owl:differentFrom	ekg-story:StoryStereoTypeValidationRule

18.4.4 Validation Rule

Instance of: ekg-story:StoryStereoType

A story whose primary purpose is to assess data against requirements and report conformance or violations. The classification does not determine whether the implementation uses SHACL constraints, a query, an API, or another mechanism.

ekg-story:StoryStereoTypeValidationRule	
https://ekgf.org/ontology/story#StoryStereoTypeValidationRule	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-story:StoryStereoType
rdfs:label	Validation Rule

18.5 Shapes

18.5.1 Legacy Concept Usage Shape

Rejects retired story input/output predicates, parameter classification classes, and the story-level server-provided flag. Data must migrate to typed concept usages and input flags.

ekg-story:LegacyConceptUsageShape	
https://ekgf.org/ontology/story#LegacyConceptUsageShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Legacy Concept Usage Shape
sh:not	(sh:or: ((sh:class: ekg-story:MandatoryParameter), (sh:class: ekg-story:OptionalParameter), (sh:class: ekg-story:OptionalStoryOutput)))
sh:property	(sh:maxCount: 0; sh:message: hasInput was removed. Use hasInputConcept to link to a StoryInput usage object.; sh:path: ekg-story:hasInput)
sh:property	(sh:maxCount: 0; sh:message: hasOutput was removed. Use hasOutputConcept to link to a StoryOutput usage object.; sh:path: ekg-story:hasOutput)
sh:property	(sh:maxCount: 0; sh:message: serverProvidedIRI was removed. Identify the input usage and set isServerProvided on that usage.; sh:path: ekg-story:serverProvidedIRI)
sh:targetClass	ekg-story:MandatoryParameter
sh:targetClass	ekg-story:OptionalParameter
sh:targetClass	ekg-story:OptionalStoryOutput
sh:targetSubjectsOf	ekg-story:hasInput
sh:targetSubjectsOf	ekg-story:hasOutput
sh:targetSubjectsOf	ekg-story:serverProvidedIRI

18.5.2 Legacy Implementation Shape

Rejects the retired hasImplementation predicate. Story implementation links use implementedBy and its inverse implementationOf.

ekg-story:LegacyImplementationShape	
https://ekgf.org/ontology/story#LegacyImplementationShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Legacy Implementation Shape
sh:property	(sh:maxCount: 0; sh:message: hasImplementation was removed. Use implementedBy or its inverse implementationOf.; sh:path: ekg-story:hasImplementation)
sh:targetSubjectsOf	ekg-story:hasImplementation

18.5.3 Statement Position Usage Role Shape

Targets each object of concept:statementPosition. The position must identify exactly one of StoryInput, StoryOutput or StoryDependent through expectedUsageRole. This prevents a missing mapping from silently disabling the undeclared-usage comparison, for example when the Story Ontology was not supplied as the ontology graph.

ekg-story:StatementPositionUsageRoleShape	
https://ekgf.org/ontology/story#StatementPositionUsageRoleShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Statement Position Usage Role Shape
sh:property	(sh:in: (ekg-story:StoryInput, ekg-story:StoryOutput, ekg-story:StoryDependent); sh:maxCount: 1; sh:message: A statement position compared with Story declarations must identify exactly one expected usage role: StoryInput, StoryOutput or StoryDependent. Supply the Story Ontology as the ontology graph for the built-in positions.; sh:minCount: 1; sh:path: ekg-story:expectedUsageRole)
sh:targetObjectsOf	ekg-concept:statementPosition

18.5.4 Story Classification Shape

Validates optional primary-purpose classification on stories, rejecting multiple stereotype terms, nonresource stories, and classification assigned to implementations, concept usages, or stereotype resources.

ekg-story:StoryClassificationShape	
https://ekgf.org/ontology/story#StoryClassificationShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Story Classification Shape
sh:nodeKind	sh:BlankNodeOrIRI

ekg-story:StoryClassificationShape	
https://ekgf.org/ontology/story#StoryClassificationShape	
RDF predicate	**Value**
sh:not	(sh:or: ((sh:class: ekg-story:StoryImplementation), (sh:class: ekg-story:StoryConceptUsage), (sh:class: ekg-story:StoryStereoType)))
sh:property	(sh:class: ekg-story:StoryStereoType; sh:maxCount: 1; sh:message: A story may have at most one primary-purpose stereotype, identified by an IRI.; sh:nodeKind: sh:IRI; sh:path: ekg-story:hasStereoType)
sh:targetClass	ekg-story:Story
sh:targetSubjectsOf	ekg-story:hasStereoType

18.5.5 Story Concept Usage Shape

Validates one owning story, one referenced concept, and exactly one input, output, or dependent role. Usage objects are distinct from stories, concepts, and manifestations.

ekg-story:StoryConceptUsageShape	
https://ekgf.org/ontology/story#StoryConceptUsageShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Story Concept Usage Shape
sh:nodeKind	sh:BlankNodeOrIRI
sh:not	(sh:or: ((sh:class: ekg-story:Story), (sh:class: ekg-concept:DataElementConcept), (sh:class: ekg-concept:ConceptManifestation)))
sh:property	(sh:class: ekg-story:Story; sh:maxCount: 1; sh:message: A concept usage must belong to exactly one story, counting general and specialized usage links together.; sh:minCount: 1; sh:nodeKind: sh:BlankNodeOrIRI; sh:path: (sh:inversePath: (sh:alternativePath: (ekg-story:hasConceptUsage, ekg-story:hasInputConcept, ekg-story:hasOutputConcept, ekg-story:hasDependentConcept))))
sh:property	(sh:class: ekg-concept:DataElementConcept; sh:maxCount: 1; sh:message: A concept usage must reference exactly one reusable data element concept through concept:hasConcept.; sh:minCount: 1; sh:nodeKind: sh:BlankNodeOrIRI; sh:path: ekg-concept:hasConcept)
sh:targetClass	ekg-story:StoryConceptUsage
sh:targetObjectsOf	ekg-story:hasConceptUsage
sh:targetObjectsOf	ekg-story:hasDependentConcept
sh:targetObjectsOf	ekg-story:hasInputConcept
sh:targetObjectsOf	ekg-story:hasOutputConcept
sh:targetSubjectsOf	ekg-story:isMandatory
sh:targetSubjectsOf	ekg-story:isServerProvided
sh:xone	((sh:class: ekg-story:StoryInput), (sh:class: ekg-story:StoryOutput), (sh:class: ekg-story:StoryDependent))

18.5.6 Story Implementation Shape

Validates exactly one owning story per implementation in the supplied version context. Counts both directions without requiring OWL inverse inference, including untyped relationship endpoints and implementation metadata.

ekg-story:StoryImplementationShape	
https://ekgf.org/ontology/story#StoryImplementationShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Story Implementation Shape
sh:nodeKind	sh:BlankNodeOrIRI
sh:property	(sh:class: ekg-story:Story; sh:maxCount: 1; sh:message: A story implementation must belong to exactly one story, counting both implementationOf and inverse implementedBy links.; sh:minCount: 1; sh:nodeKind: sh:BlankNodeOrIRI; sh:path: (sh:alternativePath: (ekg-story:implementationOf, (sh:inversePath: ekg-story:implementedBy))))
sh:targetClass	ekg-story:StoryImplementation
sh:targetObjectsOf	ekg-story:implementedBy
sh:targetSubjectsOf	ekg-story:implementationOf
sh:targetSubjectsOf	ekg-story:implementationType

18.5.7 Story Input Shape

Validates optional, single-valued boolean input flags and records their false application defaults without materializing them.

ekg-story:StoryInputShape	
https://ekgf.org/ontology/story#StoryInputShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Story Input Shape
sh:property	(sh:datatype: xsd:boolean; sh:defaultValue: false; sh:maxCount: 1; sh:name: Is mandatory; sh:path: ekg-story:isMandatory)
sh:property	(sh:datatype: xsd:boolean; sh:defaultValue: false; sh:maxCount: 1; sh:name: Is server provided; sh:path: ekg-story:isServerProvided)
sh:targetClass	ekg-story:StoryInput
sh:targetObjectsOf	ekg-story:hasInputConcept
sh:targetSubjectsOf	ekg-story:isMandatory
sh:targetSubjectsOf	ekg-story:isServerProvided

18.5.8 Story Stereotype Shape

Requires a stereotype IRI, a nonempty literal label, and a nonempty literal definition in rdfs:comment or markdownDescription. Covers built-in and application-defined stereotypes, including subclasses; the vocabulary is not a closed enumeration.

ekg-story:StoryStereoTypeShape	
https://ekgf.org/ontology/story#StoryStereoTypeShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Story Stereotype Shape
sh:nodeKind	sh:IRI
sh:not	(sh:or: ((sh:class: ekg-story:Story), (sh:class: ekg-story:StoryImplementation), (sh:class: ekg-story:StoryConceptUsage)))
sh:or	((sh:property: (sh:minCount: 1; sh:path: rdfs:comment)), (sh:property: (sh:minCount: 1; sh:path: ekg-annotation:markdownDescription)))
sh:property	(sh:minLength: 1; sh:nodeKind: sh:Literal; sh:path: rdfs:comment)
sh:property	(sh:minCount: 1; sh:minLength: 1; sh:nodeKind: sh:Literal; sh:path: rdfs:label)
sh:property	(sh:minLength: 1; sh:nodeKind: sh:Literal; sh:path: ekg-annotation:markdownDescription)
sh:targetClass	ekg-story:StoryStereoType
sh:targetObjectsOf	ekg-story:hasStereoType

18.5.9 Undeclared Concept Usage Shape

Compares sightings in a story implementation with the concept usages that its story declares, and reports gaps as **warnings**: quality findings for review, not structural violations. A sighting is in scope when it is `foundIn` the implementation and has a statement position. The implementation's story is found through `implementationOf` or inverse `implementedBy`. The sighted concept is found from the occurrence's manifestation, through `concept:hasManifestation` or `concept:concept`. With RDFS inference, preferred-manifestation links count as `hasManifestation`.

For each implementation and concept:

- A position expecting `StoryInput` requires an input usage of the concept.
- A position expecting `StoryOutput` requires an output usage of the concept.
- If the concept is sighted only in positions expecting `StoryDependent`, any declared usage of it satisfies the expectation. Graph patterns also use inputs and outputs. A concept with no declared usage is reported as an expected dependent.

Input or output evidence takes the place of the dependent expectation. Each missing role is reported once per implementation and concept, however many sightings support it. The result's value is the concept, and its message names the expected role.

