EKGF Editor's Draft

OMG - Enterprise Knowledge Graph Forum

EKGF Ontologies

EKGF Editor's Draft 25 October 2026

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

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

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

Document license

EKGF Editor's Draft

Contents

Abstract . . . 20
Status of This Document . . . 20
1 Family overview . . . 20
1.1 EKGF Ontologies . . . 20
2 Annotation Ontology . . . 22
2.1 Overview . . . 22
2.2 Properties . . . 22
2.2.1 Markdown description . . . 22
2.2.2 subtitle . . . 23
2.2.3 summary . . . 23
3 Platform API Ontology . . . 23
3.1 Overview . . . 23
3.2 Properties . . . 23
3.2.1 slug . . . 23
3.2.2 taxonomy kind . . . 24
4 Artifact Dependency Ontology . . . 24
4.1 Overview . . . 24
4.2 Classes . . . 24
4.2.1 Dependency . . . 24
4.2.2 Dependency evidence . . . 24
4.2.3 Dependency kind . . . 25
4.2.4 Dependency resolution . . . 25
4.2.5 Explicit dependency . . . 25
4.2.6 Implicit dependency . . . 25
4.2.7 Ontology load request . . . 25
4.2.8 Resolution status . . . 26
4.2.9 Semantic artifact . . . 26
4.3 Properties . . . 26
4.3.1 dependency evidence . . . 26
4.3.2 dependency kind . . . 26
4.3.3 dependency target . . . 27
4.3.4 dependent artifact . . . 27
4.3.5 depends on . . . 27
4.3.6 evidence graph . . . 27
4.3.7 evidence object . . . 28
4.3.8 evidence predicate . . . 28
4.3.9 evidence subject . . . 28
4.3.10 has dependency . . . 28
4.3.11 load hint . . . 29
4.3.12 load request authority . . . 29
4.3.13 load request dependency . . . 29
4.3.14 load request family . . . 29
4.3.15 load request target . . . 30
4.3.16 referenced namespace . . . 30
4.3.17 resolution . . . 30
4.3.18 resolution branch . . . 30
4.3.19 resolution commit . . . 31
4.3.20 resolution message . . . 31
4.3.21 resolution status . . . 31

EKGF Editor's Draft

4.3.22 resolution workspace . . . 31
4.3.23 resolved artifact . . . 32
4.4 Named Individuals . . . 32
4.4.1 Ambiguous . . . 32
4.4.2 Declared import dependency . . . 32
4.4.3 External . . . 33
4.4.4 Failed . . . 33
4.4.5 Incomplete . . . 33
4.4.6 Loaded . . . 33
4.4.7 Namespace reference dependency . . . 34
4.4.8 Profile dependency . . . 34
4.4.9 Shape graph dependency . . . 34
4.4.10 Term reference dependency . . . 34
4.4.11 Unloaded . . . 35
4.4.12 Unsupported . . . 35
4.4.13 Validation dependency . . . 35
5 Artifact Governance Ontology . . . 35
5.1 Overview . . . 35
5.2 Classes . . . 36
5.2.1 Authority . . . 36
5.2.2 Authority publishing role . . . 36
5.2.3 Family . . . 36
5.2.4 Organization category . . . 36
5.2.5 Reference . . . 37
5.2.6 Standards community role . . . 37
5.3 Properties . . . 37
5.3.1 Authority acronym . . . 37
5.3.2 Authority host . . . 37
5.3.3 Auto-load . . . 38
5.3.4 Base IRI . . . 38
5.3.5 Community organization . . . 38
5.3.6 Community role . . . 39
5.3.7 Family acronym . . . 39
5.3.8 File path prefix . . . 39
5.3.9 Git branch . . . 39
5.3.10 Git path . . . 40
5.3.11 Git repository URL . . . 40
5.3.12 Governed by . . . 40
5.3.13 Graph IRI prefix . . . 41
5.3.14 Historical name . . . 41
5.3.15 In family . . . 41
5.3.16 Legal owner . . . 41
5.3.17 Maximum family load quads . . . 42
5.3.18 Maximum source quads . . . 42
5.3.19 Namespace IRI prefix . . . 42
5.3.20 Organization category . . . 42
5.3.21 Predecessor authority . . . 43
5.3.22 Publishing role . . . 43
5.3.23 RDF file extension . . . 43
5.3.24 Refers to . . . 44
5.3.25 Source graph repository URL . . . 44

EKGF Editor's Draft

5.3.26 Successor authority . . . 44
5.4 Named Individuals . . . 44
5.4.1 co-owner . . . 44
5.4.2 maintainer . . . 45
5.4.3 owner . . . 45
5.4.4 publisher . . . 45
5.4.5 Community . . . 45
5.4.6 Consortium . . . 46
5.4.7 Organization categories for artifact governance . . . 46
5.4.8 Project . . . 46
5.4.9 Standards body . . . 47
5.4.10 standards editor . . . 47
5.4.11 standards publisher . . . 47
5.4.12 standards reviewer . . . 48
5.4.13 standards staff . . . 48
5.4.14 working-group chair . . . 48
6 Artifact Publication Ontology . . . 48
6.1 Overview . . . 48
6.2 Classes . . . 49
6.2.1 Publication artifact . . . 49
6.2.2 Publication profile . . . 49
6.2.3 Publication run . . . 49
6.2.4 Publication target . . . 49
6.2.5 Release approval . . . 50
6.2.6 Standards release . . . 50
6.3 Properties . . . 51
6.3.1 Approved at . . . 51
6.3.2 Approved by . . . 51
6.3.3 Approves release . . . 51
6.3.4 Artifact checksum . . . 51
6.3.5 Artifact media type . . . 52
6.3.6 Artifact URL . . . 52
6.3.7 Generator version . . . 52
6.3.8 Input commit CID . . . 52
6.3.9 Publication artifact . . . 53
6.3.10 Publication branch . . . 53
6.3.11 Publication completed at . . . 53
6.3.12 Publication profile . . . 53
6.3.13 Publication release . . . 54
6.3.14 Publication run . . . 54
6.3.15 Publication started at . . . 54
6.3.16 Publication target . . . 54
6.3.17 Publishes family . . . 55
6.3.18 Publishes reference . . . 55
6.3.19 Release family . . . 55
6.3.20 Release reference . . . 55
6.3.21 Release version . . . 56
6.3.22 Released at . . . 56
6.3.23 Target kind . . . 56
6.3.24 Target URL . . . 56
6.4 Named Individuals . . . 57

EKGF Editor's Draft

6.4.1 Artifact Publication Validation Shapes . . . 57
6.5 Shapes . . . 57
6.5.1 Publication Profile Shape . . . 57
6.5.2 Release Approval Shape . . . 58
6.5.3 Standards Release Shape . . . 58
7 Authorization Ontology . . . 60
7.1 Overview . . . 60
7.2 Classes . . . 60
7.2.1 Authorization Group . . . 60
7.2.2 Authorization Rule . . . 60
7.3 Properties . . . 61
7.3.1 Has Group . . . 61
7.3.2 Has Persona . . . 61
8 Capability Ontology . . . 61
8.1 Overview . . . 61
8.2 Classes . . . 61
8.2.1 Capability . . . 62
8.2.2 Capability Domain . . . 62
8.2.3 Sub-capability . . . 62
8.3 Named Individuals . . . 62
8.3.1 Data sharing and marketplace . . . 62
8.3.2 Designing and managing data products . . . 62
9 Communication Ontology . . . 63
9.1 Overview . . . 63
9.2 Classes . . . 63
9.2.1 Audience . . . 63
9.2.2 Audience membership . . . 63
9.2.3 Campaign audience snapshot . . . 64
9.2.4 Campaign recipient . . . 64
9.2.5 Communication attempt . . . 64
9.2.6 Communication channel . . . 64
9.2.7 Communication endpoint . . . 64
9.2.8 Communication program . . . 65
9.2.9 Communication purpose . . . 65
9.2.10 Delivery event . . . 65
9.2.11 Delivery status . . . 65
9.2.12 Email endpoint . . . 65
9.2.13 Email recipient . . . 66
9.2.14 Group recipient . . . 66
9.2.15 Human recipient . . . 66
9.2.16 In-app endpoint . . . 66
9.2.17 In-app recipient . . . 67
9.2.18 Legal basis . . . 67
9.2.19 List membership . . . 67
9.2.20 Mailbox recipient . . . 67
9.2.21 Marketing program . . . 68
9.2.22 Membership . . . 68
9.2.23 Membership derivation evidence . . . 68
9.2.24 Membership event . . . 68
9.2.25 Membership evidence . . . 68
9.2.26 Membership status . . . 69

EKGF Editor's Draft

9.2.27 Organization recipient . . . 69
9.2.28 Postal endpoint . . . 69
9.2.29 Postal recipient . . . 69
9.2.30 Push endpoint . . . 69
9.2.31 Push recipient . . . 70
9.2.32 Recipient . . . 70
9.2.33 Recipient collection . . . 70
9.2.34 Recipient consent . . . 70
9.2.35 Recipient consent withdrawal . . . 71
9.2.36 Recipient legal category . . . 71
9.2.37 Recipient legal classification . . . 71
9.2.38 Recipient list . . . 71
9.2.39 Recipient suppression . . . 72
9.2.40 Recipient tag . . . 72
9.2.41 Service recipient . . . 72
9.2.42 SMS endpoint . . . 72
9.2.43 SMS recipient . . . 73
9.2.44 Tag assignment . . . 73
9.2.45 Voice endpoint . . . 73
9.2.46 Voice recipient . . . 73
9.2.47 Webhook endpoint . . . 74
9.2.48 Webhook recipient . . . 74
9.3 Properties . . . 74
9.3.1 allowed channel . . . 74
9.3.2 attempt endpoint . . . 74
9.3.3 attempt recipient . . . 75
9.3.4 campaign audience . . . 75
9.3.5 captured at . . . 75
9.3.6 collection purpose . . . 75
9.3.7 communication program . . . 76
9.3.8 consent membership . . . 76
9.3.9 delivery attempt . . . 76
9.3.10 delivery status . . . 76
9.3.11 derived from membership . . . 77
9.3.12 effective from . . . 77
9.3.13 effective through . . . 77
9.3.14 endpoint address . . . 77
9.3.15 endpoint channel . . . 78
9.3.16 evidence collection . . . 78
9.3.17 evidence source . . . 78
9.3.18 has delivery event . . . 78
9.3.19 has endpoint . . . 79
9.3.20 has membership . . . 79
9.3.21 jurisdiction code . . . 79
9.3.22 legal basis . . . 79
9.3.23 legal category . . . 80
9.3.24 legal classification recipient . . . 80
9.3.25 legal jurisdiction . . . 80
9.3.26 membership collection . . . 80
9.3.27 membership endpoint . . . 81
9.3.28 membership event . . . 81

EKGF Editor's Draft

9.3.29 membership evidence . . . 81
9.3.30 membership recipient . . . 81
9.3.31 membership status . . . 82
9.3.32 occurred at . . . 82
9.3.33 policy version . . . 82
9.3.34 program channel . . . 82
9.3.35 program purpose . . . 83
9.3.36 provenance . . . 83
9.3.37 provider message id . . . 83
9.3.38 provider recipient id . . . 83
9.3.39 recorded at . . . 84
9.3.40 represented resource . . . 84
9.3.41 retention period . . . 84
9.3.42 retention policy . . . 84
9.3.43 suppression membership . . . 85
9.3.44 withdrawn consent . . . 85
9.4 Named Individuals . . . 85
9.4.1 Active membership . . . 85
9.4.2 Bounced . . . 86
9.4.3 Complained . . . 86
9.4.4 Consent legal basis . . . 86
9.4.5 Consent withdrawn membership . . . 86
9.4.6 Contract legal basis . . . 87
9.4.7 Delivered . . . 87
9.4.8 Email channel . . . 87
9.4.9 Employee category . . . 87
9.4.10 Employee operational legal basis . . . 88
9.4.11 Failed . . . 88
9.4.12 Group mailbox category . . . 88
9.4.13 In-app notification channel . . . 88
9.4.14 Inactive membership . . . 89
9.4.15 Legal notice purpose . . . 89
9.4.16 Legal obligation legal basis . . . 89
9.4.17 Legitimate interest legal basis . . . 89
9.4.18 Marketing purpose . . . 90
9.4.19 Natural person category . . . 90
9.4.20 Operational purpose . . . 90
9.4.21 Organization category . . . 90
9.4.22 Pending . . . 91
9.4.23 Postal mail channel . . . 91
9.4.24 Public authority category . . . 91
9.4.25 Push notification channel . . . 91
9.4.26 Security purpose . . . 92
9.4.27 Sent . . . 92
9.4.28 Skipped . . . 92
9.4.29 SMS channel . . . 92
9.4.30 Support purpose . . . 93
9.4.31 Suppressed . . . 93
9.4.32 Suppressed membership . . . 93
9.4.33 Technical service category . . . 93
9.4.34 Technical service legal basis . . . 94

EKGF Editor's Draft

9.4.35 Transactional purpose . . . 94
9.4.36 Unsubscribed . . . 94
9.4.37 Communication Validation Shapes . . . 94
9.4.38 Voice call channel . . . 95
9.4.39 Webhook channel . . . 95
9.5 Shapes . . . 95
9.5.1 Communication Attempt Shape . . . 95
9.5.2 Communication Endpoint Shape . . . 96
9.5.3 Communication Program Shape . . . 96
9.5.4 Delivery Event Shape . . . 97
9.5.5 Email Recipient Shape . . . 97
9.5.6 Membership Shape . . . 97
9.5.7 Recipient Collection Shape . . . 98
9.5.8 Recipient Consent Shape . . . 98
9.5.9 Recipient Legal Classification Shape . . . 99
9.5.10 Recipient Shape . . . 99
10 Concept Ontology . . . 100
10.1 Overview . . . 100
10.1.1 Shared meanings, usage, and manifestations . . . 100
10.2 Classes . . . 100
10.2.1 abbreviation . . . 100
10.2.2 business term . . . 100
10.2.3 concept manifestation . . . 100
10.2.4 concept set . . . 101
10.2.5 Cypher identifier manifestation . . . 101
10.2.6 data element concept . . . 101
10.2.7 description . . . 102
10.2.8 full plural business term . . . 102
10.2.9 full singular business term . . . 103
10.2.10 JSON field manifestation . . . 103
10.2.11 Java identifier manifestation . . . 103
10.2.12 JavaScript identifier manifestation . . . 103
10.2.13 literal datatype manifestation . . . 104
10.2.14 manifestation occurrence . . . 104
10.2.15 OWL axiom manifestation . . . 104
10.2.16 OWL class manifestation . . . 104
10.2.17 OWL named individual manifestation . . . 105
10.2.18 OWL property manifestation . . . 105
10.2.19 Python identifier manifestation . . . 105
10.2.20 RDF predicate manifestation . . . 105
10.2.21 RDFS class manifestation . . . 106
10.2.22 raw source field manifestation . . . 106
10.2.23 SHACL shape manifestation . . . 106
10.2.24 SHACL shapeset manifestation . . . 106
10.2.25 SKOS concept manifestation . . . 107
10.2.26 SKOS concept scheme manifestation . . . 107
10.2.27 SPARQL result column manifestation . . . 107
10.2.28 SPARQL variable manifestation . . . 107
10.2.29 SQL column manifestation . . . 108
10.2.30 SQL table manifestation . . . 108
10.2.31 source code identifier manifestation . . . 108