The warnings rely on the harvested evidence. Missing sightings are not reported: harvest coverage, indirect use, template rendering and service-side input handling can all hide a use. Validate a coherent version context, with the Story Ontology as the ontology graph so that the position mappings are present. Allowing warnings, for example with `pyshacl --allow-warnings`, keeps these findings visible while only violations fail conformance. See the sightings guide for an executable example.

ekg-story:UndeclaredConceptUsageShape	
https://ekgf.org/ontology/story#UndeclaredConceptUsageShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Undeclared Concept Usage Shape
sh:severity	sh:Warning
sh:sparql	(sh:message: Used, not declared: concept {?value} was sighted in this implementation in a position expecting a {?role} usage, but the implemented story declares no such usage of it.; sh:select: SPARQL SELECT query 1)
sh:sparql	(sh:message: Used, not declared: concept {?value} was sighted in this implementation only in positions expecting a {?role} usage, and the implemented story declares no usage of it.; sh:select: SPARQL SELECT query 2)
sh:targetClass	ekg-story:StoryImplementation
sh:targetObjectsOf	ekg-story:implementedBy
sh:targetSubjectsOf	ekg-story:implementationOf

18.5.9.1 SPARQL query 1

Used, not declared: concept {?value} was sighted in this implementation in a position expecting a {?role} usage, but the implemented story declares no such usage of it.

```
PREFIX concept: <https://ekgf.org/ontology/concept#>
PREFIX story: <https://ekgf.org/ontology/story#>
SELECT DISTINCT $this ?value ?role WHERE {
    $this story:implementationOf|^story:implementedBy ?story .
    ?sighting concept:foundIn $this ;
        concept:statementPosition/story:expectedUsageRole ?role ;
        concept:occurrenceOf/(^concept:hasManifestation|concept:concept) ?value .
    FILTER (?role IN (story:StoryInput, story:StoryOutput))
    FILTER NOT EXISTS {
        ?story story:hasConceptUsage|story:hasInputConcept|story:hasOutputConcept|story:ha
sDependentConcept ?usage .
        ?usage concept:hasConcept ?value .
        { ?usage a ?role }
        UNION { ?story story:hasInputConcept ?usage . FILTER (?role = story:StoryInput) }
        UNION { ?story story:hasOutputConcept ?usage . FILTER (?role = story:StoryOutput)
}
    }
}
```

18.5.9.2 SPARQL query 2

Used, not declared: concept {?value} was sighted in this implementation only in positions expecting a {?role} usage, and the implemented story declares no usage of it.

```
PREFIX concept: <https://ekgf.org/ontology/concept#>
PREFIX story: <https://ekgf.org/ontology/story#>
SELECT DISTINCT $this ?value ?role WHERE {
    $this story:implementationOf|^story:implementedBy ?story .
    ?sighting concept:foundIn $this ;
        concept:statementPosition/story:expectedUsageRole story:StoryDependent ;
```

```
        concept:occurrenceOf/(^concept:hasManifestation|concept:concept) ?value .
    BIND (story:StoryDependent AS ?role)
    FILTER NOT EXISTS {
        ?evidence concept:foundIn $this ;
            concept:statementPosition/story:expectedUsageRole ?evidenceRole ;
            concept:occurrenceOf/(^concept:hasManifestation|concept:concept) ?value .
        FILTER (?evidenceRole != story:StoryDependent)
    }
    FILTER NOT EXISTS {
        ?story story:hasConceptUsage|story:hasInputConcept|story:hasOutputConcept|story:ha
sDependentConcept ?usage .
        ?usage concept:hasConcept ?value .
    }
}
```

19 SPARQL Story Implementation Ontology

19.1 Overview

SPARQL Story Implementation Ontology

Identifier	Value
Ontology IRI / Namespace	`https://ekgf.org/ontology/story-impl-sparql#`
Prefix	`ekg-story-impl-sparql`

A direct SPARQL implementation supplies executable query/update source with `sparql` or `fileName`. A `StorySPARQLTemplateImplementation` instead uses `hasTemplate` to identify a `StatementTemplate` and `usesRenderer` to identify the selected `TemplateRenderer`.

The template document supplies `templateText`, `templateFileName`, or both, and exactly one `concept:templateLanguage` profile. Its source is interpreted in that language. The renderer must explicitly declare matching `supportsTemplateLanguage` metadata. An execution service must also approve that profile and renderer; metadata alone does not certify safe execution.

Rendering supplies validated story input values to the template and produces a SPARQL statement for that invocation. This output is parsed and validated before execution. It is a different artifact from the reusable template document and is not stored in the template implementation's native-source fields. The implementation still belongs to exactly one story.

An input concept's placeholder is a `concept:TemplateParameterManifestation` with its own declared template language. A native query variable is a `concept:SPARQLVariableManifestation`. The source format and role determine the distinction: a dollar token can be native SPARQL syntax too, so a parser accepting unexpanded template text does not authorize executing it.

`StatementTemplateShape`, `TemplateRendererShape`, and `StorySPARQLTemplateImplementationShape` validate these relationships. Load the local Concept and Story shapes as well when validating the full model. All entry shapes belong to this ontology's shapes graph.

See the repository's template processing guide for input serialization requirements and an executable metadata example.

19.2 Classes

19.2.1 ASK

A SPARQL query story implementation using the ASK query form to produce a boolean result.

ekg-story-impl-sparql:ASK	
https://ekgf.org/ontology/story-impl-sparql#ASK	
RDF predicate	**Value**
`rdf:type`	`owl:Class`
`rdfs:label`	ASK
`rdfs:seeAlso`	`https://www.w3.org/TR/sparql11-query/#ask`
`rdfs:subClassOf`	ekg-story-impl-sparql:StorySPARQLQueryImplementation

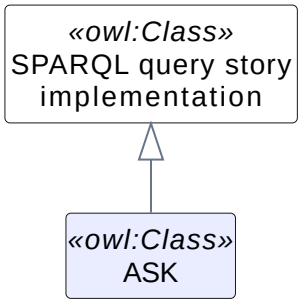


Diagram note

19.2.2 CONSTRUCT

A SPARQL query story implementation using the CONSTRUCT query form to produce an RDF graph.

ekg-story-impl-sparql:CONSTRUCT	
https://ekgf.org/ontology/story-impl-sparql#CONSTRUCT	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	CONSTRUCT
rdfs:seeAlso	https://www.w3.org/TR/sparql11-query/#construct
rdfs:subClassOf	ekg-story-impl-sparql:StorySPARQLQueryImplementation

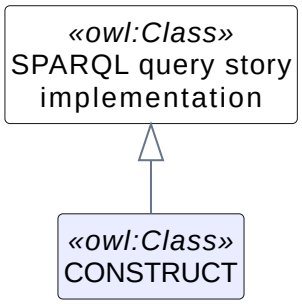


Diagram note

19.2.3 DELETE

A SPARQL update story implementation classified as deleting data.

ekg-story-impl-sparql:DELETE	
https://ekgf.org/ontology/story-impl-sparql#DELETE	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	DELETE
rdfs:seeAlso	https://www.w3.org/TR/sparql11-update/#deleteInsert

ekg-story-impl-sparql:DELETE	
https://ekgf.org/ontology/story-impl-sparql#DELETE	
RDF predicate	**Value**
rdfs:subClassOf	ekg-story-impl-sparql:StorySPARQLUpdateImplementation

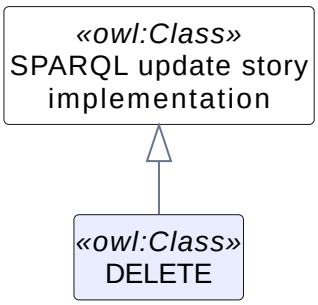


Diagram note

19.2.4 DESCRIBE

A SPARQL query story implementation using the DESCRIBE query form to obtain an RDF description of resources.

ekg-story-impl-sparql:DESCRIBE	
https://ekgf.org/ontology/story-impl-sparql#DESCRIBE	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	DESCRIBE
rdfs:seeAlso	https://www.w3.org/TR/sparql11-query/#describe
rdfs:subClassOf	ekg-story-impl-sparql:StorySPARQLQueryImplementation

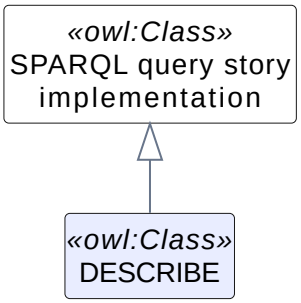


Diagram note

19.2.5 DROP

A SPARQL update story implementation classified as dropping a graph.

ekg-story-impl-sparql:DROP	
https://ekgf.org/ontology/story-impl-sparql#DROP	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	DROP
rdfs:seeAlso	https://www.w3.org/TR/sparql11-update/#drop
rdfs:subClassOf	ekg-story-impl-sparql:StorySPARQLUpdateImplementation

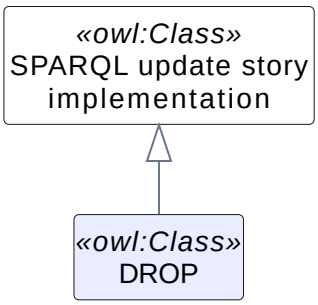


Diagram note

19.2.6 RDFox SPARQL flavor

The RDFox-specific SPARQL flavor represented in this vocabulary as a specialization of SPARQL11.

ekg-story-impl-sparql:RDFoxSPARQL	
https://ekgf.org/ontology/story-impl-sparql#RDFoxSPARQL	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	RDFox SPARQL flavor
rdfs:subClassOf	ekg-story-impl-sparql:SPARQL11

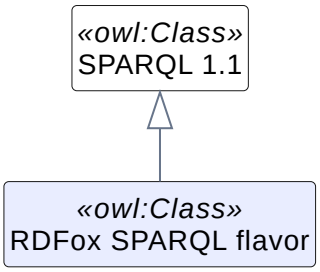


Diagram note

19.2.7 SELECT

A SPARQL query story implementation using the SELECT query form to return variable bindings.

ekg-story-impl-sparql:SELECT	
https://ekgf.org/ontology/story-impl-sparql#SELECT	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	SELECT
rdfs:seeAlso	https://www.w3.org/TR/sparql11-query/#select
rdfs:subClassOf	ekg-story-impl-sparql:StorySPARQLQueryImplementation

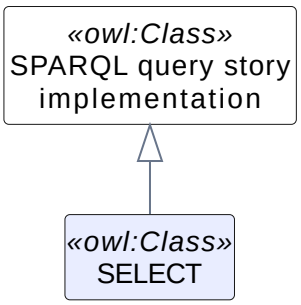


Diagram note

19.2.8 SPARQL 1.0

The SPARQL 1.0 flavor in the story implementation vocabulary.

ekg-story-impl-sparql:SPARQL10	
https://ekgf.org/ontology/story-impl-sparql#SPARQL10	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	SPARQL 1.0
rdfs:subClassOf	ekg-story-impl-sparql:SPARQLFlavor

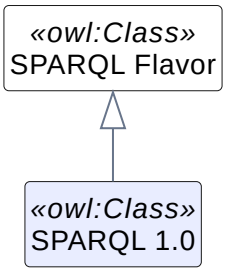


Diagram note

19.2.9 SPARQL 1.1

The SPARQL 1.1 flavor in the story implementation vocabulary.

ekg-story-impl-sparql:SPARQL11	
https://ekgf.org/ontology/story-impl-sparql#SPARQL11	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	SPARQL 1.1
rdfs:subClassOf	ekg-story-impl-sparql:SPARQLFlavor

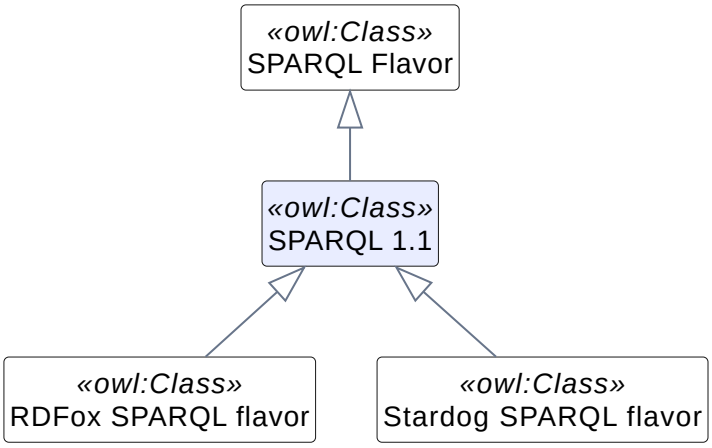


Diagram note

19.2.10 SPARQL database product

A database product entry used to identify a platform for SPARQL story implementations.

ekg-story-impl-sparql:SPARQLDatabaseProduct	
https://ekgf.org/ontology/story-impl-sparql#SPARQLDatabaseProduct	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	SPARQL database product

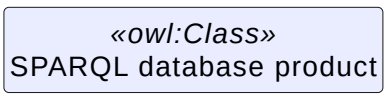


Diagram note

19.2.11 SPARQL database vendor

A vendor entry in the SPARQL database implementation vocabulary, distinct from a database product.

ekg-story-impl-sparql:SPARQLDatabaseVendor	
https://ekgf.org/ontology/story-impl-sparql#SPARQLDatabaseVendor	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	SPARQL database vendor

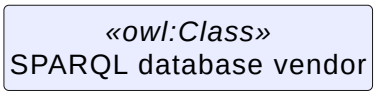


Diagram note

19.2.12 SPARQL Flavor

A SPARQL language or implementation-specific dialect used to classify story implementations.

ekg-story-impl-sparql:SPARQLFlavor	
https://ekgf.org/ontology/story-impl-sparql#SPARQLFlavor	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	SPARQL Flavor



Diagram note

19.2.13 Stardog SPARQL flavor

The Stardog-specific SPARQL flavor represented in this vocabulary as a specialization of SPARQL11.

ekg-story-impl-sparql:StardogSPARQL	
https://ekgf.org/ontology/story-impl-sparql#StardogSPARQL	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Stardog SPARQL flavor
rdfs:subClassOf	ekg-story-impl-sparql:SPARQL11

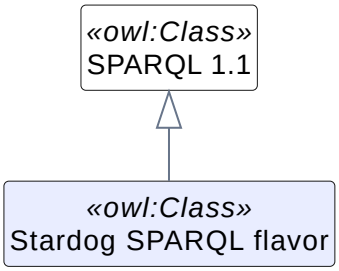


Diagram note

19.2.14 Statement Template

A document in one declared template language/profile, stored inline or in a file, that a renderer turns into a SPARQL query or update. It is template source even when its unexpanded text happens to parse as SPARQL.

ekg-story-impl-sparql:StatementTemplate	
https://ekgf.org/ontology/story-impl-sparql#StatementTemplate	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Statement Template
rdfs:subClassOf	(rdf:type: owl:Restriction; owl:minCardinality: 1; owl:onProperty: ekg-concept:templateLanguage)
rdfs:subClassOf	(rdf:type: owl:Restriction; owl:maxCardinality: 1; owl:onProperty: ekg-concept:templateLanguage)

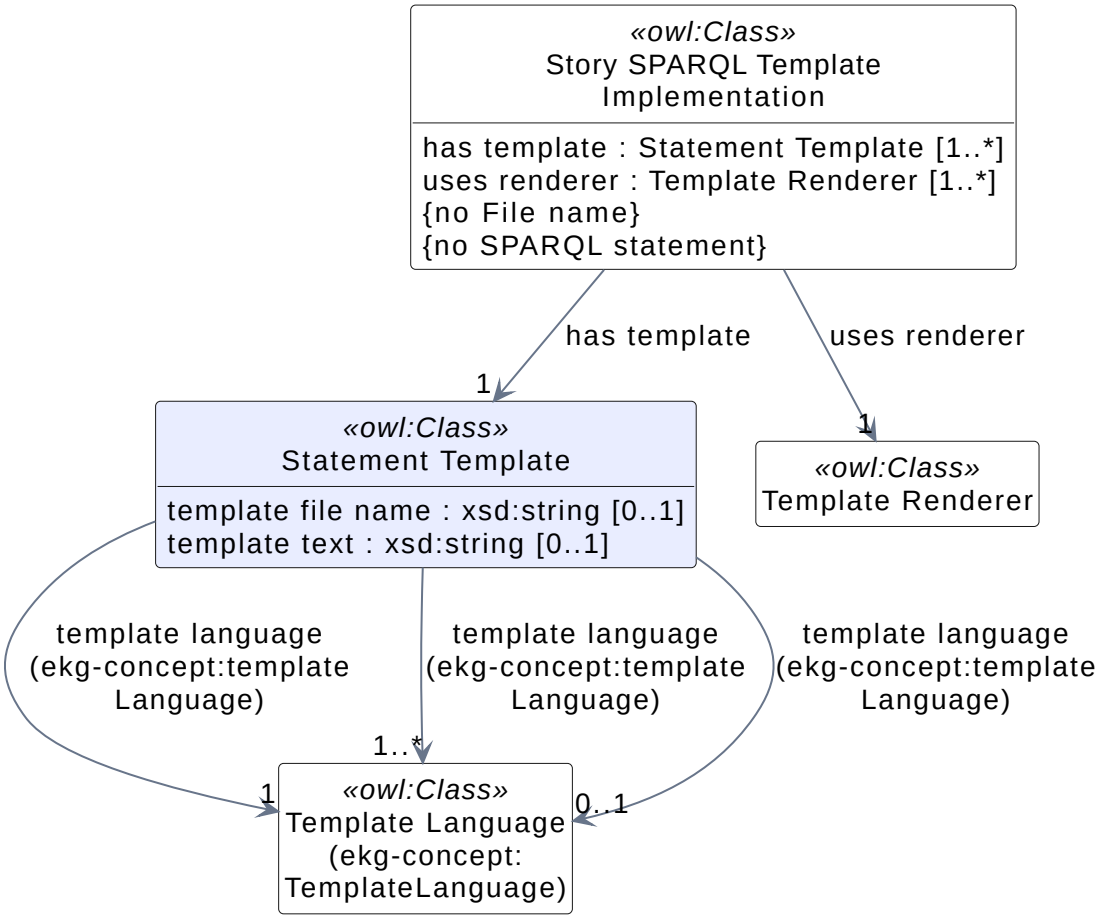


Diagram note

19.2.15 SPARQL story implementation

A story implementation expressed using a SPARQL statement, with optional file and language-flavor metadata.

ekg-story-impl-sparql:StorySPARQLImplementation	
https://ekgf.org/ontology/story-impl-sparql#StorySPARQLImplementation	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	SPARQL story implementation
rdfs:subClassOf	ekg-story:StoryImplementation

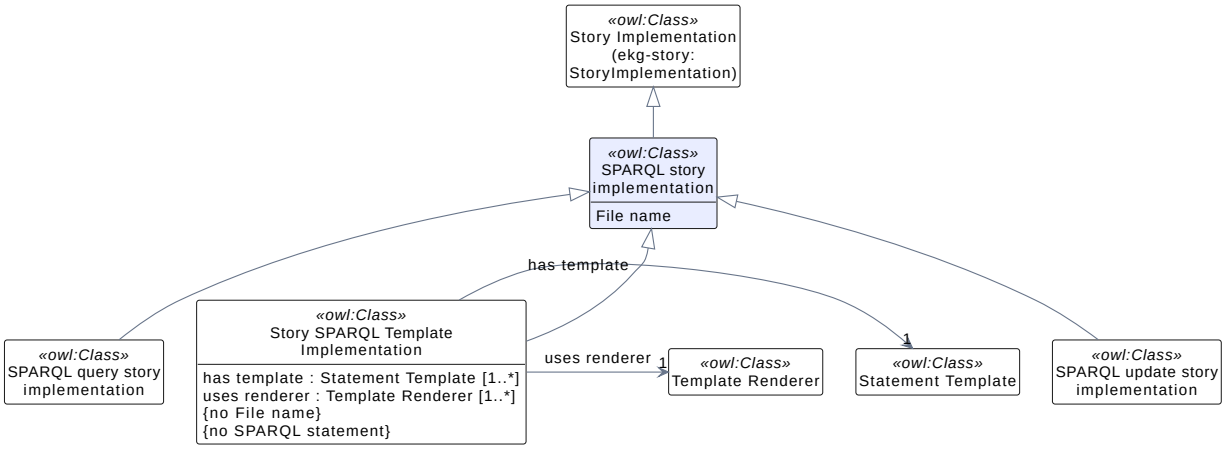


Diagram note

19.2.16 SPARQL query story implementation

Story SPARQL Implementation for query statements like `SELECT`, `CONSTRUCT`, `ASK` and `DESCRIBE`

ekg-story-impl-sparql:StorySPARQLQueryImplementation	
https://ekgf.org/ontology/story-impl-sparql#StorySPARQLQueryImplementation	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	SPARQL query story implementation
rdfs:subClassOf	ekg-story-impl-sparql:StorySPARQLImplementation

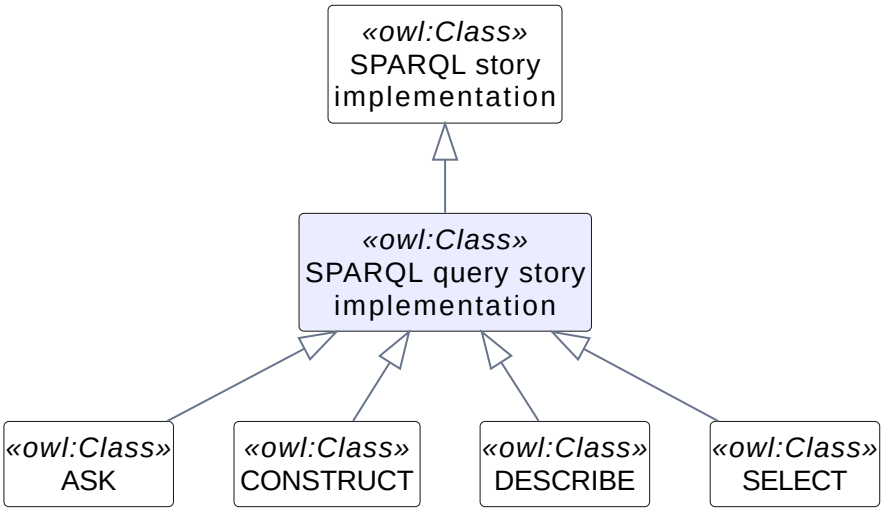


Diagram note

19.2.17 Story SPARQL Template Implementation

A story implementation that produces executable SPARQL by rendering exactly one template document with exactly one selected renderer. Its source is a document in a template language; the generated statement is SPARQL.

ekg-story-impl-sparql:StorySPARQLTemplateImplementation	
https://ekgf.org/ontology/story-impl-sparql#StorySPARQLTemplateImplementation	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Story SPARQL Template Implementation
rdfs:subClassOf	ekg-story-impl-sparql:StorySPARQLImplementation
rdfs:subClassOf	(rdf:type: owl:Restriction; owl:minCardinality: 1; owl:onProperty: ekg-story-impl-sparql:hasTemplate)
rdfs:subClassOf	(rdf:type: owl:Restriction; owl:minCardinality: 1; owl:onProperty: ekg-story-impl-sparql:usesRenderer)