EKGF Editor's Draft

10.2.32 tagline . . . 109
10.2.33 technical manifestation . . . 109
10.2.34 Template Language . . . 109
10.2.35 template parameter manifestation . . . 109
10.2.36 URI parameter manifestation . . . 110
10.3 Properties . . . 110
10.3.1 concept . . . 110
10.3.2 has concept . . . 110
10.3.3 has manifestation . . . 110
10.3.4 in concept set . . . 111
10.3.5 manifestation resource . . . 111
10.3.6 manifestation value . . . 111
10.3.7 occurrence of . . . 111
10.3.8 preferred business term . . . 112
10.3.9 preferred manifestation . . . 112
10.3.10 preferred query parameter manifestation . . . 112
10.3.11 preferred result column manifestation . . . 113
10.3.12 source column . . . 113
10.3.13 source commit . . . 113
10.3.14 source line . . . 113
10.3.15 source path . . . 114
10.3.16 source repository . . . 114
10.3.17 source system . . . 114
10.3.18 template language . . . 114
10.4 Named Individuals . . . 115
10.4.1 Concept Validation Shapes . . . 115
10.5 Shapes . . . 115
10.5.1 Legacy Template Manifestation Shape . . . 115
10.5.2 Template Language Shape . . . 116
10.5.3 Template Parameter Manifestation Shape . . . 116
11 DataOps Ontology . . . 116
11.1 Overview . . . 116
11.2 Classes . . . 117
11.2.1 Failed RDF Load Request . . . 117
11.2.2 Finished RDF Load Request . . . 117
11.2.3 RDF Load Request . . . 117
11.2.4 Loading RDF Load Request . . . 117
11.2.5 Logical RDF Dataset . . . 117
11.2.6 Pipeline . . . 118
11.2.7 Queued RDF Load Request . . . 118
11.2.8 Single-graph RDF Dataset . . . 118
11.3 Properties . . . 118
11.3.1 Created By Pipeline . . . 118
11.3.2 Load Request Target Graph . . . 119
11.3.3 In Pipeline . . . 119
11.3.4 Loaded By Load Request . . . 119
11.3.5 Load Request Source IRI . . . 120
12 Dataset Ontology . . . 120
12.1 Overview . . . 120
12.2 Classes . . . 120
12.2.1 Compound Dataset . . . 120

EKGF Editor's Draft

12.2.2 Dataset . . . 120
12.2.3 Named Graph . . . 121
12.2.4 Static Dataset . . . 121
12.3 Properties . . . 121
12.3.1 Data Source Code . . . 121
12.3.2 Dataset Code . . . 121
12.3.3 default graph . . . 122
12.3.4 include . . . 122
12.3.5 Includes Dataset . . . 122
13 File System Ontology . . . 123
13.1 Overview . . . 123
13.2 Classes . . . 123
13.2.1 archive entry kind . . . 123
13.2.2 archive format . . . 123
13.2.3 archive member . . . 123
13.2.4 archive version . . . 123
13.2.5 attachment . . . 124
13.2.6 attachment mode . . . 124
13.2.7 attachment role . . . 124
13.2.8 capability profile . . . 124
13.2.9 directory . . . 125
13.2.10 directory entry . . . 125
13.2.11 external object . . . 125
13.2.12 file series . . . 125
13.2.13 file system view . . . 126
13.2.14 file version . . . 126
13.2.15 name family rule . . . 126
13.2.16 path rule . . . 126
13.2.17 service shortcut . . . 126
13.2.18 source kind . . . 127
13.2.19 suffix pattern . . . 127
13.2.20 symbolic link . . . 127
13.2.21 tombstone . . . 127
13.3 Properties . . . 127
13.3.1 allowed suffix pattern . . . 127
13.3.2 also allow final/draft suffix . . . 128
13.3.3 archive format . . . 128
13.3.4 archive version . . . 128
13.3.5 attached to . . . 128
13.3.6 attachment mode . . . 129
13.3.7 attachment role . . . 129
13.3.8 base name . . . 129
13.3.9 byte size . . . 129
13.3.10 capability profile . . . 130
13.3.11 case sensitive names . . . 130
13.3.12 content . . . 130
13.3.13 current version . . . 130
13.3.14 deleted at . . . 131
13.3.15 entry count . . . 131
13.3.16 entry kind . . . 131
13.3.17 extension . . . 132

EKGF Editor's Draft

13.3.18 external id . . . 132
13.3.19 external object . . . 132
13.3.20 external revision id . . . 132
13.3.21 folder . . . 132
13.3.22 has attachment . . . 133
13.3.23 has version . . . 133
13.3.24 in view . . . 133
13.3.25 legal name pattern . . . 133
13.3.26 link target . . . 134
13.3.27 member count . . . 134
13.3.28 member mode . . . 134
13.3.29 member mtime . . . 134
13.3.30 member ordinal . . . 135
13.3.31 member path . . . 135
13.3.32 member target . . . 135
13.3.33 name . . . 135
13.3.34 normalized name . . . 136
13.3.35 normalized source file name . . . 136
13.3.36 observed at . . . 136
13.3.37 office package kind . . . 136
13.3.38 opaque reason . . . 137
13.3.39 parent . . . 137
13.3.40 provider . . . 137
13.3.41 root . . . 138
13.3.42 source file name . . . 138
13.3.43 source kind . . . 138
13.3.44 supports atomic move . . . 138
13.3.45 supports links . . . 139
13.3.46 supports multiple roots . . . 139
13.3.47 supports range reads . . . 139
13.3.48 supports server-side copy . . . 139
13.3.49 supports shortcuts . . . 140
13.3.50 supports stable revision ids . . . 140
13.3.51 supports trash . . . 140
13.3.52 supports versioning . . . 140
13.3.53 symbolic target path . . . 141
13.3.54 target . . . 141
13.3.55 target external object . . . 141
13.3.56 target name family rule . . . 141
13.3.57 target path rule . . . 142
13.3.58 target series . . . 142
13.3.59 target version . . . 142
13.3.60 tombstone of . . . 142
13.3.61 unicode normalization . . . 143
13.3.62 view path . . . 143
13.4 Other Terms . . . 143
13.4.1 archive directory entry . . . 143
13.4.2 exact version . . . 143
13.4.3 gzip . . . 144
13.4.4 hardlink entry . . . 144
13.4.5 native upload . . . 144

EKGF Editor's Draft

13.4.6 nested archive . . . 144
13.4.7 other entry . . . 144
13.4.8 provider sync . . . 145
13.4.9 regular file . . . 145
13.4.10 symlink entry . . . 145
13.4.11 tar . . . 145
13.4.12 tar+gzip . . . 145
13.4.13 track external id . . . 146
13.4.14 track name family . . . 146
13.4.15 track path . . . 146
13.4.16 track series . . . 146
13.4.17 windows copy suffix . . . 146
13.4.18 zip . . . 147
14 Lifecycle Ontology . . . 147
14.1 Overview . . . 147
14.2 Classes . . . 147
14.2.1 Lifecycle . . . 147
14.2.2 Phase . . . 148
14.2.3 State . . . 148
14.2.4 State transition . . . 148
14.2.5 Transition . . . 148
14.3 Properties . . . 149
14.3.1 applies to . . . 149
14.3.2 current state . . . 149
14.3.3 follows lifecycle . . . 149
14.3.4 from state . . . 149
14.3.5 in lifecycle . . . 150
14.3.6 in state . . . 150
14.3.7 is terminal . . . 150
14.3.8 order . . . 151
14.3.9 phase . . . 151
14.3.10 prior state transition . . . 151
14.3.11 to state . . . 151
14.3.12 trigger . . . 152
14.4 Named Individuals . . . 152
14.4.1 Lifecycle Validation Shapes . . . 152
14.5 Shapes . . . 152
14.5.1 Current State Shape . . . 152
14.5.2 History State Shape . . . 154
14.5.3 Legacy State Shape . . . 154
14.5.4 State Transition Shape . . . 154
14.5.5 Transition Shape . . . 155
15 Maturity Model Ontology . . . 156
15.1 Overview . . . 156
15.2 Classes . . . 156
15.2.1 Capability . . . 156
15.2.2 Capability Area . . . 156
15.2.3 Level . . . 157
15.2.4 Model . . . 157
15.2.5 Pillar . . . 157
15.3 Properties . . . 157

EKGF Editor's Draft

15.3.1 Capability Number . . . 157
15.3.2 hasCapability . . . 158
15.3.3 hasLevel . . . 158
15.3.4 hasPillar . . . 158
15.3.5 inArea . . . 158
15.3.6 inModel . . . 159
15.3.7 inPillar . . . 159
15.3.8 IRI local name . . . 159
15.3.9 IRI local type name . . . 160
15.3.10 levelInModel . . . 160
15.3.11 pillarInModel . . . 160
15.4 Named Individuals . . . 160
15.4.1 Business Pillar . . . 160
15.4.2 Data Pillar . . . 161
15.4.3 Test EKG/MM . . . 161
15.4.4 Level 1 . . . 161
15.4.5 Level 2 . . . 162
15.4.6 Level 3 . . . 162
15.4.7 Level 4 . . . 162
15.4.8 Level 5 . . . 163
15.4.9 Organization Pillar . . . 163
15.4.10 Technology Pillar . . . 163
16 Outcome Ontology . . . 164
16.1 Overview . . . 164
16.2 Classes . . . 164
16.2.1 Outcome . . . 164
16.2.2 Outcome Stereotype . . . 164
16.3 Properties . . . 164
16.3.1 Has Outcome . . . 164
16.3.2 priority . . . 165
16.3.3 stereotype . . . 165
16.3.4 updated at . . . 165
16.4 Named Individuals . . . 165
16.4.1 Benefit . . . 165
16.4.2 Goal . . . 166
16.4.3 Objective . . . 166
16.4.4 Theme . . . 166
16.4.5 Accountability . . . 166
16.4.6 Composable Business . . . 167
16.4.7 Contextual Truth . . . 167
16.4.8 Digital Assets . . . 167
16.4.9 Environmental Sustainability . . . 168
16.4.10 Fairness . . . 168
16.4.11 Manage Profitability . . . 168
16.4.12 Manage Risk & Compliance . . . 169
16.4.13 Openness . . . 169
16.4.14 Power of Abstraction . . . 169
16.4.15 Resilience . . . 170
16.4.16 Reusability . . . 170
16.4.17 Sustainability . . . 170
16.4.18 Transparency . . . 171

EKGF Editor's Draft

16.4.19 Unknown Outcome . . . 171
17 Persona Ontology . . . 171
17.1 Overview . . . 171
17.2 Classes . . . 171
17.2.1 Accountability . . . 171
17.2.2 Persona . . . 172
17.3 Properties . . . 172
17.3.1 has accountability . . . 172
17.3.2 Is Sub Persona Of . . . 172
17.3.3 Persona key . . . 173
18 Raw Data Ontology . . . 173
18.1 Overview . . . 173
18.2 Classes . . . 173
18.2.1 Column . . . 173
18.2.2 File . . . 174
18.2.3 Null value . . . 174
18.2.4 Row . . . 174
18.2.5 Source . . . 175
18.2.6 Value . . . 175
18.2.7 View . . . 176
18.2.8 XLSX file . . . 176
18.3 Properties . . . 176
18.3.1 25th percentile . . . 177
18.3.2 50th percentile . . . 177
18.3.3 75th percentile . . . 177
18.3.4 Column number . . . 177
18.3.5 Count . . . 178
18.3.6 File name . . . 178
18.3.7 For column . . . 178
18.3.8 For Predicate . . . 179
18.3.9 The most common value’s frequency . . . 179
18.3.10 Has column . . . 179
18.3.11 Has View . . . 180
18.3.12 Is column in view . . . 180
18.3.13 Is Row In View . . . 180
18.3.14 Is View In . . . 181
18.3.15 Legacy ID . . . 181
18.3.16 Legacy Id In Iri . . . 181
18.3.17 Literal value . . . 181
18.3.18 Local name . . . 182
18.3.19 Maximum . . . 182
18.3.20 Mean . . . 182
18.3.21 Minimum . . . 183
18.3.22 Predicate . . . 183
18.3.23 Source text . . . 183
18.3.24 Standard deviation . . . 184
18.3.25 Most common value . . . 184
18.3.26 Number of unique values . . . 184
18.3.27 Value of . . . 185
18.3.28 View Row Number . . . 185
18.4 Named Individuals . . . 185

EKGF Editor's Draft

18.4.1 Raw Data Validation Shapes . . . 185
18.5 Shapes . . . 186
18.5.1 Column Location Shape . . . 186
18.5.2 Row Location Shape . . . 186
18.5.3 Value Content Shape . . . 187
18.5.4 Value Location Shape . . . 187
19 Specification by Example Ontology . . . 188
19.1 Overview . . . 188
19.2 Classes . . . 188
19.2.1 Concept Value . . . 188
19.2.2 Graph . . . 189
19.2.3 Scenario . . . 189
19.2.4 State . . . 189
19.2.5 Story Request . . . 189
19.2.6 capture . . . 189
19.3 Properties . . . 190
19.3.1 Concept Value . . . 190
19.3.2 Dataset . . . 190
19.3.3 follows . . . 190
19.3.4 Given . . . 191
19.3.5 has Dataset . . . 191
19.3.6 has Graph . . . 191
19.3.7 Has Scenario . . . 191
19.3.8 ignore . . . 192
19.3.9 is depended on in state . . . 192
19.3.10 Is Given State Of . . . 192
19.3.11 is Graph in State . . . 192
19.3.12 Is Used In Graph . . . 193
19.3.13 Named Graph . . . 193
19.3.14 Story Parameters For Scenario . . . 193
19.3.15 Then . . . 194
19.3.16 Validates . . . 194
19.3.17 Value . . . 194
19.3.18 When . . . 194
19.4 Named Individuals . . . 195
19.4.1 any . . . 195
20 SHACL Shape Metadata Ontology . . . 195
20.1 Overview . . . 195
20.1.1 Shapesets and purpose . . . 195
20.2 Classes . . . 196
20.2.1 Shape or shapeset . . . 196
20.2.2 Shape purpose . . . 197
20.2.3 Shapeset . . . 197
20.3 Properties . . . 197
20.3.1 Has shape . . . 197
20.3.2 Has shapes graph . . . 198
20.3.3 In shapeset . . . 198
20.3.4 Shape purpose . . . 199
20.4 Named Individuals . . . 199
20.4.1 Validation . . . 199
20.4.2 SHACL Shape Metadata Validation Shapes . . . 199

EKGF Editor's Draft

20.5 Shapes . . . 200
20.5.1 Legacy Shape Membership Shape . . . 200
20.5.2 Shape Definition Shape . . . 200
20.5.3 Shape Purpose Shape . . . 201
20.5.4 Shape Purpose Target Shape . . . 201
20.5.5 Shapeset Coverage Shape . . . 201
20.5.6 Shapeset Shape . . . 202
20.5.7 Shapes Graph Link Shape . . . 203
21 Story Ontology . . . 204
21.1 Overview . . . 204
21.1.1 Stories and concept usage . . . 204
21.2 Classes . . . 206
21.2.1 Agent Job Implementation . . . 206
21.2.2 API Proxy Implementation . . . 206
21.2.3 Named Statement Implementation . . . 207
21.2.4 Story . . . 207
21.2.5 Story Concept Usage . . . 207
21.2.6 Story Dependent . . . 208
21.2.7 Story Implementation . . . 208
21.2.8 Story Input . . . 208
21.2.9 Story Output . . . 209
21.2.10 Story Stereotype . . . 209
21.2.11 Workflow Implementation . . . 210
21.3 Properties . . . 210
21.3.1 contributes to . . . 210
21.3.2 has concept usage . . . 211
21.3.3 has dependent concept . . . 211
21.3.4 has input concept . . . 211
21.3.5 has output concept . . . 212
21.3.6 has stereotype . . . 212
21.3.7 implementation of . . . 212
21.3.8 Implementation Type . . . 213
21.3.9 implemented by . . . 213
21.3.10 is deletion . . . 213
21.3.11 is mandatory . . . 214
21.3.12 Is Plural . . . 214
21.3.13 is server provided . . . 214
21.3.14 Is Singular . . . 215
21.3.15 Is Update . . . 215
21.3.16 not found on empty . . . 215
21.3.17 Played By . . . 216
21.3.18 provided by . . . 216
21.4 Named Individuals . . . 216
21.4.1 Query . . . 216
21.4.2 Transformation Rule . . . 217
21.4.3 Update . . . 217
21.4.4 Validation Rule . . . 217
21.4.5 Story Validation Shapes . . . 218
21.5 Shapes . . . 218
21.5.1 Legacy Concept Usage Shape . . . 218
21.5.2 Legacy Implementation Shape . . . 219