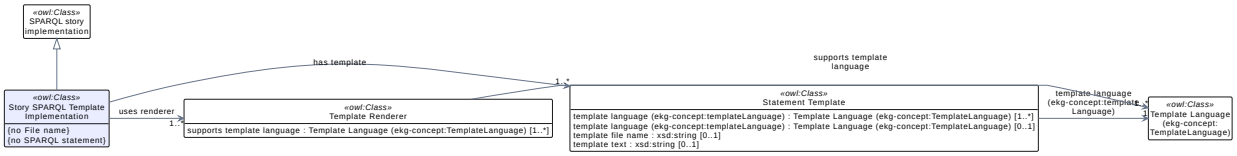


Diagram note

19.2.18 SPARQL update story implementation

Story SPARQL Implementation for update statements like `INSERT/DELETE/WHERE` etc

ekg-story-impl-sparql:StorySPARQLUpdateImplementation	
https://ekgf.org/ontology/story-impl-sparql#StorySPARQLUpdateImplementation	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	SPARQL update story implementation
rdfs:subClassOf	ekg-story-impl-sparql:StorySPARQLImplementation

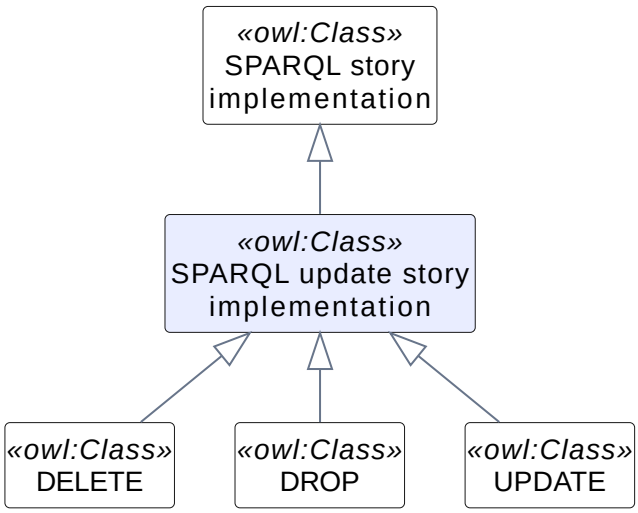


Diagram note

19.2.19 Template Renderer

A software component that renders statement-template documents using story input values and explicitly declares the template-language profiles it supports. The execution service is responsible for safe typed substitution and validation of generated SPARQL.

ekg-story-impl-sparql:TemplateRenderer	
https://ekgf.org/ontology/story-impl-sparql#TemplateRenderer	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Template Renderer
rdfs:subClassOf	(rdf:type: owl:Restriction; owl:minCardinality: 1; owl:onProperty: ekg-story-impl-sparql:supportsTemplateLanguage)

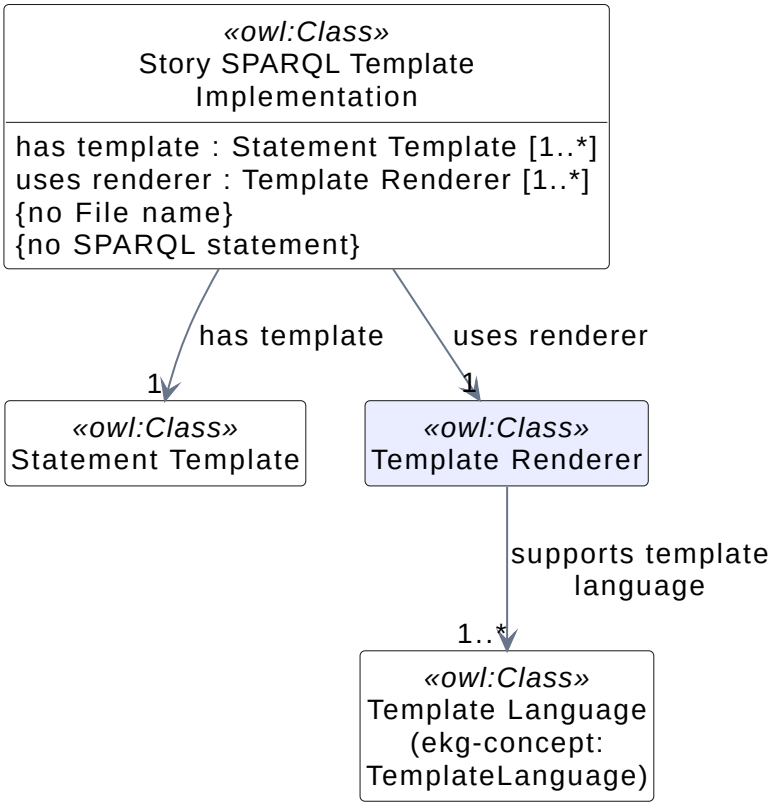


Diagram note

19.2.20 UPDATE

A story implementation classified as a SPARQL update. This class does not distinguish the individual SPARQL update forms.

ekg-story-impl-sparql:UPDATE	
https://ekgf.org/ontology/story-impl-sparql#UPDATE	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	UPDATE
rdfs:seeAlso	https://www.w3.org/TR/sparql11-update/
rdfs:subClassOf	ekg-story-impl-sparql:StorySPARQLUpdateImplementation

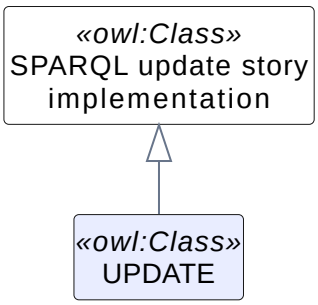


Diagram note

19.3 Properties

19.3.1 File name

The name of the file name that contains the SPARQL implementation of the given Story. This name is assumed to be relative to the directory where the Turtle file containing the StoryImplementation lives.

ekg-story-impl-sparql:fileName	
https://ekgf.org/ontology/story-impl-sparql#fileName	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-story-impl-sparql:StorySPARQLImplementation
rdfs:label	File name

19.3.2 Supports SPARQL Flavor; Supports SPARQL Flavour

Identifies the SPARQL language or implementation-specific flavor used by a SPARQL story implementation.

ekg-story-impl-sparql:flavor	
https://ekgf.org/ontology/story-impl-sparql#flavor	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-story-impl-sparql:StorySPARQLImplementation
rdfs:label	Supports SPARQL Flavor

ekg-story-impl-sparql:flavor	
https://ekgf.org/ontology/story-impl-sparql#flavor	
RDF predicate	**Value**
rdfs:label	Supports SPARQL Flavour
rdfs:range	ekg-story-impl-sparql:SPARQLFlavor

19.3.3 has template

Links a SPARQL-producing story implementation to its template document. The source language is declared on that document; its text is not executable SPARQL source.

ekg-story-impl-sparql:hasTemplate	
https://ekgf.org/ontology/story-impl-sparql#hasTemplate	
RDF predicate	**Value**
rdf:type	owl:FunctionalProperty
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-story-impl-sparql:StorySPARQLTemplateImplementation
rdfs:label	has template
rdfs:range	ekg-story-impl-sparql:StatementTemplate

19.3.4 SPARQL statement

Executable SPARQL query or update text. Template source belongs to a StatementTemplate linked through hasTemplate and must be rendered before the resulting SPARQL is parsed and executed.

ekg-story-impl-sparql:sparql	
https://ekgf.org/ontology/story-impl-sparql#sparql	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-story-impl-sparql:StorySPARQLImplementation
rdfs:label	SPARQL statement

19.3.5 supports template language

Records a documented template-language profile implemented by a renderer. Support is an explicit capability assertion, not proof of safe RDF serialization or approval to execute.

ekg-story-impl-sparql:supportsTemplateLanguage	
https://ekgf.org/ontology/story-impl-sparql#supportsTemplateLanguage	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-story-impl-sparql:TemplateRenderer

ekg-story-impl-sparql:supportsTemplateLanguage	
https://ekgf.org/ontology/story-impl-sparql#supportsTemplateLanguage	
RDF predicate	**Value**
rdfs:label	supports template language
rdfs:range	ekg-concept:TemplateLanguage

19.3.6 template file name

The file containing a template document, relative to the document's source context. It identifies template source rather than a file of executable SPARQL.

ekg-story-impl-sparql:templateFileName	
https://ekgf.org/ontology/story-impl-sparql#templateFileName	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-story-impl-sparql:StatementTemplate
rdfs:label	template file name
rdfs:range	xsd:string

19.3.7 template text

The unrendered template document's source text, interpreted in its declared template language. Parsing this text as SPARQL does not replace template rendering.

ekg-story-impl-sparql:templateText	
https://ekgf.org/ontology/story-impl-sparql#templateText	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-story-impl-sparql:StatementTemplate
rdfs:label	template text
rdfs:range	xsd:string

19.3.8 uses renderer

Identifies the renderer selected for this template-based implementation. It must explicitly support the template document's declared language/profile.

ekg-story-impl-sparql:usesRenderer	
https://ekgf.org/ontology/story-impl-sparql#usesRenderer	
RDF predicate	**Value**
rdf:type	owl:FunctionalProperty

ekg-story-impl-sparql:usesRenderer	
https://ekgf.org/ontology/story-impl-sparql#usesRenderer	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-story-impl-sparql:StorySPARQLTemplateImplementation
rdfs:label	uses renderer
rdfs:range	ekg-story-impl-sparql:TemplateRenderer

19.4 Named Individuals

19.4.1 Allegro Graph

Instance of: ekg-story-impl-sparql:SPARQLDatabaseProduct

The AllegroGraph product entry used to identify a database platform for SPARQL story implementations.

ekg-story-impl-sparql:AllegroGraph	
https://ekgf.org/ontology/story-impl-sparql#AllegroGraph	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-story-impl-sparql:SPARQLDatabaseProduct
rdfs:label	Allegro Graph

19.4.2 Amazon Web Services

Instance of: ekg-story-impl-sparql:SPARQLDatabaseVendor

The vendor entry named Amazon Web Services in the SPARQL database implementation vocabulary.

ekg-story-impl-sparql:AmazonWebServicesInc	
https://ekgf.org/ontology/story-impl-sparql#AmazonWebServicesInc	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-story-impl-sparql:SPARQLDatabaseVendor
rdfs:label	Amazon Web Services

19.4.3 Anzo Graph

Instance of: ekg-story-impl-sparql:SPARQLDatabaseProduct

The AnzoGraph product entry used to identify a database platform for SPARQL story implementations.

ekg-story-impl-sparql:AnzoGraph	
https://ekgf.org/ontology/story-impl-sparql#AnzoGraph	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-story-impl-sparql:SPARQLDatabaseProduct
rdfs:label	Anzo Graph

19.4.4 Blazegraph

Instance of: ekg-story-impl-sparql:SPARQLDatabaseProduct

The Blazegraph product entry used to identify a database platform for SPARQL story implementations.

ekg-story-impl-sparql:BlazeGraph	
https://ekgf.org/ontology/story-impl-sparql#BlazeGraph	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-story-impl-sparql:SPARQLDatabaseProduct
rdfs:label	Blazegraph

19.4.5 Cambridge Semantics

Instance of: ekg-story-impl-sparql:SPARQLDatabaseVendor

The vendor entry named Cambridge Semantics in the SPARQL database implementation vocabulary.

ekg-story-impl-sparql:CambridgeSemantics	
https://ekgf.org/ontology/story-impl-sparql#CambridgeSemantics	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-story-impl-sparql:SPARQLDatabaseVendor
rdfs:label	Cambridge Semantics

19.4.6 GraphDB

Instance of: ekg-story-impl-sparql:SPARQLDatabaseProduct

The GraphDB product entry used to identify a database platform for SPARQL story implementations.

ekg-story-impl-sparql:GraphDb	
https://ekgf.org/ontology/story-impl-sparql#GraphDb	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-story-impl-sparql:SPARQLDatabaseProduct
rdfs:label	GraphDB

19.4.7 Neptune

Instance of: ekg-story-impl-sparql:SPARQLDatabaseProduct

The Neptune product entry used to identify a database platform for SPARQL story implementations.

ekg-story-impl-sparql:Neptune	
https://ekgf.org/ontology/story-impl-sparql#Neptune	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-story-impl-sparql:SPARQLDatabaseProduct
rdfs:label	Neptune

19.4.8 Ontotext

Instance of: ekg-story-impl-sparql:SPARQLDatabaseVendor

The vendor entry named Ontotext in the SPARQL database implementation vocabulary.

ekg-story-impl-sparql:Ontotext	
https://ekgf.org/ontology/story-impl-sparql#Ontotext	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-story-impl-sparql:SPARQLDatabaseVendor
rdfs:label	Ontotext

19.4.9 Oxford Semantic Technologies

Instance of: ekg-story-impl-sparql:SPARQLDatabaseVendor

The vendor entry named Oxford Semantic Technologies in the SPARQL database implementation vocabulary.

ekg-story-impl-sparql:OxfordSemanticTechnologies	
https://ekgf.org/ontology/story-impl-sparql#OxfordSemanticTechnologies	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-story-impl-sparql:SPARQLDatabaseVendor
rdfs:label	Oxford Semantic Technologies

19.4.10 RD Fox

Instance of: ekg-story-impl-sparql:SPARQLDatabaseProduct

The RDFox product entry used to identify a database platform for SPARQL story implementations.

ekg-story-impl-sparql:RDFox	
https://ekgf.org/ontology/story-impl-sparql#RDFox	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-story-impl-sparql:SPARQLDatabaseProduct
rdfs:label	RD Fox

19.4.11 Stardog

Instance of: ekg-story-impl-sparql:SPARQLDatabaseProduct

The Stardog product entry used to identify a database platform for SPARQL story implementations.

ekg-story-impl-sparql:Stardog	
https://ekgf.org/ontology/story-impl-sparql#Stardog	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-story-impl-sparql:SPARQLDatabaseProduct
rdfs:label	Stardog

19.4.12 Stardog Union

Instance of: ekg-story-impl-sparql:SPARQLDatabaseVendor

The vendor entry named Stardog Union in the SPARQL database implementation vocabulary.

ekg-story-impl-sparql:StardogUnion	
https://ekgf.org/ontology/story-impl-sparql#StardogUnion	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-story-impl-sparql:SPARQLDatabaseVendor
rdfs:label	Stardog Union

19.4.13 Virtuoso

Instance of: ekg-story-impl-sparql:SPARQLDatabaseProduct

The Virtuoso product entry used to identify a database platform for SPARQL story implementations.

ekg-story-impl-sparql:Virtuoso	
https://ekgf.org/ontology/story-impl-sparql#Virtuoso	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-story-impl-sparql:SPARQLDatabaseProduct
rdfs:label	Virtuoso

19.5 Shapes

19.5.1 Statement Template Shape

Requires one documented template-language profile and nonempty inline or file-based template source. Source content and its file location may both be supplied.

ekg-story-impl-sparql:StatementTemplateShape	
https://ekgf.org/ontology/story-impl-sparql#StatementTemplateShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Statement Template Shape
sh:nodeKind	sh:BlankNodeOrIRI
sh:or	((sh:property: (sh:minCount: 1; sh:path: ekg-story-impl-sparql:templateText)), (sh:property: (sh:minCount: 1; sh:path: ekg-story-impl-sparql:templateFileName)))
sh:property	(sh:datatype: xsd:string; sh:maxCount: 1; sh:minLength: 1; sh:path: ekg-story-impl-sparql:templateFileName)
sh:property	(sh:class: ekg-concept:TemplateLanguage; sh:maxCount: 1; sh:minCount: 1; sh:nodeKind: sh:IRI; sh:path: ekg-concept:templateLanguage)
sh:property	(sh:datatype: xsd:string; sh:maxCount: 1; sh:minLength: 1; sh:path: ekg-story-impl-sparql:templateText)
sh:targetClass	ekg-story-impl-sparql:StatementTemplate
sh:targetObjectsOf	ekg-story-impl-sparql:hasTemplate
sh:targetSubjectsOf	ekg-story-impl-sparql:templateFileName
sh:targetSubjectsOf	ekg-story-impl-sparql:templateText

19.5.2 Story SPARQL Template Implementation Shape

Requires one template and one renderer with matching language support, and rejects mixing template source with executable-SPARQL source fields.

ekg-story-impl-sparql:StorySPARQLTemplateImplementationShape	
https://ekgf.org/ontology/story-impl-sparql#StorySPARQLTemplateImplementationShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Story SPARQL Template Implementation Shape
sh:nodeKind	sh:BlankNodeOrIRI
sh:property	(sh:maxCount: 0; sh:message: A template implementation supplies source through hasTemplate; executable SPARQL is produced during rendering.; sh:path: ekg-story-impl-sparql:sparql)
sh:property	(sh:maxCount: 0; sh:message: A template source file belongs to the linked StatementTemplate through templateFileName.; sh:path: ekg-story-impl-sparql:fileName)

ekg-story-impl-sparql:StorySPARQLTemplateImplementationShape	
https://ekgf.org/ontology/story-impl-sparql#StorySPARQLTemplateImplementationShape	
RDF predicate	**Value**
sh:property	(sh:class: ekg-story-impl-sparql:StatementTemplate; sh:maxCount: 1; sh:minCount: 1; sh:nodeKind: sh:BlankNodeOrIRI; sh:path: ekg-story-impl-sparql:hasTemplate)
sh:property	(sh:class: ekg-story-impl-sparql:TemplateRenderer; sh:maxCount: 1; sh:minCount: 1; sh:nodeKind: sh:BlankNodeOrIRI; sh:path: ekg-story-impl-sparql:usesRenderer)
sh:sparql	(sh:message: The selected renderer must explicitly support the template document's declared language/profile.; sh:select: SPARQL SELECT query 1)
sh:targetClass	ekg-story-impl-sparql:StorySPARQLTemplateImplementation
sh:targetSubjectsOf	ekg-story-impl-sparql:hasTemplate
sh:targetSubjectsOf	ekg-story-impl-sparql:usesRenderer

19.5.2.1 SPARQL query 1

The selected renderer must explicitly support the template document's declared language/profile.

```
SELECT $this WHERE {
    $this <https://ekgf.org/ontology/story-impl-sparql#hasTemplate> ?template ;
          <https://ekgf.org/ontology/story-impl-sparql#usesRenderer> ?renderer .
    ?template <https://ekgf.org/ontology/concept#templateLanguage> ?language .
    FILTER NOT EXISTS {
        ?renderer <https://ekgf.org/ontology/story-impl-sparql#supportsTemplateLanguage> ?
language .
    }
}
```

19.5.3 Template Renderer Shape

Requires a renderer to explicitly identify the template-language profiles it supports. Execution approval and safe substitution remain application responsibilities.

ekg-story-impl-sparql:TemplateRendererShape	
https://ekgf.org/ontology/story-impl-sparql#TemplateRendererShape	
RDF predicate	**Value**
rdf:type	sh:NodeShape
rdfs:label	Template Renderer Shape
sh:nodeKind	sh:BlankNodeOrIRI
sh:property	(sh:class: ekg-concept:TemplateLanguage; sh:minCount: 1; sh:nodeKind: sh:IRI; sh:path: ekg-story-impl-sparql:supportsTemplateLanguage)
sh:targetClass	ekg-story-impl-sparql:TemplateRenderer
sh:targetObjectsOf	ekg-story-impl-sparql:usesRenderer
sh:targetSubjectsOf	ekg-story-impl-sparql:supportsTemplateLanguage

20 Story Service Ontology

20.1 Overview

Story Service Ontology

Identifier	Value
Ontology IRI / Namespace	`https://ekgf.org/ontology/story-service#`
Prefix	`ekg-story-service`

A Story Service is an application that executes stories through their implementations. The story describes business concept usage; an implementation binds that usage to technical manifestations such as API names or SPARQL variables. Reusing a concept preserves its meaning across implementations.

20.1.1 Input-concept defaults

The story model gives each story its own `StoryInput` usage objects, referencing shared data element concepts. On each input usage:

Flag	Meaning when true	Effective value when absent
`ekg-story:isMandatory`	The input is required for execution.	false
`ekg-story:isServerProvided`	The Story Service supplies the input.	false

The flags are independent. An input can be required for execution and supplied by the service. Explicit values take precedence over the declared defaults. The service must reject invalid or conflicting values.

Missing flag assertions remain valid. A Story Service applies the false defaults when executing a story. Storing explicit false triples is optional; consumers must not assume that loading or validating the RDF populated them.

The Story Ontology's SHACL shapes declare `sh:defaultValue false` and validate supplied boolean values with a maximum count of one. SHACL Core default-value annotations do not populate data, and OWL does not infer false from an absent assertion. See SHACL default values.

20.1.2 Business concepts and technical bindings

The Story Ontology declares the common usage class, specialized story predicates, input flags, and validation shapes. Its input flags describe a particular story's use of a shared concept. This source's `Parameter`, `Result`, and statement properties describe a technical statement interface; they do not replace story concept usages or establish a mapping to a particular execution binding. Variable manifestations record technical expressions independently of the mechanism used to supply their values.

For a statement template, a template processor renders the source using the story's validated input values before parsing and executing the resulting SPARQL query or update. Support must be limited to explicitly documented language profiles. Value substitution requires RDF-aware serialization and escaping; shell evaluation and arbitrary query-fragment insertion are not input-value substitution. Unresolved placeholders and invalid generated statements are execution errors. A template-placeholder manifestation does not select or certify a template engine.

The SPARQL Story Implementation Ontology declares `StorySPARQLTemplateImplementation`, which selects a template document and renderer through `hasTemplate` and `usesRenderer`. The document's `concept:templateLanguage` identifies its source language/profile, and the renderer declares matching `supportsTemplateLanguage` capabilities. The placeholder manifestation also identifies its template lan-

guage. The service renders that document into a separate SPARQL statement for each invocation; the generated text is parsed before it is sent for execution.