EKGF Editor's Draft

21.5.3 Story Classification Shape . . . 219
21.5.4 Story Concept Usage Shape . . . 219
21.5.5 Story Implementation Shape . . . 220
21.5.6 Story Input Shape . . . 221
21.5.7 Story Stereotype Shape . . . 221
22 SPARQL Story Implementation Ontology . . . 222
22.1 Overview . . . 222
22.1.1 Source language, renderer, and generated SPARQL . . . 222
22.2 Classes . . . 222
22.2.1 ASK . . . 223
22.2.2 CONSTRUCT . . . 223
22.2.3 DELETE . . . 223
22.2.4 DESCRIBE . . . 223
22.2.5 DROP . . . 224
22.2.6 RDFox SPARQL flavor . . . 224
22.2.7 SELECT . . . 224
22.2.8 SPARQL 1.0 . . . 224
22.2.9 SPARQL 1.1 . . . 225
22.2.10 SPARQL database product . . . 225
22.2.11 SPARQL database vendor . . . 225
22.2.12 SPARQL Flavor . . . 225
22.2.13 Stardog SPARQL flavor . . . 225
22.2.14 Statement Template . . . 226
22.2.15 SPARQL story implementation . . . 226
22.2.16 SPARQL query story implementation . . . 226
22.2.17 Story SPARQL Template Implementation . . . 226
22.2.18 SPARQL update story implementation . . . 227
22.2.19 Template Renderer . . . 227
22.2.20 UPDATE . . . 227
22.3 Properties . . . 228
22.3.1 File name . . . 228
22.3.2 Supports SPARQL Flavour . . . 228
22.3.3 has template . . . 228
22.3.4 SPARQL statement . . . 229
22.3.5 supports template language . . . 229
22.3.6 template file name . . . 229
22.3.7 template text . . . 229
22.3.8 uses renderer . . . 230
22.4 Named Individuals . . . 230
22.4.1 Allegro Graph . . . 230
22.4.2 Amazon Web Services . . . 230
22.4.3 Anzo Graph . . . 231
22.4.4 Blazegraph . . . 231
22.4.5 Cambridge Semantics . . . 231
22.4.6 GraphDB . . . 231
22.4.7 Neptune . . . 232
22.4.8 Ontotext . . . 232
22.4.9 Oxford Semantic Technologies . . . 232
22.4.10 RD Fox . . . 232
22.4.11 Stardog . . . 233
22.4.12 Stardog Union . . . 233

EKGF Editor's Draft

22.4.13 SPARQL Story Implementation Validation Shapes . . . 233
22.4.14 Virtuoso . . . 234
22.5 Shapes . . . 234
22.5.1 Statement Template Shape . . . 234
22.5.2 Story SPARQL Template Implementation Shape . . . 234
22.5.3 Template Renderer Shape . . . 235
23 Story Service Ontology . . . 236
23.1 Overview . . . 236
23.1.1 Executing stories . . . 236
23.2 Classes . . . 237
23.2.1 Parameter . . . 237
23.2.2 Result . . . 237
23.2.3 SHACL Statement . . . 237
23.2.4 SPARQL Parameter . . . 237
23.2.5 SPARQL Results . . . 238
23.2.6 SPARQL Statement . . . 238
23.2.7 Statement . . . 238
23.2.8 Story Service . . . 239
23.3 Properties . . . 239
23.3.1 Has Parameter . . . 239
23.3.2 Is Service Parameter . . . 239
23.3.3 On . . . 240
23.3.4 Parameter Direction . . . 240
23.3.5 Parameter For . . . 240
23.3.6 Parameter Key . . . 241
23.3.7 Parameter Op . . . 241
23.3.8 Parameter Type . . . 241
23.3.9 should be supplied by . . . 242
23.3.10 Result For . . . 242
23.3.11 Result Key . . . 242
23.3.12 Result Type . . . 243
23.3.13 Should Be Generated By . . . 243
23.3.14 Sparql Query Name . . . 243
23.3.15 Sparql Statement File Name . . . 244
24 Use Case Ontology . . . 244
24.1 Overview . . . 244
24.1.1 Use cases, stories, and concept vocabularies . . . 244
24.2 Classes . . . 244
24.2.1 Use Case . . . 244
24.2.2 Use Case Stereotype . . . 244
24.3 Properties . . . 245
24.3.1 has colour . . . 245
24.3.2 has outcome . . . 245
24.3.3 has stereotype . . . 245
24.3.4 hex color . . . 245
24.3.5 Involves persona . . . 246
24.3.6 Is involved in . . . 246
24.3.7 is part of . . . 246
24.3.8 is used in . . . 247
24.3.9 Is used in use case . . . 247
24.3.10 owns concept-set . . . 247

EKGF Editor's Draft

24.3.11 screen layout index . . . 248
24.3.12 Used in . . . 248
24.3.13 Uses concept . . . 248
24.4 Named Individuals . . . 249
24.4.1 Application . . . 249
24.4.2 Business Capability . . . 249
24.4.3 Business Capability Area . . . 249
24.4.4 Data Domain . . . 250
24.4.5 Data Landscape . . . 250
24.4.6 Data Subdomain . . . 250
24.4.7 Data Transformation . . . 251
24.4.8 Root . . . 251
24.4.9 Use Case . . . 251
25 User Experience Ontology . . . 252
25.1 Overview . . . 252
25.2 Properties . . . 252
25.2.1 Hexadecimal RGB color . . . 252
25.2.2 Sort key . . . 252
A Documentation coverage . . . 253

Abstract

A reference to the related EKGF ontologies. These vocabularies describe enterprise knowledge graphs, their 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 24 ontologies in this family. Select an ontology to read its vocabulary reference.

- **Annotation Ontology**
  Descriptive text that any resource can carry: a description in Markdown, a subtitle and a summary. These are annotation properties, so they apply to classes and properties as well as to individuals.
- **Platform API Ontology**
  The EKG Platform API Ontology.
- **Artifact Dependency Ontology**
  Generic dependency vocabulary for Semantic Modelling artifacts such as OWL ontologies, SHACL shapesets, 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 shapesets, 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 concepts for short, identify reusable meanings independently of their business and technical manifestations. The scope includes individual data elements and compound

EKGF Editor's Draft

data constructs such as whole classes and shapes. ManifestationOccurrence records where a manifestation appears in a source.

- **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.
- **Dataset Ontology**
  A vocabulary for RDF datasets and their source files, named graphs, identifiers, and composition into static or compound datasets.
- **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**
  The outcomes that use cases and stories contribute to. An outcome has a stereotype and a priority. Its lifecycle is 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.
- **SHACL Shape Metadata Ontology**
  Metadata and validation for grouping SHACL node and property shapes into shapesets, recording their purpose, and linking collections to concrete shapes graphs with separate identities.
- **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.
- **User Experience Ontology**
  This ontology covers various minor details, "hints", that can be used by User Interface programmers to optimize the user experience. Details like the default sorting of lists of objects, the default color of things etc.

2 Annotation Ontology

2.1 Overview

Annotation Ontology

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

Descriptive text that any resource can carry: a description in Markdown, a subtitle and a summary. These are annotation properties, so they apply to classes and properties as well as to individuals.

2.2 Properties

2.2.1 Markdown description

Label: Markdown description

Identifier: `ekg-annotation:markdownDescription`

2.2.1.1 Explanatory documentation

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.

IRI: `https://ekgf.org/ontology/annotation#markdownDescription`

RDF predicate	Value
`rdf:type`	`owl:AnnotationProperty`
`rdfs:isDefinedBy`	ekg-annotation:

2.2.2 subtitle

Label: subtitle

Identifier: `ekg-annotation:subtitle`

One line shown under the label of the resource.

IRI: `https://ekgf.org/ontology/annotation#subtitle`

RDF predicate	Value
`rdf:type`	`owl:AnnotationProperty`
`rdfs:isDefinedBy`	ekg-annotation:

2.2.3 summary

Label: summary

Identifier: `ekg-annotation:summary`

A short summary of the resource, in plain text.

IRI: `https://ekgf.org/ontology/annotation#summary`

RDF predicate	Value
`rdf:type`	`owl:AnnotationProperty`
`rdfs:isDefinedBy`	ekg-annotation:

3 Platform API Ontology

3.1 Overview

Platform API Ontology

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

The EKG Platform API Ontology.

3.2 Properties

3.2.1 slug

Label: slug

Identifier: `ekg-api:slug`

The short, lowercase, hyphenated name by which a resource is addressed in a URL. Replaces api:key.

IRI: `https://ekgf.org/ontology/api#slug`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`

RDF predicate	Value
`rdfs:isDefinedBy`	ekg-api:
`rdfs:range`	`xsd:string`

3.2.2 taxonomy kind

Label: taxonomy kind

Identifier: `ekg-api:taxonomyKind`

What kind of taxonomy a SKOS concept scheme is, such as thesaurus.

IRI: `https://ekgf.org/ontology/api#taxonomyKind`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:isDefinedBy`	ekg-api:
`rdfs:range`	`xsd:string`

4 Artifact Dependency Ontology

4.1 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.

4.2 Classes

4.2.1 Dependency

Label: Dependency

Identifier: `ekg-artdep:Dependency`

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

IRI: `https://ekgf.org/ontology/artifact-dependency#Dependency`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-artdep:

4.2.2 Dependency evidence

Label: Dependency evidence

Identifier: `ekg-artdep:DependencyEvidence`

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

IRI: `https://ekgf.org/ontology/artifact-dependency#DependencyEvidence`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-artdep:

4.2.3 Dependency kind

Label: Dependency kind

Identifier: `ekg-artdep:DependencyKind`

Controlled values describing why the dependency exists.

IRI: `https://ekgf.org/ontology/artifact-dependency#DependencyKind`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-artdep:

4.2.4 Dependency resolution

Label: Dependency resolution

Identifier: `ekg-artdep:DependencyResolution`

Workspace and branch scoped resolution state for a dependency target.

IRI: `https://ekgf.org/ontology/artifact-dependency#DependencyResolution`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-artdep:

4.2.5 Explicit dependency

Label: Explicit dependency

Identifier: `ekg-artdep:ExplicitDependency`

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

IRI: `https://ekgf.org/ontology/artifact-dependency#ExplicitDependency`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-artdep:
`rdfs:subClassOf`	ekg-artdep:Dependency

4.2.6 Implicit dependency

Label: Implicit dependency

Identifier: `ekg-artdep:ImplicitDependency`

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

IRI: `https://ekgf.org/ontology/artifact-dependency#ImplicitDependency`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-artdep:
`rdfs:subClassOf`	ekg-artdep:Dependency

4.2.7 Ontology load request

Label: Ontology load request

Identifier: `ekg-artdep:OntologyLoadRequest`

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.

IRI: `https://ekgf.org/ontology/artifact-dependency#OntologyLoadRequest`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-artdep:

4.2.8 Resolution status

Label: Resolution status

Identifier: `ekg-artdep:ResolutionStatus`

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

IRI: `https://ekgf.org/ontology/artifact-dependency#ResolutionStatus`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-artdep:

4.2.9 Semantic artifact

Label: Semantic artifact

Identifier: `ekg-artdep:SemanticArtifact`

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

IRI: `https://ekgf.org/ontology/artifact-dependency#SemanticArtifact`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-artdep:

4.3 Properties

4.3.1 dependency evidence

Label: dependency evidence

Identifier: `ekg-artdep:dependencyEvidence`

Links a dependency edge to one supporting evidence record.

IRI: `https://ekgf.org/ontology/artifact-dependency#dependencyEvidence`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-artdep:Dependency
`rdfs:isDefinedBy`	ekg-artdep:
`rdfs:range`	ekg-artdep:DependencyEvidence

4.3.2 dependency kind

Label: dependency kind

Identifier: `ekg-artdep:dependencyKind`

Classifies why the dependency exists.

IRI: `https://ekgf.org/ontology/artifact-dependency#dependencyKind`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-artdep:Dependency

RDF predicate	Value
rdfs:isDefinedBy	ekg-artdep:
rdfs:range	ekg-artdep:DependencyKind

4.3.3 dependency target

Label: dependency target

Identifier: `ekg-artdep:dependencyTarget`

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

IRI: `https://ekgf.org/ontology/artifact-dependency#dependencyTarget`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-artdep:Dependency
rdfs:isDefinedBy	ekg-artdep:
rdfs:range	rdfs:Resource

4.3.4 dependent artifact

Label: dependent artifact

Identifier: `ekg-artdep:dependentArtifact`

The artifact that owns the dependency edge.

IRI: `https://ekgf.org/ontology/artifact-dependency#dependentArtifact`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-artdep:Dependency
rdfs:isDefinedBy	ekg-artdep:
rdfs:range	ekg-artdep:SemanticArtifact

4.3.5 depends on

Label: depends on

Identifier: `ekg-artdep:dependsOn`

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

IRI: `https://ekgf.org/ontology/artifact-dependency#dependsOn`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-artdep:SemanticArtifact
rdfs:isDefinedBy	ekg-artdep:
rdfs:range	rdfs:Resource

4.3.6 evidence graph

Label: evidence graph

Identifier: `ekg-artdep:evidenceGraph`

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

IRI: `https://ekgf.org/ontology/artifact-dependency#evidenceGraph`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-artdep:DependencyEvidence
rdfs:isDefinedBy	ekg-artdep:
rdfs:range	rdfs:Resource

4.3.7 evidence object

Label: evidence object

Identifier: `ekg-artdep:evidenceObject`

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

IRI: `https://ekgf.org/ontology/artifact-dependency#evidenceObject`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-artdep:DependencyEvidence
rdfs:isDefinedBy	ekg-artdep:
rdfs:range	rdfs:Resource

4.3.8 evidence predicate

Label: evidence predicate

Identifier: `ekg-artdep:evidencePredicate`

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.

IRI: `https://ekgf.org/ontology/artifact-dependency#evidencePredicate`

RDF predicate	Value
rdf:type	owl:AnnotationProperty
rdfs:domain	ekg-artdep:DependencyEvidence
rdfs:isDefinedBy	ekg-artdep:
rdfs:range	rdf:Property

4.3.9 evidence subject

Label: evidence subject

Identifier: `ekg-artdep:evidenceSubject`

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

IRI: `https://ekgf.org/ontology/artifact-dependency#evidenceSubject`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-artdep:DependencyEvidence
rdfs:isDefinedBy	ekg-artdep:
rdfs:range	rdfs:Resource

4.3.10 has dependency

Label: has dependency

Identifier: `ekg-artdep:hasDependency`

Links a semantic artifact to a reified dependency edge.

IRI: `https://ekgf.org/ontology/artifact-dependency#hasDependency`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-artdep:SemanticArtifact
`rdfs:isDefinedBy`	ekg-artdep:
`rdfs:range`	ekg-artdep:Dependency

4.3.11 load hint

Label: load hint

Identifier: `ekg-artdep:loadHint`

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

IRI: `https://ekgf.org/ontology/artifact-dependency#loadHint`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-artdep:DependencyResolution
`rdfs:isDefinedBy`	ekg-artdep:
`rdfs:range`	`xsd:anyURI`

4.3.12 load request authority

Label: load request authority

Identifier: `ekg-artdep:loadRequestAuthority`

Authority governing the artifact family named by the load request.