20.2 Classes

20.2.1 Parameter

A parameter description used by an executable statement in the story service. Parameters are distinct from statements and service instances.

ekg-story-service:Parameter	
https://ekgf.org/ontology/story-service#Parameter	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Parameter
owl:disjointWith	ekg-story-service:Statement
owl:disjointWith	ekg-story-service:StoryService

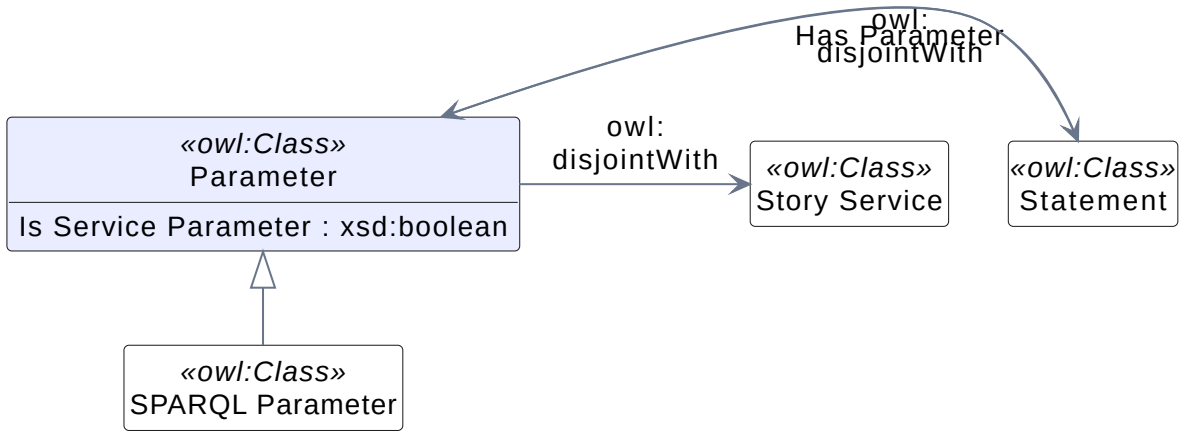


Diagram note

20.2.2 Result

A description of an output supplied by a story-service execution.

ekg-story-service:Result	
https://ekgf.org/ontology/story-service#Result	
RDF predicate	**Value**
rdf:type	owl:Class
rdfs:label	Result

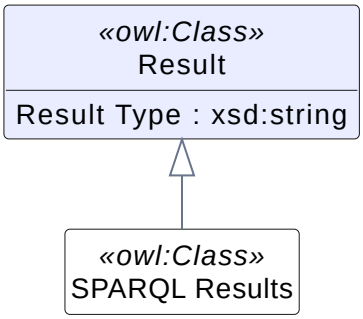


Diagram note

20.2.3 SHACL Statement

A story-service statement expressed using SHACL.

ekg-story-service:SHACLStatement	
https://ekgf.org/ontology/story-service#SHACLStatement	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	1 April 2020, 18:14:13 UTC
rdf:type	owl:Class
rdfs:label	SHACL Statement
rdfs:subClassOf	ekg-story-service:Statement

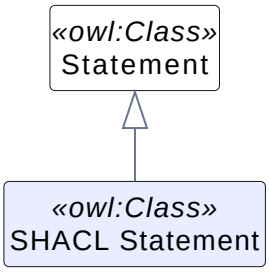


Diagram note

20.2.4 SPARQL Parameter

A statement parameter used by a SPARQL story implementation.

ekg-story-service:SPARQLParameter	
https://ekgf.org/ontology/story-service#SPARQLParameter	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	1 April 2020, 17:25:05 UTC
rdf:type	owl:Class

ekg-story-service:SPARQLParameter	
https://ekgf.org/ontology/story-service#SPARQLParameter	
RDF predicate	**Value**
rdfs:label	SPARQL Parameter
rdfs:subClassOf	ekg-story-service:Parameter

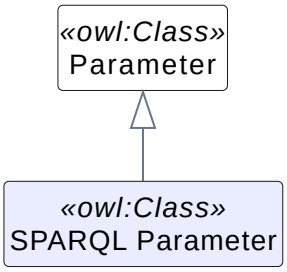


Diagram note

20.2.5 SPARQL Results

A story-service result description for the output of a SPARQL execution.

ekg-story-service:SPARQLResults	
https://ekgf.org/ontology/story-service#SPARQLResults	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	7 April 2020, 20:40:18 UTC
rdf:type	owl:Class
rdfs:label	SPARQL Results
rdfs:subClassOf	ekg-story-service:Result

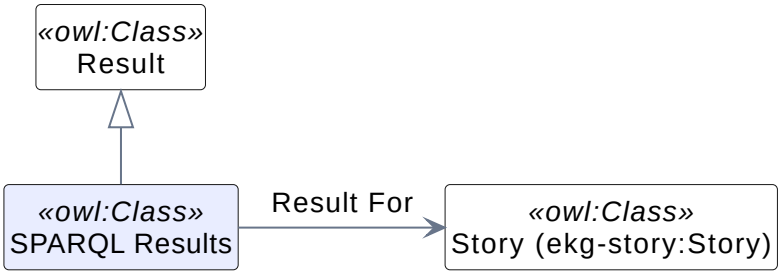


Diagram note

20.2.6 SPARQL Statement

A story-service statement expressed using SPARQL.

ekg-story-service:SPARQLStatement	
https://ekgf.org/ontology/story-service#SPARQLStatement	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	1 April 2020, 17:24:10 UTC
rdf:type	owl:Class
rdfs:label	SPARQL Statement
rdfs:subClassOf	ekg-story-service:Statement

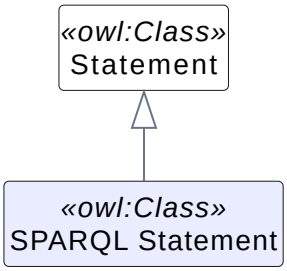


Diagram note

20.2.7 Statement

An executable statement used by the story service to implement a story, such as a SPARQL or SHACL statement. It is distinct from the service that executes it.

ekg-story-service:Statement	
https://ekgf.org/ontology/story-service#Statement	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	1 April 2020, 17:24:43 UTC
rdf:type	owl:Class
rdfs:label	Statement
owl:disjointWith	ekg-story-service:StoryService

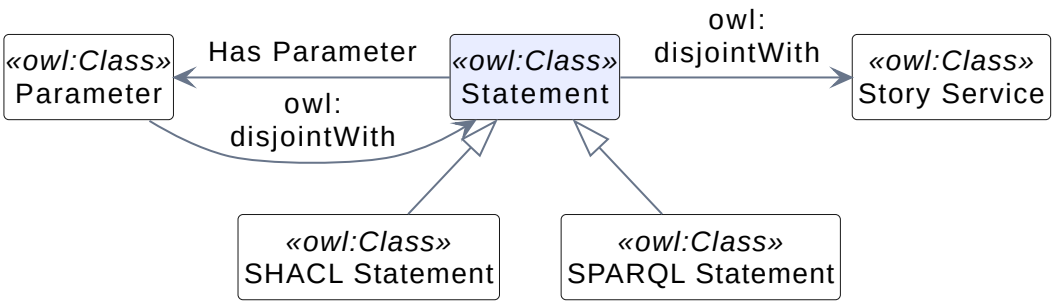


Diagram note

20.2.8 Story Service

A service application executes stories and applies their input-concept requirements. Under the agreed contract, it interprets absent `ekg-story:isMandatory` and `ekg-story:isServerProvided` flags as false on the story's input usages. The ontology's Markdown description records the full contract and links to the declared Story vocabulary.

ekg-story-service:StoryService	
https://ekgf.org/ontology/story-service#StoryService	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	1 April 2020, 17:28:32 UTC
rdf:type	owl:Class
rdfs:label	Story Service

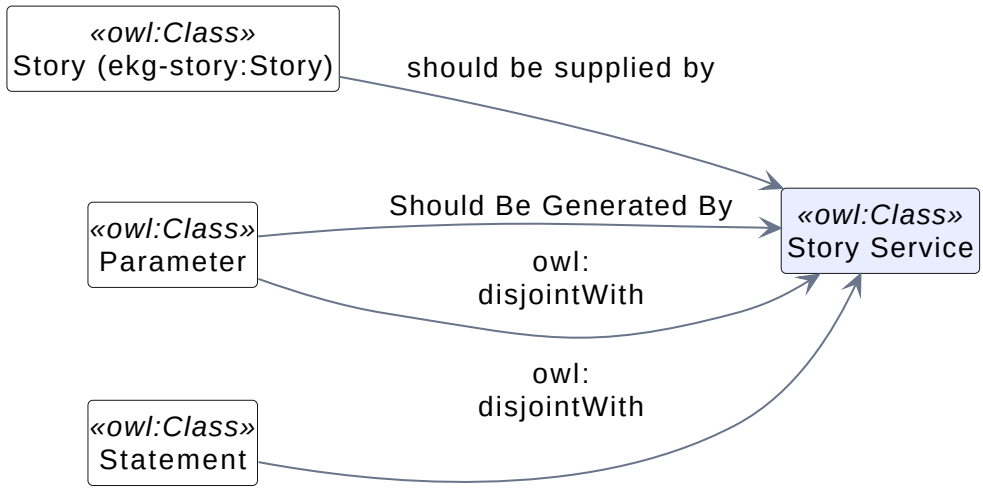


Diagram note

20.3 Properties

20.3.1 Has Parameter

Associates a statement with one of its parameters; it is the inverse of parameterFor.

ekg-story-service:hasParameter	
https://ekgf.org/ontology/story-service#hasParameter	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	1 April 2020, 18:16:18 UTC
rdf:type	owl:InverseFunctionalProperty
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-story-service:Statement

ekg-story-service:hasParameter	
https://ekgf.org/ontology/story-service#hasParameter	
RDF predicate	**Value**
rdfs:label	Has Parameter
rdfs:range	ekg-story-service:Parameter
owl:inverseOf	ekg-story-service:parameterFor

20.3.2 Is Service Parameter

Records whether a parameter is classified as a service parameter by the story-service configuration.

ekg-story-service:isServiceParameter	
https://ekgf.org/ontology/story-service#isServiceParameter	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	8 April 2020, 00:06:23 UTC
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-story-service:Parameter
rdfs:label	Is Service Parameter
rdfs:range	xsd:boolean

20.3.3 On

Identifies the story to which a parameter or result description relates.

ekg-story-service:on	
https://ekgf.org/ontology/story-service#on	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	(rdf:type: owl:Class; owl:unionOf: (ekg-story-service:Parameter, ekg-story-service:Result))
rdfs:label	On
rdfs:range	ekg-story:Story

20.3.4 Parameter Direction

Records the direction assigned to a parameter as a string. The ontology does not declare a controlled set of direction values.

ekg-story-service:parameterDirection	
https://ekgf.org/ontology/story-service#parameterDirection	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068

ekg-story-service:parameterDirection	
https://ekgf.org/ontology/story-service#parameterDirection	
RDF predicate	**Value**
dc:date	8 April 2020, 00:03:25 UTC
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-story-service:Parameter
rdfs:label	Parameter Direction
rdfs:range	xsd:string

20.3.5 Parameter For

Identifies the statement for which a parameter is defined. The functional-property axiom limits a parameter to one statement.

ekg-story-service:parameterFor	
https://ekgf.org/ontology/story-service#parameterFor	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	1 April 2020, 17:32:57 UTC
rdf:type	owl:FunctionalProperty
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-story-service:Parameter
rdfs:label	Parameter For
rdfs:range	ekg-story-service:Statement

20.3.6 Parameter Key

Records the string key by which a parameter is identified in a statement or service configuration.

ekg-story-service:parameterKey	
https://ekgf.org/ontology/story-service#parameterKey	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	1 April 2020, 18:19:13 UTC
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-story-service:Parameter
rdfs:label	Parameter Key
rdfs:range	xsd:string

20.3.7 Parameter Op

Records an operation value associated with a parameter. The ontology does not specify its literal datatype or the allowed operations.

ekg-story-service:parameterOp	
https://ekgf.org/ontology/story-service#parameterOp	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	1 April 2020, 18:21:12 UTC
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-story-service:Parameter
rdfs:label	Parameter Op

20.3.8 Parameter Type

Records the parameter type identifier used by the story-service configuration. The value is a string rather than a link to an RDF class or datatype.