IRI: `https://ekgf.org/ontology/artifact-dependency#loadRequestAuthority`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-artdep:OntologyLoadRequest
`rdfs:isDefinedBy`	ekg-artdep:
`rdfs:range`	`rdfs:Resource`

4.3.13 load request dependency

Label: load request dependency

Identifier: `ekg-artdep:loadRequestDependency`

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

IRI: `https://ekgf.org/ontology/artifact-dependency#loadRequestDependency`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-artdep:OntologyLoadRequest
`rdfs:isDefinedBy`	ekg-artdep:
`rdfs:range`	ekg-artdep:Dependency

4.3.14 load request family

Label: load request family

Identifier: `ekg-artdep:loadRequestFamily`

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

IRI: `https://ekgf.org/ontology/artifact-dependency#loadRequestFamily`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-artdep:OntologyLoadRequest
`rdfs:isDefinedBy`	ekg-artdep:
`rdfs:range`	`rdfs:Resource`

4.3.15 load request target

Label: load request target

Identifier: `ekg-artdep:loadRequestTarget`

Dependency target IRI that caused this load request.

IRI: `https://ekgf.org/ontology/artifact-dependency#loadRequestTarget`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-artdep:OntologyLoadRequest
`rdfs:isDefinedBy`	ekg-artdep:
`rdfs:range`	`rdfs:Resource`

4.3.16 referenced namespace

Label: referenced namespace

Identifier: `ekg-artdep:referencedNamespace`

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

IRI: `https://ekgf.org/ontology/artifact-dependency#referencedNamespace`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-artdep:DependencyEvidence
`rdfs:isDefinedBy`	ekg-artdep:
`rdfs:range`	`rdfs:Resource`

4.3.17 resolution

Label: resolution

Identifier: `ekg-artdep:resolution`

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

IRI: `https://ekgf.org/ontology/artifact-dependency#resolution`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-artdep:Dependency
`rdfs:isDefinedBy`	ekg-artdep:
`rdfs:range`	ekg-artdep:DependencyResolution

4.3.18 resolution branch

Label: resolution branch

Identifier: `ekg-artdep:resolutionBranch`

Branch context in which the dependency resolution was computed.

IRI: `https://ekgf.org/ontology/artifact-dependency#resolutionBranch`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-artdep:DependencyResolution
`rdfs:isDefinedBy`	ekg-artdep:
`rdfs:range`	`rdfs:Resource`

4.3.19 resolution commit

Label: resolution commit

Identifier: `ekg-artdep:resolutionCommit`

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

IRI: `https://ekgf.org/ontology/artifact-dependency#resolutionCommit`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-artdep:DependencyResolution
`rdfs:isDefinedBy`	ekg-artdep:
`rdfs:range`	`rdfs:Resource`

4.3.20 resolution message

Label: resolution message

Identifier: `ekg-artdep:resolutionMessage`

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

IRI: `https://ekgf.org/ontology/artifact-dependency#resolutionMessage`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-artdep:DependencyResolution
`rdfs:isDefinedBy`	ekg-artdep:
`rdfs:range`	`xsd:string`

4.3.21 resolution status

Label: resolution status

Identifier: `ekg-artdep:resolutionStatus`

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

IRI: `https://ekgf.org/ontology/artifact-dependency#resolutionStatus`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-artdep:DependencyResolution
`rdfs:isDefinedBy`	ekg-artdep:
`rdfs:range`	ekg-artdep:ResolutionStatus

4.3.22 resolution workspace

Label: resolution workspace

Identifier: `ekg-artdep:resolutionWorkspace`
Workspace context in which the dependency resolution was computed.
IRI: `https://ekgf.org/ontology/artifact-dependency#resolutionWorkspace`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-artdep:DependencyResolution
`rdfs:isDefinedBy`	ekg-artdep:
`rdfs:range`	`rdfs:Resource`

4.3.23 resolved artifact
Label: resolved artifact
Identifier: `ekg-artdep:resolvedArtifact`
The loaded artifact that satisfies the dependency target, when resolution status is Loaded.
IRI: `https://ekgf.org/ontology/artifact-dependency#resolvedArtifact`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-artdep:DependencyResolution
`rdfs:isDefinedBy`	ekg-artdep:
`rdfs:range`	ekg-artdep:SemanticArtifact

4.4 Named Individuals
4.4.1 Ambiguous
Label: Ambiguous
Identifier: `ekg-artdep:Ambiguous`
Instance of: ekg-artdep:ResolutionStatus
The dependency target matches more than one candidate artifact in the current workspace context.
IRI: `https://ekgf.org/ontology/artifact-dependency#Ambiguous`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-artdep:ResolutionStatus
`rdfs:isDefinedBy`	ekg-artdep:

4.4.2 Declared import dependency
Label: Declared import dependency
Identifier: `ekg-artdep:DeclaredImportDependency`
Instance of: ekg-artdep:DependencyKind
A dependency declared by an import-like predicate such as owl:imports.
IRI: `https://ekgf.org/ontology/artifact-dependency#DeclaredImportDependency`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-artdep:DependencyKind
`rdfs:isDefinedBy`	ekg-artdep:

4.4.3 External

Label: External

Identifier: `ekg-artdep:External`

Instance of: ekg-artdep:ResolutionStatus

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

IRI: `https://ekgf.org/ontology/artifact-dependency#External`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-artdep:ResolutionStatus
`rdfs:isDefinedBy`	ekg-artdep:

4.4.4 Failed

Label: Failed

Identifier: `ekg-artdep:Failed`

Instance of: ekg-artdep:ResolutionStatus

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

IRI: `https://ekgf.org/ontology/artifact-dependency#Failed`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-artdep:ResolutionStatus
`rdfs:isDefinedBy`	ekg-artdep:

4.4.5 Incomplete

Label: Incomplete

Identifier: `ekg-artdep:Incomplete`

Instance of: ekg-artdep:ResolutionStatus

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

IRI: `https://ekgf.org/ontology/artifact-dependency#Incomplete`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-artdep:ResolutionStatus
`rdfs:isDefinedBy`	ekg-artdep:

4.4.6 Loaded

Label: Loaded

Identifier: `ekg-artdep:Loaded`

Instance of: ekg-artdep:ResolutionStatus

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

IRI: `https://ekgf.org/ontology/artifact-dependency#Loaded`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-artdep:ResolutionStatus
`rdfs:isDefinedBy`	ekg-artdep:

4.4.7 Namespace reference dependency

Label: Namespace reference dependency

Identifier: `ekg-artdep:NamespaceReferenceDependency`

Instance of: ekg-artdep:DependencyKind

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

IRI: `https://ekgf.org/ontology/artifact-dependency#NamespaceReferenceDependency`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-artdep:DependencyKind
`rdfs:isDefinedBy`	ekg-artdep:

4.4.8 Profile dependency

Label: Profile dependency

Identifier: `ekg-artdep:ProfileDependency`

Instance of: ekg-artdep:DependencyKind

A dependency on a standards or implementation profile.

IRI: `https://ekgf.org/ontology/artifact-dependency#ProfileDependency`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-artdep:DependencyKind
`rdfs:isDefinedBy`	ekg-artdep:

4.4.9 Shape graph dependency

Label: Shape graph dependency

Identifier: `ekg-artdep:ShapeGraphDependency`

Instance of: ekg-artdep:DependencyKind

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

IRI: `https://ekgf.org/ontology/artifact-dependency#ShapeGraphDependency`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-artdep:DependencyKind
`rdfs:isDefinedBy`	ekg-artdep:

4.4.10 Term reference dependency

Label: Term reference dependency

Identifier: `ekg-artdep:TermReferenceDependency`

Instance of: ekg-artdep:DependencyKind

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

IRI: `https://ekgf.org/ontology/artifact-dependency#TermReferenceDependency`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-artdep:DependencyKind
`rdfs:isDefinedBy`	ekg-artdep:

4.4.11 Unloaded

Label: Unloaded

Identifier: `ekg-artdep:Unloaded`

Instance of: ekg-artdep:ResolutionStatus

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

IRI: `https://ekgf.org/ontology/artifact-dependency#Unloaded`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-artdep:ResolutionStatus
`rdfs:isDefinedBy`	ekg-artdep:

4.4.12 Unsupported

Label: Unsupported

Identifier: `ekg-artdep:Unsupported`

Instance of: ekg-artdep:ResolutionStatus

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

IRI: `https://ekgf.org/ontology/artifact-dependency#Unsupported`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-artdep:ResolutionStatus
`rdfs:isDefinedBy`	ekg-artdep:

4.4.13 Validation dependency

Label: Validation dependency

Identifier: `ekg-artdep:ValidationDependency`

Instance of: ekg-artdep:DependencyKind

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

IRI: `https://ekgf.org/ontology/artifact-dependency#ValidationDependency`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-artdep:DependencyKind
`rdfs:isDefinedBy`	ekg-artdep:

5 Artifact Governance Ontology

5.1 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 shapesets, SKOS taxonomies and concept sets. Replaces the Ontology Ontology (https://ekgf.org/ontology/ontology#).

5.2 Classes

5.2.1 Authority

Label: Authority

Identifier: `ekg-artgov: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.

IRI: `https://ekgf.org/ontology/artifact-governance#Authority`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-artgov:
`rdfs:subClassOf`	`cmns-pts:PartyRole`

5.2.2 Authority publishing role

Label: Authority publishing role

Identifier: `ekg-artgov:AuthorityPublishingRole`

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

IRI: `https://ekgf.org/ontology/artifact-governance#AuthorityPublishingRole`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-artgov:

5.2.3 Family

Label: Family

Identifier: `ekg-artgov:Family`

An optional, enduring collection of related artifacts governed by one or more authorities, such as FIBO. A family spans artifact types: its ontologies, shapesets 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.

IRI: `https://ekgf.org/ontology/artifact-governance#Family`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-artgov:

5.2.4 Organization category

Label: Organization category

Identifier: `ekg-artgov:OrganizationCategory`

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.

IRI: `https://ekgf.org/ontology/artifact-governance#OrganizationCategory`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-artgov:
`rdfs:subClassOf`	`cmns-cls:Classifier`
`owl:disjointWith`	ekg-artgov:AuthorityPublishingRole
`owl:disjointWith`	ekg-artgov:StandardsCommunityRole

5.2.5 Reference

Label: Reference

Identifier: `ekg-artgov: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.

IRI: `https://ekgf.org/ontology/artifact-governance#Reference`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-artgov:

5.2.6 Standards community role

Label: Standards community role

Identifier: `ekg-artgov:StandardsCommunityRole`

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

IRI: `https://ekgf.org/ontology/artifact-governance#StandardsCommunityRole`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-artgov:

5.3 Properties

5.3.1 Authority acronym

Label: Authority acronym

Identifier: `ekg-artgov:authorityAcronym`

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

IRI: `https://ekgf.org/ontology/artifact-governance#authorityAcronym`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-artgov:Authority
`rdfs:isDefinedBy`	ekg-artgov:
`rdfs:range`	`xsd:string`

5.3.2 Authority host

Label: Authority host

Identifier: `ekg-artgov:authorityHost`

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.

IRI: `https://ekgf.org/ontology/artifact-governance#authorityHost`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-artgov:Authority
`rdfs:isDefinedBy`	ekg-artgov:
`rdfs:range`	`xsd:string`

5.3.3 Auto-load

Label: Auto-load

Identifier: `ekg-artgov:autoLoad`

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

IRI: `https://ekgf.org/ontology/artifact-governance#autoLoad`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-artgov:Authority
`rdfs:isDefinedBy`	ekg-artgov:
`rdfs:range`	`xsd:boolean`

5.3.4 Base IRI

Label: Base IRI

Identifier: `ekg-artgov:baseIri`

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.

IRI: `https://ekgf.org/ontology/artifact-governance#baseIri`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-artgov:Authority
`rdfs:isDefinedBy`	ekg-artgov:
`rdfs:range`	`xsd:anyURI`

5.3.5 Community organization

Label: Community organization

Identifier: `ekg-artgov:communityOrganization`

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

IRI: `https://ekgf.org/ontology/artifact-governance#communityOrganization`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-artgov:Authority
`rdfs:isDefinedBy`	ekg-artgov:

RDF predicate	Value
rdfs:range	cmns-org:Organization

5.3.6 Community role

Label: Community role

Identifier: ekg-artgov:communityRole

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

IRI: https://ekgf.org/ontology/artifact-governance#communityRole

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-artgov:Authority
rdfs:isDefinedBy	ekg-artgov:
rdfs:range	ekg-artgov:StandardsCommunityRole

5.3.7 Family acronym

Label: Family acronym

Identifier: ekg-artgov:familyAcronym

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

IRI: https://ekgf.org/ontology/artifact-governance#familyAcronym

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-artgov:Family
rdfs:isDefinedBy	ekg-artgov:
rdfs:range	xsd:string

5.3.8 File path prefix

Label: File path prefix

Identifier: ekg-artgov:filePathPrefix

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.

IRI: https://ekgf.org/ontology/artifact-governance#filePathPrefix

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-artgov:Family
rdfs:isDefinedBy	ekg-artgov:
rdfs:range	xsd:string

5.3.9 Git branch

Label: Git branch

Identifier: `ekg-artgov:gitBranch`
The git branch or tag to clone.
IRI: `https://ekgf.org/ontology/artifact-governance#gitBranch`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-artgov:Family
`rdfs:isDefinedBy`	ekg-artgov:
`rdfs:range`	`xsd:string`

5.3.10 Git path
Label: Git path
Identifier: `ekg-artgov:gitPath`
Directory inside the git repository to scan for RDF files. The repository root when absent.
IRI: `https://ekgf.org/ontology/artifact-governance#gitPath`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-artgov:Family
`rdfs:isDefinedBy`	ekg-artgov:
`rdfs:range`	`xsd:string`

5.3.11 Git repository URL
Label: Git repository URL
Identifier: `ekg-artgov:gitRepositoryUrl`
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.
IRI: `https://ekgf.org/ontology/artifact-governance#gitRepositoryUrl`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-artgov:Family
`rdfs:isDefinedBy`	ekg-artgov:
`rdfs:range`	`xsd:anyURI`

5.3.12 Governed by
Label: Governed by
Identifier: `ekg-artgov:governedBy`
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.
IRI: `https://ekgf.org/ontology/artifact-governance#governedBy`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:isDefinedBy`	ekg-artgov:
`rdfs:range`	ekg-artgov:Authority

5.3.13 Graph IRI prefix

Label: Graph IRI prefix

Identifier: `ekg-artgov:graphIriPrefix`

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.

IRI: `https://ekgf.org/ontology/artifact-governance#graphIriPrefix`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-artgov:Family
`rdfs:isDefinedBy`	ekg-artgov:
`rdfs:range`	`xsd:anyURI`

5.3.14 Historical name

Label: Historical name

Identifier: `ekg-artgov:historicalName`

A former public name of an authority or organization.

IRI: `https://ekgf.org/ontology/artifact-governance#historicalName`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:isDefinedBy`	ekg-artgov:
`rdfs:range`	`xsd:string`

5.3.15 In family

Label: In family

Identifier: `ekg-artgov:inFamily`

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.

IRI: `https://ekgf.org/ontology/artifact-governance#inFamily`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:isDefinedBy`	ekg-artgov:
`rdfs:range`	ekg-artgov:Family

5.3.16 Legal owner

Label: Legal owner

Identifier: `ekg-artgov:legalOwner`

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.