ekg-story-service:parameterType	
https://ekgf.org/ontology/story-service#parameterType	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	1 April 2020, 18:20:08 UTC
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-story-service:Parameter
rdfs:label	Parameter Type
rdfs:range	xsd:string

20.3.9 should be supplied by

The execution of any given story can be done by (supplied by) a given (logical) instance of the story Service.

ekg-story-service:providedBy	
https://ekgf.org/ontology/story-service#providedBy	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	8 April 2020, 00:10:29 UTC
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-story:Story

ekg-story-service:providedBy	
https://ekgf.org/ontology/story-service#providedBy	
RDF predicate	**Value**
rdfs:label	should be supplied by
rdfs:range	ekg-story-service:StoryService

20.3.10 Result For

Relates a SPARQL result description to the story that produces the result.

ekg-story-service:resultFor	
https://ekgf.org/ontology/story-service#resultFor	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-story-service:SPARQLResults
rdfs:label	Result For
rdfs:range	ekg-story:Story

20.3.11 Result Key

Records a result-entry key in the story-service configuration. Its currently declared domain is Parameter, which requires review before treating it as a property restricted to Result.

ekg-story-service:resultKey	
https://ekgf.org/ontology/story-service#resultKey	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	7 April 2020, 20:46:55 UTC
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-story-service:Parameter
rdfs:label	Result Key
rdfs:range	xsd:string

20.3.12 Result Type

Records the result type identifier used by the story-service configuration as a string.

ekg-story-service:resultType	
https://ekgf.org/ontology/story-service#resultType	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	7 April 2020, 20:51:59 UTC

ekg-story-service:resultType	
https://ekgf.org/ontology/story-service#resultType	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-story-service:Result
rdfs:label	Result Type
rdfs:range	xsd:string

20.3.13 Should Be Generated By

Identifies the story-service instance expected to generate a parameter. The property is functional; its intended distinction from other parameter-to-service relationships requires modeling review.

ekg-story-service:shouldBeGeneratedBy	
https://ekgf.org/ontology/story-service#shouldBeGeneratedBy	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	1 April 2020, 17:25:50 UTC
rdf:type	owl:FunctionalProperty
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-story-service:Parameter
rdfs:label	Should Be Generated By
rdfs:range	ekg-story-service:StoryService

20.3.14 Sparql Query Name

Records the configured name of a story's SPARQL query.

ekg-story-service:sparqlQueryName	
https://ekgf.org/ontology/story-service#sparqlQueryName	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	7 April 2020, 20:50:13 UTC
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-story:Story
rdfs:label	Sparql Query Name
rdfs:range	xsd:string

20.3.15 Sparql Statement File Name

Records the name of the SPARQL statement file associated with a story.

ekg-story-service:sparqlStatementFileName	
https://ekgf.org/ontology/story-service#sparqlStatementFileName	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	7 April 2020, 20:51:10 UTC
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-story:Story
rdfs:label	Sparql Statement File Name
rdfs:range	xsd:string

21 Use Case Ontology

21.1 Overview

Use Case Ontology

Identifier	Value
Ontology IRI / Namespace	`https://ekgf.org/ontology/use-case#`
Prefix	`ekg-use-case`

Use cases organize business needs in a tree. Stories belong to use cases, describe interactions, and use data element concepts to express their inputs, outputs, and dependencies.

The Method assigns concept-vocabulary ownership to use cases. Descendant use cases inherit access to their ancestors' vocabularies. A story references the concepts through its own usage objects; the shared concepts retain their vocabulary memberships.

See `ownsConceptSet` for the ownership policy and the Story Ontology for the declared usage-object model, validation shapes, and data migration instructions.

21.1.1 Outcome contributions

`outcomeLink` connects a use case to an `ekg-outcome:OutcomeRelationship` with its own identity, one target outcome and optional contribution text. The inverse source property is `ekg-outcome:useCase`. Several use cases can refer to the same outcome through different relationships. `hasOutcome` supplies an unqualified summary inferred from the relationship path.

An outcome relationship records contribution, not outcome ownership. Referring to an ancestor's outcome requires an explicit relationship. `isPartOf` and `isUsedIn` create no outcome relationships or inherited contribution metadata. The Outcome Ontology declares and validates the relationship model.

21.2 Classes

21.2.1 Use Case

A description of a business interaction or need, organized in a use-case tree and connected to its stories, outcomes, and concepts.

`ekg-use-case:UseCase`	
`https://ekgf.org/ontology/use-case#UseCase`	
RDF predicate	**Value**
`dc:creator`	`http://orcid.org/0000-0003-2309-5068`
`dc:date`	8 April 2020, 00:00:01 UTC
`rdf:type`	`owl:Class`
`rdfs:label`	Use Case

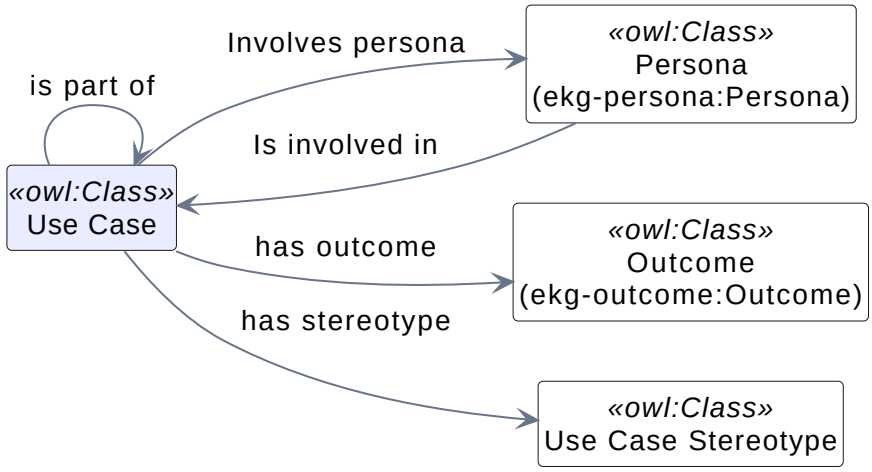


Diagram note

21.2.2 Use Case Stereotype

The kind of node a use case is in the use case tree.

ekg-use-case:UseCaseStereoType	
https://ekgf.org/ontology/use-case#UseCaseStereoType	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
rdf:type	owl:Class
rdfs:label	Use Case Stereotype

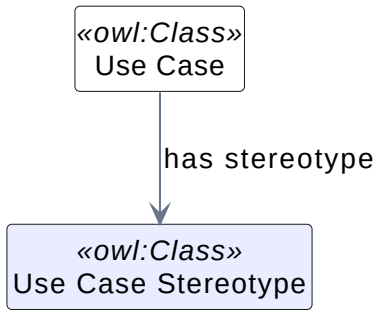


Diagram note

21.3 Properties

21.3.1 has color; has colour

Associates a resource with a color resource used for presentation. It is an object property, distinct from a hexadecimal color literal annotation.

ekg-use-case:hasColor	
https://ekgf.org/ontology/use-case#hasColor	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:label	has color
rdfs:label	has colour

21.3.2 has outcome

The property chain `outcomeLink / ekg-outcome:targetOutcome` derives a direct UseCase-to-Outcome relationship under OWL reasoning. This summary is convenient for queries that only need the target outcomes. Query the OutcomeRelationship objects when contribution identity or descriptions matter.

A direct assertion carries no contribution metadata and does not create an OutcomeRelationship or establish outcome ownership. RDFS inference alone does not execute OWL property chains. Queries can follow the relationship path directly when summary triples have not been materialized.

This property has no inheritance chain through `isPartOf` or `isUsedIn`. Use cases referencing the same ancestor outcome must each state their own relationship when recording a contribution.

ekg-use-case:hasOutcome	
https://ekgf.org/ontology/use-case#hasOutcome	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-use-case:UseCase
rdfs:label	has outcome
rdfs:range	ekg-outcome:Outcome
owl:propertyChainAxiom	(ekg-use-case:outcomeLink, ekg-outcome:targetOutcome)

21.3.3 has stereotype

Assigns a use-case stereotype to a use case. Its functional-property axiom limits a use case to one stereotype through this property.

ekg-use-case:hasStereoType	
https://ekgf.org/ontology/use-case#hasStereoType	
RDF predicate	**Value**
rdf:type	owl:FunctionalProperty
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-use-case:UseCase
rdfs:label	has stereotype
rdfs:range	ekg-use-case:UseCaseStereoType

21.3.4 hex color

Display color as a CSS hex string, such as #3b82f6.

ekg-use-case:hexColor	
https://ekgf.org/ontology/use-case#hexColor	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:label	hex color
rdfs:range	xsd:string

21.3.5 Involves persona

The given persona is involved in the current use case

ekg-use-case:involvesPersona	
https://ekgf.org/ontology/use-case#involvesPersona	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	20 June 2020, 14:13:55 UTC
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-use-case:UseCase
rdfs:label	Involves persona
rdfs:range	ekg-persona:Persona
owl:inverseOf	ekg-use-case:isInvolvedInUseCase

21.3.6 Is involved in

This persona is involved in the given use case

ekg-use-case:isInvolvedInUseCase	
https://ekgf.org/ontology/use-case#isInvolvedInUseCase	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	8 April 2020, 00:01:05 UTC
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-persona:Persona
rdfs:label	Is involved in
rdfs:range	ekg-use-case:UseCase

21.3.7 is part of

The parent of the use case in the use case tree. A use case without a parent is a root.

ekg-use-case:isPartOf	
https://ekgf.org/ontology/use-case#isPartOf	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	20 June 2020, 12:36:22 UTC
rdf:type	owl:IrreflexiveProperty
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-use-case:UseCase
rdfs:label	is part of
rdfs:range	ekg-use-case:UseCase

21.3.8 is used in

Another use case in which this use case is used.

ekg-use-case:isUsedIn	
https://ekgf.org/ontology/use-case#isUsedIn	
RDF predicate	**Value**
rdf:type	owl:IrreflexiveProperty
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-use-case:UseCase
rdfs:label	is used in
rdfs:range	ekg-use-case:UseCase

21.3.9 Is used in use case

The current data element concept is used in the given UseCase

ekg-use-case:isUsedInUseCase	
https://ekgf.org/ontology/use-case#isUsedInUseCase	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	20 June 2020, 12:40:56 UTC
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-concept:DataElementConcept
rdfs:label	Is used in use case
rdfs:range	ekg-use-case:UseCase
owl:inverseOf	ekg-use-case:usesConcept

21.3.10 outcome link

Links a UseCase to an OutcomeRelationship describing that use case's contribution. Its inverse is `ekg-outcome:useCase`; assert either direction. The range lets an OWL reasoner infer the relationship class, while the domain infers the source UseCase class.

A use case can have several relationships. Each relationship has exactly one source and one target outcome, enforced by the Outcome Ontology's validator, which counts both source directions without requiring OWL inverse inference. Sharing an outcome or a source-target pair does not merge relationship identities. Use-case parentage and reuse do not create these links automatically.

ekg-use-case:outcomeLink	
https://ekgf.org/ontology/use-case#outcomeLink	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-use-case:UseCase
rdfs:label	outcome link
rdfs:range	ekg-outcome:OutcomeRelationship
owl:inverseOf	ekg-outcome:useCase

21.3.11 owns concept-set

Under the Use Case Tree Method, each concept vocabulary is owned by exactly one use case; a use case can own several vocabularies. The vocabulary is represented here by `ekg-concept:ConceptSet`, with concepts linked by `ekg-concept:inConceptSet`.