IRI: `https://ekgf.org/ontology/artifact-governance#legalOwner`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:isDefinedBy`	ekg-artgov:
`rdfs:range`	`cmns-org:Organization`

5.3.17 Maximum family load quads

Label: Maximum family load quads

Identifier: `ekg-artgov:maxFamilyLoadQuads`

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

IRI: `https://ekgf.org/ontology/artifact-governance#maxFamilyLoadQuads`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-artgov:Family
`rdfs:isDefinedBy`	ekg-artgov:
`rdfs:range`	`xsd:unsignedLong`

5.3.18 Maximum source quads

Label: Maximum source quads

Identifier: `ekg-artgov:maxSourceQuads`

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

IRI: `https://ekgf.org/ontology/artifact-governance#maxSourceQuads`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-artgov:Family
`rdfs:isDefinedBy`	ekg-artgov:
`rdfs:range`	`xsd:unsignedLong`

5.3.19 Namespace IRI prefix

Label: Namespace IRI prefix

Identifier: `ekg-artgov:namespaceIriPrefix`

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.

IRI: `https://ekgf.org/ontology/artifact-governance#namespaceIriPrefix`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-artgov:Family
`rdfs:isDefinedBy`	ekg-artgov:
`rdfs:range`	`xsd:anyURI`

5.3.20 Organization category

Label: Organization category

Identifier: `ekg-artgov:organizationCategory`

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.

IRI: `https://ekgf.org/ontology/artifact-governance#organizationCategory`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	`cmns-org:Organization`
`rdfs:isDefinedBy`	ekg-artgov:
`rdfs:range`	ekg-artgov:OrganizationCategory
`rdfs:subPropertyOf`	`cmns-cls:isClassifiedBy`

5.3.21 Predecessor authority

Label: Predecessor authority

Identifier: `ekg-artgov:predecessorAuthority`

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

IRI: `https://ekgf.org/ontology/artifact-governance#predecessorAuthority`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-artgov:Authority
`rdfs:isDefinedBy`	ekg-artgov:
`rdfs:range`	ekg-artgov:Authority

5.3.22 Publishing role

Label: Publishing role

Identifier: `ekg-artgov:publishingRole`

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

IRI: `https://ekgf.org/ontology/artifact-governance#publishingRole`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-artgov:Authority
`rdfs:isDefinedBy`	ekg-artgov:
`rdfs:range`	ekg-artgov:AuthorityPublishingRole

5.3.23 RDF file extension

Label: RDF file extension

Identifier: `ekg-artgov:rdfFileExtension`

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

IRI: `https://ekgf.org/ontology/artifact-governance#rdfFileExtension`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-artgov:Family

RDF predicate	Value
rdfs:isDefinedBy	ekg-artgov:
rdfs:range	xsd:string

5.3.24 Refers to

Label: Refers to

Identifier: `ekg-artgov:refersTo`

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.

IRI: `https://ekgf.org/ontology/artifact-governance#refersTo`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-artgov:Reference
rdfs:isDefinedBy	ekg-artgov:

5.3.25 Source graph repository URL

Label: Source graph repository URL

Identifier: `ekg-artgov:sourceGraphRepositoryUrl`

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.

IRI: `https://ekgf.org/ontology/artifact-governance#sourceGraphRepositoryUrl`

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-artgov:Family
rdfs:isDefinedBy	ekg-artgov:
rdfs:range	xsd:anyURI

5.3.26 Successor authority

Label: Successor authority

Identifier: `ekg-artgov:successorAuthority`

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

IRI: `https://ekgf.org/ontology/artifact-governance#successorAuthority`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-artgov:Authority
rdfs:isDefinedBy	ekg-artgov:
rdfs:range	ekg-artgov:Authority

5.4 Named Individuals

5.4.1 co-owner

Label: co-owner

Identifier: `ekg-artgov:AuthorityCoOwnerRole`

Instance of: ekg-artgov:AuthorityPublishingRole

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

IRI: `https://ekgf.org/ontology/artifact-governance#AuthorityCoOwnerRole`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-artgov:AuthorityPublishingRole
rdfs:isDefinedBy	ekg-artgov:
ekg-api:slug	co-owner

5.4.2 maintainer

Label: maintainer

Identifier: `ekg-artgov:AuthorityMaintainerRole`

Instance of: ekg-artgov:AuthorityPublishingRole

The authority maintains the artifacts between publication releases.

IRI: `https://ekgf.org/ontology/artifact-governance#AuthorityMaintainerRole`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-artgov:AuthorityPublishingRole
rdfs:isDefinedBy	ekg-artgov:
ekg-api:slug	maintainer

5.4.3 owner

Label: owner

Identifier: `ekg-artgov:AuthorityOwnerRole`

Instance of: ekg-artgov:AuthorityPublishingRole

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

IRI: `https://ekgf.org/ontology/artifact-governance#AuthorityOwnerRole`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-artgov:AuthorityPublishingRole
rdfs:isDefinedBy	ekg-artgov:
ekg-api:slug	owner

5.4.4 publisher

Label: publisher

Identifier: `ekg-artgov:AuthorityPublisherRole`

Instance of: ekg-artgov:AuthorityPublishingRole

The authority publishes the artifacts or their specification site.

IRI: `https://ekgf.org/ontology/artifact-governance#AuthorityPublisherRole`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-artgov:AuthorityPublishingRole
rdfs:isDefinedBy	ekg-artgov:
ekg-api:slug	publisher

5.4.5 Community

Label: Community

Identifier: `ekg-artgov:CommunityCategory`
Instance of: ekg-artgov:OrganizationCategory
An organized community whose participants collaborate around a shared topic or practice.
IRI: `https://ekgf.org/ontology/artifact-governance#CommunityCategory`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-artgov:OrganizationCategory
`rdfs:isDefinedBy`	ekg-artgov:
`ekg-api:slug`	community
`cmns-dsg:isDefinedIn`	ekg-artgov:OrganizationCategoryScheme

5.4.6 Consortium
Label: Consortium
Identifier: `ekg-artgov:ConsortiumCategory`
Instance of: ekg-artgov:OrganizationCategory
An organization through which independent organizations cooperate toward shared objectives.
IRI: `https://ekgf.org/ontology/artifact-governance#ConsortiumCategory`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-artgov:OrganizationCategory
`rdfs:isDefinedBy`	ekg-artgov:
`ekg-api:slug`	consortium
`cmns-dsg:isDefinedIn`	ekg-artgov:OrganizationCategoryScheme

5.4.7 Organization categories for artifact governance
Label: Organization categories for artifact governance
Identifier: `ekg-artgov:OrganizationCategoryScheme`
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.
IRI: `https://ekgf.org/ontology/artifact-governance#OrganizationCategoryScheme`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	`cmns-cls:ClassificationScheme`
`rdfs:isDefinedBy`	ekg-artgov:
`cmns-cls:isExclusive`	false

5.4.8 Project
Label: Project
Identifier: `ekg-artgov:ProjectCategory`
Instance of: ekg-artgov:OrganizationCategory
An organized group carrying out a defined initiative and producing its work products.
IRI: `https://ekgf.org/ontology/artifact-governance#ProjectCategory`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-artgov:OrganizationCategory
rdfs:isDefinedBy	ekg-artgov:
ekg-api:slug	project
cmns-dsg:isDefinedIn	ekg-artgov:OrganizationCategoryScheme

5.4.9 Standards body

Label: Standards body

Identifier: `ekg-artgov:StandardsBodyCategory`

Instance of: ekg-artgov:OrganizationCategory

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

IRI: `https://ekgf.org/ontology/artifact-governance#StandardsBodyCategory`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-artgov:OrganizationCategory
rdfs:isDefinedBy	ekg-artgov:
ekg-api:slug	standards-body
cmns-dsg:isDefinedIn	ekg-artgov:OrganizationCategoryScheme

5.4.10 standards editor

Label: standards editor

Identifier: `ekg-artgov:StandardsEditorRole`

Instance of: ekg-artgov:StandardsCommunityRole

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

IRI: `https://ekgf.org/ontology/artifact-governance#StandardsEditorRole`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-artgov:StandardsCommunityRole
rdfs:isDefinedBy	ekg-artgov:
ekg-api:slug	standards-editor

5.4.11 standards publisher

Label: standards publisher

Identifier: `ekg-artgov:StandardsPublisherRole`

Instance of: ekg-artgov:StandardsCommunityRole

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

IRI: `https://ekgf.org/ontology/artifact-governance#StandardsPublisherRole`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-artgov:StandardsCommunityRole

RDF predicate	Value
rdfs:isDefinedBy	ekg-artgov:
ekg-api:slug	standards-publisher

5.4.12 standards reviewer

Label: standards reviewer

Identifier: ekg-artgov:StandardsReviewerRole

Instance of: ekg-artgov:StandardsCommunityRole

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

IRI: https://ekgf.org/ontology/artifact-governance#StandardsReviewerRole

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-artgov:StandardsCommunityRole
rdfs:isDefinedBy	ekg-artgov:
ekg-api:slug	standards-reviewer

5.4.13 standards staff

Label: standards staff

Identifier: ekg-artgov:StandardsStaffRole

Instance of: ekg-artgov:StandardsCommunityRole

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

IRI: https://ekgf.org/ontology/artifact-governance#StandardsStaffRole

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-artgov:StandardsCommunityRole
rdfs:isDefinedBy	ekg-artgov:
ekg-api:slug	standards-staff

5.4.14 working-group chair

Label: working-group chair

Identifier: ekg-artgov:WorkingGroupChairRole

Instance of: ekg-artgov:StandardsCommunityRole

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

IRI: https://ekgf.org/ontology/artifact-governance#WorkingGroupChairRole

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-artgov:StandardsCommunityRole
rdfs:isDefinedBy	ekg-artgov:
ekg-api:slug	working-group-chair

6 Artifact Publication Ontology

6.1 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.

6.2 Classes

6.2.1 Publication artifact

Label: Publication artifact

Identifier: `ekg-artpub:PublicationArtifact`

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

IRI: `https://ekgf.org/ontology/artifact-publication#PublicationArtifact`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-artpub:

6.2.2 Publication profile

Label: Publication profile

Identifier: `ekg-artpub:PublicationProfile`

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.

IRI: `https://ekgf.org/ontology/artifact-publication#PublicationProfile`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-artpub:

6.2.3 Publication run

Label: Publication run

Identifier: `ekg-artpub:PublicationRun`

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.

IRI: `https://ekgf.org/ontology/artifact-publication#PublicationRun`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-artpub:

6.2.4 Publication target

Label: Publication target

Identifier: `ekg-artpub:PublicationTarget`

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

IRI: `https://ekgf.org/ontology/artifact-publication#PublicationTarget`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-artpub:

6.2.5 Release approval

Label: Release approval

Identifier: `ekg-artpub:ReleaseApproval`

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.

IRI: `https://ekgf.org/ontology/artifact-publication#ReleaseApproval`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-artpub:
`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

6.2.6 Standards release

Label: Standards release

Identifier: `ekg-artpub:StandardsRelease`

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.

IRI: `https://ekgf.org/ontology/artifact-publication#StandardsRelease`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-artpub:
`owl:disjointWith`	ekg-artgov:Family
`owl:disjointWith`	ekg-artgov:Reference
`owl:disjointWith`	ekg-artpub:PublicationArtifact
`owl:disjointWith`	ekg-artpub:PublicationProfile

RDF predicate	Value
owl:disjointWith	ekg-artpub:PublicationRun

6.3 Properties

6.3.1 Approved at

Label: Approved at

Identifier: `ekg-artpub:approvedAt`

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

IRI: `https://ekgf.org/ontology/artifact-publication#approvedAt`

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-artpub:ReleaseApproval
rdfs:isDefinedBy	ekg-artpub:
rdfs:range	xsd:dateTime

6.3.2 Approved by

Label: Approved by

Identifier: `ekg-artpub:approvedBy`

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.

IRI: `https://ekgf.org/ontology/artifact-publication#approvedBy`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-artpub:ReleaseApproval
rdfs:isDefinedBy	ekg-artpub:
rdfs:range	ekg-artgov:Authority

6.3.3 Approves release

Label: Approves release

Identifier: `ekg-artpub:approvesRelease`

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

IRI: `https://ekgf.org/ontology/artifact-publication#approvesRelease`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-artpub:ReleaseApproval
rdfs:isDefinedBy	ekg-artpub:
rdfs:range	ekg-artpub:StandardsRelease

6.3.4 Artifact checksum

Label: Artifact checksum

Identifier: `ekg-artpub:artifactChecksum`

Checksum or digest of a generated publication artifact.

IRI: `https://ekgf.org/ontology/artifact-publication#artifactChecksum`

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-artpub:PublicationArtifact
rdfs:isDefinedBy	ekg-artpub:
rdfs:range	xsd:string

6.3.5 Artifact media type

Label: Artifact media type

Identifier: `ekg-artpub:artifactMediaType`

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

IRI: `https://ekgf.org/ontology/artifact-publication#artifactMediaType`

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-artpub:PublicationArtifact
rdfs:isDefinedBy	ekg-artpub:
rdfs:range	xsd:string

6.3.6 Artifact URL

Label: Artifact URL

Identifier: `ekg-artpub:artifactUrl`

Stable URL where a generated publication artifact can be read.

IRI: `https://ekgf.org/ontology/artifact-publication#artifactUrl`

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-artpub:PublicationArtifact
rdfs:isDefinedBy	ekg-artpub:
rdfs:range	xsd:anyURI

6.3.7 Generator version

Label: Generator version

Identifier: `ekg-artpub:generatorVersion`

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

IRI: `https://ekgf.org/ontology/artifact-publication#generatorVersion`

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdfs:isDefinedBy	ekg-artpub:
rdfs:range	xsd:string

6.3.8 Input commit CID

Label: Input commit CID

Identifier: `ekg-artpub:inputCommitCid`

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

IRI: `https://ekgf.org/ontology/artifact-publication#inputCommitCid`

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdfs:isDefinedBy	ekg-artpub:
rdfs:range	xsd:string

6.3.9 Publication artifact

Label: Publication artifact

Identifier: `ekg-artpub:publicationArtifact`

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

IRI: `https://ekgf.org/ontology/artifact-publication#publicationArtifact`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:isDefinedBy	ekg-artpub:
rdfs:range	ekg-artpub:PublicationArtifact

6.3.10 Publication branch

Label: Publication branch

Identifier: `ekg-artpub:publicationBranch`

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

IRI: `https://ekgf.org/ontology/artifact-publication#publicationBranch`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:isDefinedBy	ekg-artpub:
rdfs:range	rdfs:Resource

6.3.11 Publication completed at

Label: Publication completed at

Identifier: `ekg-artpub:publicationCompletedAt`

Timestamp at which a publication run completed.

IRI: `https://ekgf.org/ontology/artifact-publication#publicationCompletedAt`

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-artpub:PublicationRun
rdfs:isDefinedBy	ekg-artpub:
rdfs:range	xsd:dateTime

6.3.12 Publication profile

Label: Publication profile

Identifier: `ekg-artpub:publicationProfile`

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

IRI: `https://ekgf.org/ontology/artifact-publication#publicationProfile`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:isDefinedBy	ekg-artpub:

RDF predicate	Value
rdfs:range	ekg-artpub:PublicationProfile

6.3.13 Publication release

Label: Publication release

Identifier: `ekg-artpub:publicationRelease`

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

IRI: `https://ekgf.org/ontology/artifact-publication#publicationRelease`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:isDefinedBy	ekg-artpub:
rdfs:range	ekg-artpub:StandardsRelease

6.3.14 Publication run

Label: Publication run

Identifier: `ekg-artpub:publicationRun`

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

IRI: `https://ekgf.org/ontology/artifact-publication#publicationRun`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:isDefinedBy	ekg-artpub:
rdfs:range	ekg-artpub:PublicationRun

6.3.15 Publication started at

Label: Publication started at

Identifier: `ekg-artpub:publicationStartedAt`

Timestamp at which a publication run started.