Stories can draw concepts from vocabularies owned by their use case and those inherited from any ancestor through the `isPartOf` tree. The Method gives descendants read-only access to ancestor vocabularies. Inheritance makes concepts available without copying them or transferring vocabulary ownership.

The story owns the particular usage relationship, including how the concept is used as input, output, or dependent. Deleting a story does not delete the shared concept.

This annotation records the Method's ownership and access policy. The current OWL domain and range do not enforce unique ownership, compute inherited availability, or grant application access.

ekg-use-case:ownsConceptSet	
https://ekgf.org/ontology/use-case#ownsConceptSet	
RDF predicate	**Value**
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-use-case:UseCase
rdfs:label	owns concept-set
rdfs:range	ekg-concept:ConceptSet

21.3.12 screen layout index

Index that places the use case in the screen layout of the use case tree.

ekg-use-case:screenLayoutIndex	
https://ekgf.org/ontology/use-case#screenLayoutIndex	
RDF predicate	**Value**
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-use-case:UseCase
rdfs:label	screen layout index
rdfs:range	xsd:integer

21.3.13 Used in

The use case that a story or an outcome belongs to. It has no domain because both stories and outcomes carry it.

ekg-use-case:usedIn	
https://ekgf.org/ontology/use-case#usedIn	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	8 April 2020, 00:02:11 UTC
rdf:type	owl:ObjectProperty
rdfs:label	Used in
rdfs:range	ekg-use-case:UseCase

21.3.14 Uses concept

Relates a use case to a data element concept involved in its interactions.

ekg-use-case:usesConcept	
https://ekgf.org/ontology/use-case#usesConcept	
RDF predicate	**Value**
dc:creator	http://orcid.org/0000-0003-2309-5068
dc:date	20 June 2020, 12:40:26 UTC
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-use-case:UseCase
rdfs:label	Uses concept
rdfs:range	ekg-concept:DataElementConcept

21.4 Named Individuals

21.4.1 Application

Instance of: ekg-use-case:UseCaseStereoType

An application-level use case.

ekg-use-case:UseCaseStereoTypeApplication	
https://ekgf.org/ontology/use-case#UseCaseStereoTypeApplication	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-use-case:UseCaseStereoType
rdfs:label	Application
ekg-annotation:slug	app
ekg-use-case:hexColor	#2563eb

21.4.2 Business Capability

Instance of: ekg-use-case:UseCaseStereoType

A business capability in the enterprise capability map.

ekg-use-case:UseCaseStereoTypeBusinessCapability	
https://ekgf.org/ontology/use-case#UseCaseStereoTypeBusinessCapability	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-use-case:UseCaseStereoType
rdfs:label	Business Capability
ekg-annotation:slug	business-capability
ekg-use-case:hexColor	#365885

21.4.3 Business Capability Area

Instance of: ekg-use-case:UseCaseStereoType

A grouping of related business capabilities.

ekg-use-case:UseCaseStereoTypeBusinessCapabilityArea	
https://ekgf.org/ontology/use-case#UseCaseStereoTypeBusinessCapabilityArea	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-use-case:UseCaseStereoType
rdfs:label	Business Capability Area
ekg-annotation:slug	business-capability-area
ekg-use-case:hexColor	#5b6d8a

21.4.4 Data Domain

Instance of: ekg-use-case:UseCaseStereoType

A top-level data domain in the enterprise data landscape. A data domain owns and manages concept sets and their concepts, and usually has no stories.

ekg-use-case:UseCaseStereoTypeDataDomain	
https://ekgf.org/ontology/use-case#UseCaseStereoTypeDataDomain	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-use-case:UseCaseStereoType
rdfs:label	Data Domain
ekg-annotation:slug	data-domain
ekg-use-case:hexColor	#0f766e

21.4.5 Data Landscape

Instance of: ekg-use-case:UseCaseStereoType

The controlled use-case stereotype identifying a data-landscape use case.

ekg-use-case:UseCaseStereoTypeDataLandscape	
https://ekgf.org/ontology/use-case#UseCaseStereoTypeDataLandscape	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-use-case:UseCaseStereoType
rdfs:label	Data Landscape
ekg-annotation:slug	data-landscape

21.4.6 Data Subdomain

Instance of: ekg-use-case:UseCaseStereoType

A subdomain that belongs to a parent data domain.

ekg-use-case:UseCaseStereoTypeDataSubdomain	
https://ekgf.org/ontology/use-case#UseCaseStereoTypeDataSubdomain	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-use-case:UseCaseStereoType
rdfs:label	Data Subdomain
ekg-annotation:slug	data-subdomain
ekg-use-case:hexColor	#2ea39a

21.4.7 Data Transformation

Instance of: ekg-use-case:UseCaseStereoType

A use case that transforms data from one or more input datasets into one or more output datasets. Most of its stories are update stories.

ekg-use-case:UseCaseStereoTypeDataTransformation	
https://ekgf.org/ontology/use-case#UseCaseStereoTypeDataTransformation	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-use-case:UseCaseStereoType
rdfs:label	Data Transformation
ekg-annotation:slug	data-transformation
ekg-use-case:hexColor	#b7791f

21.4.8 Enterprise

Instance of: ekg-use-case:UseCaseStereoType

The Enterprise stereotype identifies the enterprise scope at the root of a use-case tree. Its public name is `Enterprise`, its identifier is `UseCaseStereoTypeEnterprise`, and its resource slug is `enterprise`.

Assign it to a use case through `hasStereoType`, for example:

```
@prefix use-case: <https://ekgf.org/ontology/use-case#> .
@prefix ex: <https://example.test/> .

ex:enterprise use-case:hasStereoType use-case:UseCaseStereoTypeEnterprise .
```

A structural root has no parent through `isPartOf`. Several enterprise roots can be represented in the same graph; the stereotype imposes no global single-enterprise constraint or automatic parent assignment.

ekg-use-case:UseCaseStereoTypeEnterprise	
https://ekgf.org/ontology/use-case#UseCaseStereoTypeEnterprise	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-use-case:UseCaseStereoType
rdfs:label	Enterprise
ekg-annotation:slug	enterprise
ekg-use-case:hexColor	#6b7280

21.4.9 Use Case

Instance of: ekg-use-case:UseCaseStereoType

A concrete use case delivered within a business or data context. The default stereotype of a new use case.

ekg-use-case:UseCaseStereoTypeUseCase	
https://ekgf.org/ontology/use-case#UseCaseStereoTypeUseCase	
RDF predicate	**Value**
rdf:type	owl:NamedIndividual
rdf:type	ekg-use-case:UseCaseStereoType

ekg-use-case:UseCaseStereoTypeUseCase	
https://ekgf.org/ontology/use-case#UseCaseStereoTypeUseCase	
RDF predicate	**Value**
rdfs:label	Use Case
ekg-annotation:slug	use-case
ekg-use-case:hexColor	#5b5bd6

A Documentation coverage

20 ontologies; 770 documented named resources. 0 resources have no English source definition.

Missing annotations are shown explicitly and recorded as KG diagnostics. No inferred definitions or external imports are added.

Ontology	Resources	Missing definitions
Annotation Ontology	8	0
Artifact Dependency Ontology	45	0
Artifact Governance Ontology	50	0
Artifact Publication Ontology	33	0
Authorization Ontology	4	0
Capability Ontology	5	0
Communication Ontology	140	0
Concept Ontology	69	0
DataOps Ontology	13	0
File System Ontology	101	0
Lifecycle Ontology	22	0
Maturity Model Ontology	26	0
Outcome Ontology	30	0
Persona Ontology	5	0
Raw Data Ontology	59	0
Specification by Example Ontology	25	0
Story Ontology	43	0
SPARQL Story Implementation Ontology	44	0
Story Service Ontology	23	0
Use Case Ontology	25	0

B Document provenance

This record identifies the generator and source revisions collected for this publication.

Generator	GraphArch 0.2.29 (build 412)
Generator commit	`8908a9610d77`
Build operator	Rust CI, run 44327712575 attempt 1 (GitHub Actions, jgeluk)
Repository	https://github.com/EKGF/ekg-ontologies.git
Source commit	`3a07397e139d`
Source state	Exact committed revision.

Recent commits

Up to five commits, following the first-parent history from the source commit, with the author of each change. This is not the full contributor list.

Jacobus Geluk 18 November 2026, 13:58 UTC source commit
Jacobus Geluk 18 November 2026, 13:29 UTC `37eed23027a2`
Jacobus Geluk 18 November 2026, 00:23 UTC `e5cdbecda70c`
Jacobus Geluk 17 November 2026, 23:23 UTC `a09ac589055a`
Jacobus Geluk 17 November 2026, 22:53 UTC `3bc3433c978d`

Diagram interpretation

Class diagrams use UML shorthand for asserted OWL facts and declared SHACL constraints, without inference or a claim that validation passed. RDF properties remain independent resources even when drawn as class members or labelled associations. Domain/range declarations alone do not impose requiredness or cardinality. A property with a declared domain and range is a labelled association from the domain to the range; a datatype property, or one without a specific range, is an attribute of its domain. Multiple domains or ranges apply conjunctively, not as alternatives: each declared domain-range pair is drawn and holds for every use of the property. An explicitly declared union domain is drawn as one source box whose class alternatives are joined by or. A dotted rdfs:Resource source box is a presentation placeholder when no specific domain is declared in the selected inputs, not an asserted domain or a claim that none can be entailed. OWL class restrictions on the documented class and on neighbouring classes use UML notation instead of expression boxes, scoped to the restricted class and never declaring a global range or count. A datatype restriction is an attribute, name : type [multiplicity]; an object restriction is an association to its filler class with the multiplicity at the target end (a neighbouring class shows it as a typed attribute instead). An unqualified count uses the declared property range. someValuesFrom gives 1..* {qualified}; allValuesFrom gives * {all values}, narrowing the value type; a qualified count counts only values in the named filler and adds {qualified}; hasValue adds {= value}. Multiplicities follow one convention: equal bounds show one number such as 1, an unconstrained range shows *, and otherwise lower..upper such as 0..1 or 1..*. A maximum or exact count of zero, in OWL or SHACL, is a prohibition rather than a multiplicity: it appears as a class constraint row after the attributes, {no p}, or {no p in C} when only values in C are excluded. A zero minimum imposes no restriction and is omitted. {redefines} marks a restriction on a property whose declared domain is an asserted superclass, redefining the inherited property of the same name. An existential with a simple zero-minimum filler still requires a value; a positive count on the same class and property already guarantees that existence. Boolean class expressions, nested restrictions, inverse properties and property chains remain explicit boxes named after their form, such as Anonymous OWL intersection 1. Types and counts from simple property shapes enrich the UML notation. Identical qualifiers are shown once; where declaring shapes disagree, the disputed type or multiplicity is omitted from the drawing and remains in

the dedicated shape documentation. No shape is chosen arbitrarily and conflicting counts are not silently intersected. An absent count imposes no lower or upper bound beyond zero or unbounded, respectively. Complex constraints require the linked full shape definition. Original OWL facts and shape provenance remain in the published source graph. Some diagrams show at most five complete OWL relationship groups within an eighteen-fact budget, selected in deterministic IRI order; omitted groups remain in the published source graph. These views are not exhaustive. Names from the same ontology as the documented class are shown by label; names from another ontology add their prefixed name in parentheses, and unlabelled names show their prefixed name.