IRI: `https://ekgf.org/ontology/artifact-publication#publicationStartedAt`

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-artpub:PublicationRun
rdfs:isDefinedBy	ekg-artpub:
rdfs:range	xsd:dateTime

6.3.16 Publication target

Label: Publication target

Identifier: `ekg-artpub:publicationTarget`

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

IRI: `https://ekgf.org/ontology/artifact-publication#publicationTarget`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:isDefinedBy	ekg-artpub:
rdfs:range	ekg-artpub:PublicationTarget

6.3.17 Publishes family

Label: Publishes family

Identifier: `ekg-artpub:publishesFamily`

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

IRI: `https://ekgf.org/ontology/artifact-publication#publishesFamily`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-artpub:PublicationProfile
`rdfs:isDefinedBy`	ekg-artpub:
`rdfs:range`	ekg-artgov:Family

6.3.18 Publishes reference

Label: Publishes reference

Identifier: `ekg-artpub:publishesReference`

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.

IRI: `https://ekgf.org/ontology/artifact-publication#publishesReference`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-artpub:PublicationProfile
`rdfs:isDefinedBy`	ekg-artpub:
`rdfs:range`	ekg-artgov:Reference

6.3.19 Release family

Label: Release family

Identifier: `ekg-artpub:releaseFamily`

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.

IRI: `https://ekgf.org/ontology/artifact-publication#releaseFamily`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-artpub:StandardsRelease
`rdfs:isDefinedBy`	ekg-artpub:
`rdfs:range`	ekg-artgov:Family

6.3.20 Release reference

Label: Release reference

Identifier: `ekg-artpub:releaseReference`

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.

IRI: `https://ekgf.org/ontology/artifact-publication#releaseReference`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-artpub:StandardsRelease
rdfs:isDefinedBy	ekg-artpub:
rdfs:range	ekg-artgov:Reference

6.3.21 Release version

Label: Release version

Identifier: `ekg-artpub:releaseVersion`

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.

IRI: `https://ekgf.org/ontology/artifact-publication#releaseVersion`

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-artpub:StandardsRelease
rdfs:isDefinedBy	ekg-artpub:
rdfs:range	xsd:string

6.3.22 Released at

Label: Released at

Identifier: `ekg-artpub:releasedAt`

Timestamp at which a standards release was published.

IRI: `https://ekgf.org/ontology/artifact-publication#releasedAt`

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-artpub:StandardsRelease
rdfs:isDefinedBy	ekg-artpub:
rdfs:range	xsd:dateTime

6.3.23 Target kind

Label: Target kind

Identifier: `ekg-artpub:targetKind`

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

IRI: `https://ekgf.org/ontology/artifact-publication#targetKind`

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-artpub:PublicationTarget
rdfs:isDefinedBy	ekg-artpub:
rdfs:range	xsd:string

6.3.24 Target URL

Label: Target URL

Identifier: `ekg-artpub:targetUrl`

Base URL or destination URL of a publication target.

IRI: `https://ekgf.org/ontology/artifact-publication#targetUrl`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-artpub:PublicationTarget
`rdfs:isDefinedBy`	ekg-artpub:
`rdfs:range`	`xsd:anyURI`

6.4 Named Individuals

6.4.1 Artifact Publication Validation Shapes

Label: Artifact Publication Validation Shapes

Identifier: `ekg-artpub:ValidationShapeSet`

Instance of: ekg-shape:ShapeSet

Validates publication profiles, release scope and provenance, and release-approval records.

IRI: `https://ekgf.org/ontology/artifact-publication#ValidationShapeSet`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-shape:ShapeSet
`rdfs:isDefinedBy`	ekg-artpub:
`ekg-shape:hasShape`	ekg-artpub:PublicationProfileShape
`ekg-shape:hasShape`	ekg-artpub:ReleaseApprovalShape
`ekg-shape:hasShape`	ekg-artpub:StandardsReleaseShape
`ekg-shape:hasShapesGraph`	ekg-artpub:
`ekg-shape:shapePurpose`	ekg-shape:Validation

6.5 Shapes

6.5.1 Publication Profile Shape

Label: Publication Profile Shape

Identifier: `ekg-artpub:PublicationProfileShape`

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

IRI: `https://ekgf.org/ontology/artifact-publication#PublicationProfileShape`

RDF predicate	Value
`rdf:type`	`sh:NodeShape`
`rdfs:isDefinedBy`	ekg-artpub:
`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:Reference; `sh:nodeKind`: `sh:IRI`; `sh:path`: ekg-artpub:publishesReference)

RDF predicate	Value
sh:property	(sh:class: ekg-artgov:Family; sh:maxCount: 1; sh:nodeKind: sh:IRI; sh:path: ekg-artpub:publishesFamily)
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:maxCount: 0; sh:path: ekg-artpub:publishesReference); sh:property: (sh:minCount: 1; sh:path: ekg-artpub:publishesFamily)), (sh:property: (sh:minCount: 1; sh:path: ekg-artpub:publishesReference); sh:property: (sh:maxCount: 0; sh:path: ekg-artpub:publishesFamily)))

6.5.2 Release Approval Shape

Label: Release Approval Shape

Identifier: `ekg-artpub:ReleaseApprovalShape`

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

IRI: `https://ekgf.org/ontology/artifact-publication#ReleaseApprovalShape`

RDF predicate	Value
rdf:type	sh:NodeShape
rdfs:isDefinedBy	ekg-artpub:
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:datatype: xsd:dateTime; sh:maxCount: 1; sh:minCount: 1; sh:path: ekg-artpub:approvedAt)
sh:property	(sh:class: ekg-artpub:StandardsRelease; sh:maxCount: 1; sh:minCount: 1; sh:nodeKind: sh:IRI; sh:path: ekg-artpub:approvesRelease)
sh:targetClass	ekg-artpub:ReleaseApproval
sh:targetSubjectsOf	ekg-artpub:approvedAt
sh:targetSubjectsOf	ekg-artpub:approvedBy
sh:targetSubjectsOf	ekg-artpub:approvesRelease

6.5.3 Standards Release Shape

Label: Standards Release Shape

Identifier: `ekg-artpub:StandardsReleaseShape`

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

IRI: `https://ekgf.org/ontology/artifact-publication#StandardsReleaseShape`

RDF predicate	Value
`rdf:type`	`sh:NodeShape`
`rdfs:isDefinedBy`	ekg-artpub:
`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-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: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:minCount`: 1; `sh:minLength`: 1; `sh:path`: ekg-artpub:releaseVersion)
`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:class`: ekg-artgov:Family; `sh:maxCount`: 1; `sh:nodeKind`: `sh:IRI`; `sh:path`: ekg-artpub:releaseFamily)
`sh:sparql`	(`sh:message`: A release linked to a publication run must record the same input commit.; `sh:select`: PREFIX artpub: <https://ekgf.org/ontology/artifact-publication#> SELECT $this WHERE { $this artpub:publicationRun/ artpub:inputCommitCid ?commit . FILTER NOT EXISTS { $this artpub:inputCommitCid ? commit } })
`sh:sparql`	(`sh:message`: A release and every linked publication profile must select the same family or the same artifact references.; `sh:select`: PREFIX artpub: <https://ekgf.org/ontology/artifact-publication#> SELECT $this WHERE { $this (artpub:publicationProfile|artpub:publicationRun/ artpub:publicationProfile) ?profile . { { $this artpub:releaseFamily ?scope . FILTER NOT EXISTS { ?profile artpub:publishesFamily ? scope } } UNION { ?profile

RDF predicate	Value
	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 } } } })
`sh:targetClass`	ekg-artpub:StandardsRelease
`sh:targetObjectsOf`	ekg-artpub:approvesRelease
`sh:targetObjectsOf`	ekg-artpub:publicationRelease
`sh:targetSubjectsOf`	ekg-artpub:releaseFamily
`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:minCount`: 1; `sh:path`: ekg-artpub:releaseReference); `sh:property`: (`sh:maxCount`: 0; `sh:path`: ekg-artpub:releaseFamily)))

7 Authorization Ontology

7.1 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.
```

7.2 Classes

7.2.1 Authorization Group

Label: Authorization Group

Identifier: `ekg-authorization:AuthorizationGroup`

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.

IRI: `https://ekgf.org/ontology/authorization#AuthorizationGroup`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-authorization:

7.2.2 Authorization Rule

Label: Authorization Rule

Identifier: `ekg-authorization:AuthorizationRule`
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.
IRI: `https://ekgf.org/ontology/authorization#AuthorizationRule`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-authorization:

7.3 Properties

7.3.1 Has Group

Label: Has Group
Identifier: `ekg-authorization:hasGroup`
Associates an authorization rule with an authorization group participating in that rule.
IRI: `https://ekgf.org/ontology/authorization#hasGroup`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-authorization:AuthorizationRule
`rdfs:isDefinedBy`	ekg-authorization:
`rdfs:range`	ekg-authorization:AuthorizationGroup

7.3.2 Has Persona

Label: Has Persona
Identifier: `ekg-authorization:hasPersona`
Associates an authorization rule with a persona to which the rule relates.
IRI: `https://ekgf.org/ontology/authorization#hasPersona`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-authorization:AuthorizationRule
`rdfs:isDefinedBy`	ekg-authorization:
`rdfs:range`	ekg-persona:Persona

8 Capability Ontology

8.1 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.

8.2 Classes

8.2.1 Capability

Label: Capability

Identifier: `ekg-capability:Capability`

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

IRI: `https://ekgf.org/ontology/capability#Capability`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-capability:

8.2.2 Capability Domain

Label: Capability Domain

Identifier: `ekg-capability:CapabilityDomain`

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

IRI: `https://ekgf.org/ontology/capability#CapabilityDomain`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-capability:

8.2.3 Sub-capability

Label: Sub-capability

Identifier: `ekg-capability:SubCapability`

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.

IRI: `https://ekgf.org/ontology/capability#SubCapability`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-capability:

8.3 Named Individuals

8.3.1 Data sharing and marketplace

Label: Data sharing and marketplace

Identifier: `ekg-capability:Data_Sharing_and_Marketplace`

Instance of: ekg-capability:Capability

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

IRI: `https://ekgf.org/ontology/capability#Data_Sharing_and_Marketplace`

RDF predicate	Value
`http://purl.org/dc/elements/1.1/creator`	Jacobus Geluk
`http://purl.org/dc/elements/1.1/date`	2023-08-28T12:55:43Z
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-capability:Capability
`rdfs:isDefinedBy`	ekg-capability:

8.3.2 Designing and managing data products

Label: Designing and managing data products

Identifier: `ekg-capability:Designing_and_Managing_Data_Products`
Instance of: ekg-capability:SubCapability
The example sub-capability covering the design and management of data products.
IRI: `https://ekgf.org/ontology/capability#Designing_and_Managing_Data_Products`

RDF predicate	Value
`http://purl.org/dc/elements/1.1/creator`	Jacobus Geluk
`http://purl.org/dc/elements/1.1/date`	2023-08-28T12:56:51Z
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-capability:SubCapability
`rdfs:isDefinedBy`	ekg-capability:

9 Communication Ontology

9.1 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.

9.2 Classes

9.2.1 Audience

Label: Audience
Identifier: `ekg-communication:Audience`
A targeting collection that may cluster recipients, endpoints, tags, lists, channel eligibility, and exclusion rules.
IRI: `https://ekgf.org/ontology/communication#Audience`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:subClassOf`	ekg-communication:RecipientCollection

9.2.2 Audience membership

Label: Audience membership
Identifier: `ekg-communication:AudienceMembership`
A membership in an audience.
IRI: `https://ekgf.org/ontology/communication#AudienceMembership`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:subClassOf`	ekg-communication:Membership

9.2.3 Campaign audience snapshot

Label: Campaign audience snapshot

Identifier: `ekg-communication:CampaignAudienceSnapshot`

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

IRI: `https://ekgf.org/ontology/communication#CampaignAudienceSnapshot`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:subClassOf`	ekg-communication:RecipientCollection

9.2.4 Campaign recipient

Label: Campaign recipient

Identifier: `ekg-communication:CampaignRecipient`

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

IRI: `https://ekgf.org/ontology/communication#CampaignRecipient`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:subClassOf`	ekg-communication:Membership

9.2.5 Communication attempt

Label: Communication attempt

Identifier: `ekg-communication:CommunicationAttempt`

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

IRI: `https://ekgf.org/ontology/communication#CommunicationAttempt`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-communication:

9.2.6 Communication channel

Label: Communication channel

Identifier: `ekg-communication:CommunicationChannel`

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

IRI: `https://ekgf.org/ontology/communication#CommunicationChannel`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-communication:

9.2.7 Communication endpoint

Label: Communication endpoint

Identifier: `ekg-communication:CommunicationEndpoint`

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

IRI: `https://ekgf.org/ontology/communication#CommunicationEndpoint`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-communication:

9.2.8 Communication program

Label: Communication program

Identifier: `ekg-communication:CommunicationProgram`

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

IRI: `https://ekgf.org/ontology/communication#CommunicationProgram`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-communication:

9.2.9 Communication purpose

Label: Communication purpose

Identifier: `ekg-communication:CommunicationPurpose`

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

IRI: `https://ekgf.org/ontology/communication#CommunicationPurpose`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-communication:

9.2.10 Delivery event

Label: Delivery event

Identifier: `ekg-communication:DeliveryEvent`

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

IRI: `https://ekgf.org/ontology/communication#DeliveryEvent`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-communication:

9.2.11 Delivery status

Label: Delivery status

Identifier: `ekg-communication:DeliveryStatus`

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

IRI: `https://ekgf.org/ontology/communication#DeliveryStatus`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-communication:

9.2.12 Email endpoint

Label: Email endpoint

Identifier: `ekg-communication:EmailEndpoint`

An email address endpoint.

IRI: `https://ekgf.org/ontology/communication#EmailEndpoint`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:subClassOf`	ekg-communication:CommunicationEndpoint

9.2.13 Email recipient

Label: Email recipient

Identifier: `ekg-communication:EmailRecipient`

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

IRI: `https://ekgf.org/ontology/communication#EmailRecipient`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:subClassOf`	ekg-communication:Recipient

9.2.14 Group recipient

Label: Group recipient

Identifier: `ekg-communication:GroupRecipient`

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

IRI: `https://ekgf.org/ontology/communication#GroupRecipient`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:subClassOf`	ekg-communication:Recipient

9.2.15 Human recipient

Label: Human recipient

Identifier: `ekg-communication:HumanRecipient`

A recipient representing a natural person in a communication context.

IRI: `https://ekgf.org/ontology/communication#HumanRecipient`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:subClassOf`	ekg-communication:Recipient

9.2.16 In-app endpoint

Label: In-app endpoint

Identifier: `ekg-communication:InAppEndpoint`

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

IRI: `https://ekgf.org/ontology/communication#InAppEndpoint`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-communication:

RDF predicate	Value
rdfs:subClassOf	ekg-communication:CommunicationEndpoint

9.2.17 In-app recipient

Label: In-app recipient

Identifier: `ekg-communication:InAppRecipient`

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

IRI: `https://ekgf.org/ontology/communication#InAppRecipient`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-communication:
rdfs:subClassOf	ekg-communication:Recipient

9.2.18 Legal basis

Label: Legal basis

Identifier: `ekg-communication:LegalBasis`

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.

IRI: `https://ekgf.org/ontology/communication#LegalBasis`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-communication:

9.2.19 List membership

Label: List membership

Identifier: `ekg-communication:ListMembership`

A membership in a recipient list or communication scope.

IRI: `https://ekgf.org/ontology/communication#ListMembership`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-communication:
rdfs:subClassOf	ekg-communication:Membership

9.2.20 Mailbox recipient

Label: Mailbox recipient

Identifier: `ekg-communication:MailboxRecipient`

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

IRI: `https://ekgf.org/ontology/communication#MailboxRecipient`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-communication:
rdfs:subClassOf	ekg-communication:Recipient

9.2.21 Marketing program

Label: Marketing program

Identifier: `ekg-communication:MarketingProgram`

A communication program whose purpose is marketing.

IRI: `https://ekgf.org/ontology/communication#MarketingProgram`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:subClassOf`	ekg-communication:CommunicationProgram

9.2.22 Membership

Label: Membership

Identifier: `ekg-communication:Membership`

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

IRI: `https://ekgf.org/ontology/communication#Membership`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-communication:

9.2.23 Membership derivation evidence

Label: Membership derivation evidence

Identifier: `ekg-communication:MembershipDerivationEvidence`

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

IRI: `https://ekgf.org/ontology/communication#MembershipDerivationEvidence`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:subClassOf`	ekg-communication:MembershipEvidence

9.2.24 Membership event

Label: Membership event

Identifier: `ekg-communication:MembershipEvent`

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

IRI: `https://ekgf.org/ontology/communication#MembershipEvent`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-communication:

9.2.25 Membership evidence

Label: Membership evidence

Identifier: `ekg-communication:MembershipEvidence`

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

IRI: `https://ekgf.org/ontology/communication#MembershipEvidence`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-communication:

9.2.26 Membership status

Label: Membership status

Identifier: `ekg-communication:MembershipStatus`

Controlled current status of a membership.

IRI: `https://ekgf.org/ontology/communication#MembershipStatus`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-communication:

9.2.27 Organization recipient

Label: Organization recipient

Identifier: `ekg-communication:OrganizationRecipient`

A recipient representing an organization or other legal entity.

IRI: `https://ekgf.org/ontology/communication#OrganizationRecipient`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-communication:
rdfs:subClassOf	ekg-communication:Recipient

9.2.28 Postal endpoint

Label: Postal endpoint

Identifier: `ekg-communication:PostalEndpoint`

A postal address endpoint.

IRI: `https://ekgf.org/ontology/communication#PostalEndpoint`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-communication:
rdfs:subClassOf	ekg-communication:CommunicationEndpoint

9.2.29 Postal recipient

Label: Postal recipient

Identifier: `ekg-communication:PostalRecipient`

A recipient addressable by postal mail or courier delivery.

IRI: `https://ekgf.org/ontology/communication#PostalRecipient`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-communication:
rdfs:subClassOf	ekg-communication:Recipient

9.2.30 Push endpoint

Label: Push endpoint

Identifier: `ekg-communication:PushEndpoint`

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

IRI: `https://ekgf.org/ontology/communication#PushEndpoint`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:subClassOf`	ekg-communication:CommunicationEndpoint

9.2.31 Push recipient

Label: Push recipient

Identifier: `ekg-communication:PushRecipient`

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

IRI: `https://ekgf.org/ontology/communication#PushRecipient`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:subClassOf`	ekg-communication:Recipient

9.2.32 Recipient

Label: Recipient

Identifier: `ekg-communication:Recipient`

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

IRI: `https://ekgf.org/ontology/communication#Recipient`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-communication:

9.2.33 Recipient collection

Label: Recipient collection

Identifier: `ekg-communication:RecipientCollection`

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

IRI: `https://ekgf.org/ontology/communication#RecipientCollection`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-communication:

9.2.34 Recipient consent

Label: Recipient consent

Identifier: `ekg-communication:RecipientConsent`

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

IRI: `https://ekgf.org/ontology/communication#RecipientConsent`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-communication:
rdfs:subClassOf	ekg-communication:MembershipEvent
rdfs:subClassOf	ekg-communication:MembershipEvidence

9.2.35 Recipient consent withdrawal

Label: Recipient consent withdrawal

Identifier: `ekg-communication:RecipientConsentWithdrawal`

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

IRI: `https://ekgf.org/ontology/communication#RecipientConsentWithdrawal`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-communication:
rdfs:subClassOf	ekg-communication:MembershipEvent
rdfs:subClassOf	ekg-communication:MembershipEvidence

9.2.36 Recipient legal category

Label: Recipient legal category

Identifier: `ekg-communication:RecipientLegalCategory`

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

IRI: `https://ekgf.org/ontology/communication#RecipientLegalCategory`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-communication:

9.2.37 Recipient legal classification

Label: Recipient legal classification

Identifier: `ekg-communication:RecipientLegalClassification`

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

IRI: `https://ekgf.org/ontology/communication#RecipientLegalClassification`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-communication:

9.2.38 Recipient list

Label: Recipient list

Identifier: `ekg-communication:RecipientList`

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

IRI: `https://ekgf.org/ontology/communication#RecipientList`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-communication:
rdfs:subClassOf	ekg-communication:RecipientCollection

9.2.39 Recipient suppression

Label: Recipient suppression

Identifier: `ekg-communication:RecipientSuppression`

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

IRI: `https://ekgf.org/ontology/communication#RecipientSuppression`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-communication:
rdfs:subClassOf	ekg-communication:MembershipEvent
rdfs:subClassOf	ekg-communication:MembershipEvidence

9.2.40 Recipient tag

Label: Recipient tag

Identifier: `ekg-communication:RecipientTag`

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

IRI: `https://ekgf.org/ontology/communication#RecipientTag`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-communication:
rdfs:subClassOf	ekg-communication:RecipientCollection

9.2.41 Service recipient

Label: Service recipient

Identifier: `ekg-communication:ServiceRecipient`

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

IRI: `https://ekgf.org/ontology/communication#ServiceRecipient`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-communication:
rdfs:subClassOf	ekg-communication:Recipient

9.2.42 SMS endpoint

Label: SMS endpoint

Identifier: `ekg-communication:SmsEndpoint`

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

IRI: `https://ekgf.org/ontology/communication#SmsEndpoint`

RDF predicate	Value
rdf:type	owl:Class

RDF predicate	Value
rdfs:isDefinedBy	ekg-communication:
rdfs:subClassOf	ekg-communication:CommunicationEndpoint

9.2.43 SMS recipient

Label: SMS recipient

Identifier: `ekg-communication:SmsRecipient`

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

IRI: `https://ekgf.org/ontology/communication#SmsRecipient`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-communication:
rdfs:subClassOf	ekg-communication:Recipient

9.2.44 Tag assignment

Label: Tag assignment

Identifier: `ekg-communication:TagAssignment`

A membership assigning a recipient or endpoint to a tag.

IRI: `https://ekgf.org/ontology/communication#TagAssignment`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-communication:
rdfs:subClassOf	ekg-communication:Membership

9.2.45 Voice endpoint

Label: Voice endpoint

Identifier: `ekg-communication:VoiceEndpoint`

A phone-number endpoint for voice calls.

IRI: `https://ekgf.org/ontology/communication#VoiceEndpoint`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-communication:
rdfs:subClassOf	ekg-communication:CommunicationEndpoint

9.2.46 Voice recipient

Label: Voice recipient

Identifier: `ekg-communication:VoiceRecipient`

A recipient addressable by voice call.

IRI: `https://ekgf.org/ontology/communication#VoiceRecipient`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-communication:
rdfs:subClassOf	ekg-communication:Recipient

9.2.47 Webhook endpoint

Label: Webhook endpoint

Identifier: `ekg-communication:WebhookEndpoint`

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

IRI: `https://ekgf.org/ontology/communication#WebhookEndpoint`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:subClassOf`	ekg-communication:CommunicationEndpoint

9.2.48 Webhook recipient

Label: Webhook recipient

Identifier: `ekg-communication:WebhookRecipient`

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

IRI: `https://ekgf.org/ontology/communication#WebhookRecipient`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:subClassOf`	ekg-communication:Recipient

9.3 Properties

9.3.1 allowed channel

Label: allowed channel

Identifier: `ekg-communication:allowedChannel`

A channel allowed or expected for members of a collection.

IRI: `https://ekgf.org/ontology/communication#allowedChannel`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-communication:RecipientCollection
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:range`	ekg-communication:CommunicationChannel

9.3.2 attempt endpoint

Label: attempt endpoint

Identifier: `ekg-communication:attemptEndpoint`

Concrete endpoint used by a communication attempt.

IRI: `https://ekgf.org/ontology/communication#attemptEndpoint`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-communication:CommunicationAttempt
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:range`	ekg-communication:CommunicationEndpoint

9.3.3 attempt recipient

Label: attempt recipient

Identifier: `ekg-communication:attemptRecipient`

Recipient targeted by a communication attempt.

IRI: `https://ekgf.org/ontology/communication#attemptRecipient`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-communication:CommunicationAttempt
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:range`	ekg-communication:Recipient

9.3.4 campaign audience

Label: campaign audience

Identifier: `ekg-communication:campaignAudience`

Source audience resolved into a campaign snapshot.

IRI: `https://ekgf.org/ontology/communication#campaignAudience`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-communication:CampaignAudienceSnapshot
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:range`	ekg-communication:Audience

9.3.5 captured at

Label: captured at

Identifier: `ekg-communication:capturedAt`

Timestamp when consent or lawful-basis evidence was captured.

IRI: `https://ekgf.org/ontology/communication#capturedAt`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-communication:RecipientConsent
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:range`	`xsd:dateTime`

9.3.6 collection purpose

Label: collection purpose

Identifier: `ekg-communication:collectionPurpose`

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

IRI: `https://ekgf.org/ontology/communication#collectionPurpose`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-communication:RecipientCollection
`rdfs:isDefinedBy`	ekg-communication:

RDF predicate	Value
rdfs:range	ekg-communication:CommunicationPurpose

9.3.7 communication program

Label: communication program

Identifier: `ekg-communication:communicationProgram`

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

IRI: `https://ekgf.org/ontology/communication#communicationProgram`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	rdfs:Resource
rdfs:isDefinedBy	ekg-communication:
rdfs:range	ekg-communication:CommunicationProgram

9.3.8 consent membership

Label: consent membership

Identifier: `ekg-communication:consentMembership`

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

IRI: `https://ekgf.org/ontology/communication#consentMembership`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-communication:RecipientConsent
rdfs:isDefinedBy	ekg-communication:
rdfs:range	ekg-communication:Membership

9.3.9 delivery attempt

Label: delivery attempt

Identifier: `ekg-communication:deliveryAttempt`

Communication attempt described by a delivery event.

IRI: `https://ekgf.org/ontology/communication#deliveryAttempt`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-communication:DeliveryEvent
rdfs:isDefinedBy	ekg-communication:
rdfs:range	ekg-communication:CommunicationAttempt

9.3.10 delivery status

Label: delivery status

Identifier: `ekg-communication:deliveryStatus`

Delivery outcome status for a delivery event.

IRI: `https://ekgf.org/ontology/communication#deliveryStatus`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-communication:DeliveryEvent

RDF predicate	Value
rdfs:isDefinedBy	ekg-communication:
rdfs:range	ekg-communication:DeliveryStatus

9.3.11 derived from membership

Label: derived from membership

Identifier: `ekg-communication:derivedFromMembership`

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

IRI: `https://ekgf.org/ontology/communication#derivedFromMembership`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-communication:Membership
rdfs:isDefinedBy	ekg-communication:
rdfs:range	ekg-communication:Membership

9.3.12 effective from

Label: effective from

Identifier: `ekg-communication:effectiveFrom`

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

IRI: `https://ekgf.org/ontology/communication#effectiveFrom`

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdfs:domain	rdfs:Resource
rdfs:isDefinedBy	ekg-communication:
rdfs:range	xsd:dateTime

9.3.13 effective through

Label: effective through

Identifier: `ekg-communication:effectiveThrough`

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

IRI: `https://ekgf.org/ontology/communication#effectiveThrough`

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdfs:domain	rdfs:Resource
rdfs:isDefinedBy	ekg-communication:
rdfs:range	xsd:dateTime

9.3.14 endpoint address

Label: endpoint address

Identifier: `ekg-communication:endpointAddress`

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

IRI: `https://ekgf.org/ontology/communication#endpointAddress`

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-communication:CommunicationEndpoint
rdfs:isDefinedBy	ekg-communication:
rdfs:range	xsd:string

9.3.15 endpoint channel

Label: endpoint channel

Identifier: `ekg-communication:endpointChannel`

The transport channel used by an endpoint.

IRI: `https://ekgf.org/ontology/communication#endpointChannel`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-communication:CommunicationEndpoint
rdfs:isDefinedBy	ekg-communication:
rdfs:range	ekg-communication:CommunicationChannel

9.3.16 evidence collection

Label: evidence collection

Identifier: `ekg-communication:evidenceCollection`

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

IRI: `https://ekgf.org/ontology/communication#evidenceCollection`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-communication:MembershipEvidence
rdfs:isDefinedBy	ekg-communication:
rdfs:range	ekg-communication:RecipientCollection

9.3.17 evidence source

Label: evidence source

Identifier: `ekg-communication:evidenceSource`

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

IRI: `https://ekgf.org/ontology/communication#evidenceSource`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-communication:MembershipEvidence
rdfs:isDefinedBy	ekg-communication:
rdfs:range	rdfs:Resource

9.3.18 has delivery event

Label: has delivery event

Identifier: `ekg-communication:hasDeliveryEvent`

Links an attempt to one delivery event.

IRI: `https://ekgf.org/ontology/communication#hasDeliveryEvent`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-communication:CommunicationAttempt
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:range`	ekg-communication:DeliveryEvent

9.3.19 has endpoint

Label: has endpoint

Identifier: `ekg-communication:hasEndpoint`

Links a recipient to one concrete channel endpoint.

IRI: `https://ekgf.org/ontology/communication#hasEndpoint`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-communication:Recipient
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:range`	ekg-communication:CommunicationEndpoint

9.3.20 has membership

Label: has membership

Identifier: `ekg-communication:hasMembership`

Links a collection to a reified membership row.

IRI: `https://ekgf.org/ontology/communication#hasMembership`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-communication:RecipientCollection
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:range`	ekg-communication:Membership

9.3.21 jurisdiction code

Label: jurisdiction code

Identifier: `ekg-communication:jurisdictionCode`

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

IRI: `https://ekgf.org/ontology/communication#jurisdictionCode`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-communication:RecipientLegalClassification
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:range`	`xsd:string`

9.3.22 legal basis

Label: legal basis

Identifier: `ekg-communication:legalBasis`

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

IRI: `https://ekgf.org/ontology/communication#legalBasis`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	`rdfs:Resource`
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:range`	ekg-communication:LegalBasis

9.3.23 legal category

Label: legal category

Identifier: `ekg-communication:legalCategory`

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

IRI: `https://ekgf.org/ontology/communication#legalCategory`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-communication:RecipientLegalClassification
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:range`	ekg-communication:RecipientLegalCategory

9.3.24 legal classification recipient

Label: legal classification recipient

Identifier: `ekg-communication:legalClassificationRecipient`

Recipient classified for legal, consent, or retention decisions.

IRI: `https://ekgf.org/ontology/communication#legalClassificationRecipient`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-communication:RecipientLegalClassification
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:range`	ekg-communication:Recipient

9.3.25 legal jurisdiction

Label: legal jurisdiction

Identifier: `ekg-communication:legalJurisdiction`

Jurisdiction in which a legal classification applies.

IRI: `https://ekgf.org/ontology/communication#legalJurisdiction`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-communication:RecipientLegalClassification
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:range`	`rdfs:Resource`

9.3.26 membership collection

Label: membership collection

Identifier: `ekg-communication:membershipCollection`

The collection that owns a membership.

IRI: `https://ekgf.org/ontology/communication#membershipCollection`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-communication:Membership
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:range`	ekg-communication:RecipientCollection

9.3.27 membership endpoint

Label: membership endpoint

Identifier: `ekg-communication:membershipEndpoint`

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

IRI: `https://ekgf.org/ontology/communication#membershipEndpoint`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-communication:Membership
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:range`	ekg-communication:CommunicationEndpoint

9.3.28 membership event

Label: membership event

Identifier: `ekg-communication:membershipEvent`

Links a membership to an append-only event.

IRI: `https://ekgf.org/ontology/communication#membershipEvent`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-communication:Membership
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:range`	ekg-communication:MembershipEvent

9.3.29 membership evidence

Label: membership evidence

Identifier: `ekg-communication:membershipEvidence`

Links a membership to supporting evidence.

IRI: `https://ekgf.org/ontology/communication#membershipEvidence`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-communication:Membership
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:range`	ekg-communication:MembershipEvidence

9.3.30 membership recipient

Label: membership recipient

Identifier: `ekg-communication:membershipRecipient`

The recipient that belongs to a collection.

IRI: `https://ekgf.org/ontology/communication#membershipRecipient`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-communication:Membership
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:range`	ekg-communication:Recipient

9.3.31 membership status

Label: membership status

Identifier: `ekg-communication:membershipStatus`

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

IRI: `https://ekgf.org/ontology/communication#membershipStatus`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-communication:Membership
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:range`	ekg-communication:MembershipStatus

9.3.32 occurred at

Label: occurred at

Identifier: `ekg-communication:occurredAt`

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

IRI: `https://ekgf.org/ontology/communication#occurredAt`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	`rdfs:Resource`
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:range`	`xsd:dateTime`

9.3.33 policy version

Label: policy version

Identifier: `ekg-communication:policyVersion`

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

IRI: `https://ekgf.org/ontology/communication#policyVersion`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	`rdfs:Resource`
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:range`	`xsd:string`

9.3.34 program channel

Label: program channel

Identifier: `ekg-communication:programChannel`

The transport channel used by a communication program.

IRI: `https://ekgf.org/ontology/communication#programChannel`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-communication:CommunicationProgram
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:range`	ekg-communication:CommunicationChannel

9.3.35 program purpose

Label: program purpose

Identifier: `ekg-communication:programPurpose`

The communication purpose served by a communication program.

IRI: `https://ekgf.org/ontology/communication#programPurpose`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-communication:CommunicationProgram
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:range`	ekg-communication:CommunicationPurpose

9.3.36 provenance

Label: provenance

Identifier: `ekg-communication:provenance`

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

IRI: `https://ekgf.org/ontology/communication#provenance`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	`rdfs:Resource`
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:range`	`xsd:string`

9.3.37 provider message id

Label: provider message id

Identifier: `ekg-communication:providerMessageId`

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

IRI: `https://ekgf.org/ontology/communication#providerMessageId`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	`rdfs:Resource`
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:range`	`xsd:string`

9.3.38 provider recipient id

Label: provider recipient id

Identifier: `ekg-communication:providerRecipientId`

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

IRI: `https://ekgf.org/ontology/communication#providerRecipientId`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	`rdfs:Resource`
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:range`	`xsd:string`

9.3.39 recorded at

Label: recorded at

Identifier: `ekg-communication:recordedAt`

Timestamp when a fact or evidence record was recorded.

IRI: `https://ekgf.org/ontology/communication#recordedAt`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	`rdfs:Resource`
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:range`	`xsd:dateTime`

9.3.40 represented resource

Label: represented resource

Identifier: `ekg-communication:representedResource`

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

IRI: `https://ekgf.org/ontology/communication#representedResource`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-communication:Recipient
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:range`	`rdfs:Resource`

9.3.41 retention period

Label: retention period

Identifier: `ekg-communication:retentionPeriod`

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

IRI: `https://ekgf.org/ontology/communication#retentionPeriod`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	`rdfs:Resource`
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:range`	`xsd:duration`

9.3.42 retention policy

Label: retention policy

Identifier: `ekg-communication:retentionPolicy`
Retention policy resource applicable to a classification, membership, evidence record, attempt, or delivery event.
IRI: `https://ekgf.org/ontology/communication#retentionPolicy`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	`rdfs:Resource`
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:range`	`rdfs:Resource`

9.3.43 suppression membership
Label: suppression membership
Identifier: `ekg-communication:suppressionMembership`
Membership scope suppressed by a suppression event.
IRI: `https://ekgf.org/ontology/communication#suppressionMembership`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-communication:RecipientSuppression
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:range`	ekg-communication:Membership

9.3.44 withdrawn consent
Label: withdrawn consent
Identifier: `ekg-communication:withdrawnConsent`
Consent record withdrawn by a later withdrawal event.
IRI: `https://ekgf.org/ontology/communication#withdrawnConsent`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-communication:RecipientConsentWithdrawal
`rdfs:isDefinedBy`	ekg-communication:
`rdfs:range`	ekg-communication:RecipientConsent

9.4 Named Individuals
9.4.1 Active membership
Label: Active membership
Identifier: `ekg-communication:ActiveMembership`
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.
IRI: `https://ekgf.org/ontology/communication#ActiveMembership`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-communication:MembershipStatus

RDF predicate	Value
rdfs:isDefinedBy	ekg-communication:

9.4.2 Bounced

Label: Bounced

Identifier: `ekg-communication:Bounced`

Instance of: ekg-communication:DeliveryStatus

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

IRI: `https://ekgf.org/ontology/communication#Bounced`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:DeliveryStatus
rdfs:isDefinedBy	ekg-communication:

9.4.3 Complained

Label: Complained

Identifier: `ekg-communication:Complained`

Instance of: ekg-communication:DeliveryStatus

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

IRI: `https://ekgf.org/ontology/communication#Complained`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:DeliveryStatus
rdfs:isDefinedBy	ekg-communication:

9.4.4 Consent legal basis

Label: Consent legal basis

Identifier: `ekg-communication:ConsentLegalBasis`

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.

IRI: `https://ekgf.org/ontology/communication#ConsentLegalBasis`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:LegalBasis
rdfs:isDefinedBy	ekg-communication:

9.4.5 Consent withdrawn membership

Label: Consent withdrawn membership

Identifier: `ekg-communication:ConsentWithdrawnMembership`

Instance of: ekg-communication:MembershipStatus

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

IRI: `https://ekgf.org/ontology/communication#ConsentWithdrawnMembership`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:MembershipStatus
rdfs:isDefinedBy	ekg-communication:

9.4.6 Contract legal basis

Label: Contract legal basis

Identifier: `ekg-communication:ContractLegalBasis`

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.

IRI: `https://ekgf.org/ontology/communication#ContractLegalBasis`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:LegalBasis
rdfs:isDefinedBy	ekg-communication:

9.4.7 Delivered

Label: Delivered

Identifier: `ekg-communication:Delivered`

Instance of: ekg-communication:DeliveryStatus

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

IRI: `https://ekgf.org/ontology/communication#Delivered`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:DeliveryStatus
rdfs:isDefinedBy	ekg-communication:

9.4.8 Email channel

Label: Email channel

Identifier: `ekg-communication:EmailChannel`

Instance of: ekg-communication:CommunicationChannel

The communication-channel value identifying email transport.

IRI: `https://ekgf.org/ontology/communication#EmailChannel`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:CommunicationChannel
rdfs:isDefinedBy	ekg-communication:

9.4.9 Employee category

Label: Employee category

Identifier: `ekg-communication:EmployeeCategory`

Instance of: ekg-communication:RecipientLegalCategory

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

IRI: `https://ekgf.org/ontology/communication#EmployeeCategory`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:RecipientLegalCategory
rdfs:isDefinedBy	ekg-communication:

9.4.10 Employee operational legal basis

Label: Employee operational legal basis

Identifier: `ekg-communication:EmployeeOperationalLegalBasis`

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.

IRI: `https://ekgf.org/ontology/communication#EmployeeOperationalLegalBasis`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:LegalBasis
rdfs:isDefinedBy	ekg-communication:

9.4.11 Failed

Label: Failed

Identifier: `ekg-communication:Failed`

Instance of: ekg-communication:DeliveryStatus

The delivery-status value recording that an attempt failed.

IRI: `https://ekgf.org/ontology/communication#Failed`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:DeliveryStatus
rdfs:isDefinedBy	ekg-communication:

9.4.12 Group mailbox category

Label: Group mailbox category

Identifier: `ekg-communication:GroupMailboxCategory`

Instance of: ekg-communication:RecipientLegalCategory

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

IRI: `https://ekgf.org/ontology/communication#GroupMailboxCategory`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:RecipientLegalCategory
rdfs:isDefinedBy	ekg-communication:

9.4.13 In-app notification channel

Label: In-app notification channel

Identifier: `ekg-communication:InAppNotificationChannel`

Instance of: ekg-communication:CommunicationChannel

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

IRI: `https://ekgf.org/ontology/communication#InAppNotificationChannel`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-communication:CommunicationChannel
`rdfs:isDefinedBy`	ekg-communication:

9.4.14 Inactive membership

Label: Inactive membership

Identifier: `ekg-communication:InactiveMembership`

Instance of: ekg-communication:MembershipStatus

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

IRI: `https://ekgf.org/ontology/communication#InactiveMembership`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-communication:MembershipStatus
`rdfs:isDefinedBy`	ekg-communication:

9.4.15 Legal notice purpose

Label: Legal notice purpose

Identifier: `ekg-communication:LegalNoticePurpose`

Instance of: ekg-communication:CommunicationPurpose

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

IRI: `https://ekgf.org/ontology/communication#LegalNoticePurpose`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-communication:CommunicationPurpose
`rdfs:isDefinedBy`	ekg-communication:

9.4.16 Legal obligation legal basis

Label: Legal obligation legal basis

Identifier: `ekg-communication:LegalObligationLegalBasis`

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.

IRI: `https://ekgf.org/ontology/communication#LegalObligationLegalBasis`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-communication:LegalBasis
`rdfs:isDefinedBy`	ekg-communication:

9.4.17 Legitimate interest legal basis

Label: Legitimate interest legal basis

Identifier: `ekg-communication:LegitimateInterestLegalBasis`

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.

IRI: `https://ekgf.org/ontology/communication#LegitimateInterestLegalBasis`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-communication:LegalBasis
`rdfs:isDefinedBy`	ekg-communication:

9.4.18 Marketing purpose

Label: Marketing purpose

Identifier: `ekg-communication:MarketingPurpose`

Instance of: ekg-communication:CommunicationPurpose

The communication-purpose value identifying marketing communications.

IRI: `https://ekgf.org/ontology/communication#MarketingPurpose`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-communication:CommunicationPurpose
`rdfs:isDefinedBy`	ekg-communication:

9.4.19 Natural person category

Label: Natural person category

Identifier: `ekg-communication:NaturalPersonCategory`

Instance of: ekg-communication:RecipientLegalCategory

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

IRI: `https://ekgf.org/ontology/communication#NaturalPersonCategory`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-communication:RecipientLegalCategory
`rdfs:isDefinedBy`	ekg-communication:

9.4.20 Operational purpose

Label: Operational purpose

Identifier: `ekg-communication:OperationalPurpose`

Instance of: ekg-communication:CommunicationPurpose

The communication-purpose value identifying operational communications.

IRI: `https://ekgf.org/ontology/communication#OperationalPurpose`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-communication:CommunicationPurpose
`rdfs:isDefinedBy`	ekg-communication:

9.4.21 Organization category

Label: Organization category

Identifier: `ekg-communication:OrganizationCategory`

Instance of: ekg-communication:RecipientLegalCategory

The recipient legal-category value identifying an organization in a recorded jurisdictional classification.
IRI: `https://ekgf.org/ontology/communication#OrganizationCategory`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-communication:RecipientLegalCategory
`rdfs:isDefinedBy`	ekg-communication:

9.4.22 Pending
Label: Pending
Identifier: `ekg-communication:Pending`
Instance of: ekg-communication:DeliveryStatus
The delivery-status value recording an attempt awaiting further processing or an outcome.
IRI: `https://ekgf.org/ontology/communication#Pending`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-communication:DeliveryStatus
`rdfs:isDefinedBy`	ekg-communication:

9.4.23 Postal mail channel
Label: Postal mail channel
Identifier: `ekg-communication:PostalMailChannel`
Instance of: ekg-communication:CommunicationChannel
The communication-channel value identifying postal-mail transport.
IRI: `https://ekgf.org/ontology/communication#PostalMailChannel`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-communication:CommunicationChannel
`rdfs:isDefinedBy`	ekg-communication:

9.4.24 Public authority category
Label: Public authority category
Identifier: `ekg-communication:PublicAuthorityCategory`
Instance of: ekg-communication:RecipientLegalCategory
The recipient legal-category value identifying a public authority in a recorded jurisdictional classification.
IRI: `https://ekgf.org/ontology/communication#PublicAuthorityCategory`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-communication:RecipientLegalCategory
`rdfs:isDefinedBy`	ekg-communication:

9.4.25 Push notification channel
Label: Push notification channel
Identifier: `ekg-communication:PushNotificationChannel`
Instance of: ekg-communication:CommunicationChannel
The communication-channel value identifying push-notification transport.

IRI: `https://ekgf.org/ontology/communication#PushNotificationChannel`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-communication:CommunicationChannel
`rdfs:isDefinedBy`	ekg-communication:

9.4.26 Security purpose

Label: Security purpose

Identifier: `ekg-communication:SecurityPurpose`

Instance of: ekg-communication:CommunicationPurpose

The communication-purpose value identifying security communications.

IRI: `https://ekgf.org/ontology/communication#SecurityPurpose`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-communication:CommunicationPurpose
`rdfs:isDefinedBy`	ekg-communication:

9.4.27 Sent

Label: Sent

Identifier: `ekg-communication:Sent`

Instance of: ekg-communication:DeliveryStatus

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

IRI: `https://ekgf.org/ontology/communication#Sent`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-communication:DeliveryStatus
`rdfs:isDefinedBy`	ekg-communication:

9.4.28 Skipped

Label: Skipped

Identifier: `ekg-communication:Skipped`

Instance of: ekg-communication:DeliveryStatus

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

IRI: `https://ekgf.org/ontology/communication#Skipped`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-communication:DeliveryStatus
`rdfs:isDefinedBy`	ekg-communication:

9.4.29 SMS channel

Label: SMS channel

Identifier: `ekg-communication:SmsChannel`

Instance of: ekg-communication:CommunicationChannel

The communication-channel value identifying SMS transport.

IRI: `https://ekgf.org/ontology/communication#SmsChannel`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:CommunicationChannel
rdfs:isDefinedBy	ekg-communication:

9.4.30 Support purpose

Label: Support purpose

Identifier: `ekg-communication:SupportPurpose`

Instance of: ekg-communication:CommunicationPurpose

The communication-purpose value identifying support communications.

IRI: `https://ekgf.org/ontology/communication#SupportPurpose`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:CommunicationPurpose
rdfs:isDefinedBy	ekg-communication:

9.4.31 Suppressed

Label: Suppressed

Identifier: `ekg-communication:Suppressed`

Instance of: ekg-communication:DeliveryStatus

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

IRI: `https://ekgf.org/ontology/communication#Suppressed`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:DeliveryStatus
rdfs:isDefinedBy	ekg-communication:

9.4.32 Suppressed membership

Label: Suppressed membership

Identifier: `ekg-communication:SuppressedMembership`

Instance of: ekg-communication:MembershipStatus

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

IRI: `https://ekgf.org/ontology/communication#SuppressedMembership`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:MembershipStatus
rdfs:isDefinedBy	ekg-communication:

9.4.33 Technical service category

Label: Technical service category

Identifier: `ekg-communication:TechnicalServiceCategory`

Instance of: ekg-communication:RecipientLegalCategory

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

IRI: `https://ekgf.org/ontology/communication#TechnicalServiceCategory`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:RecipientLegalCategory
rdfs:isDefinedBy	ekg-communication:

9.4.34 Technical service legal basis

Label: Technical service legal basis

Identifier: `ekg-communication:TechnicalServiceLegalBasis`

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.

IRI: `https://ekgf.org/ontology/communication#TechnicalServiceLegalBasis`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:LegalBasis
rdfs:isDefinedBy	ekg-communication:

9.4.35 Transactional purpose

Label: Transactional purpose

Identifier: `ekg-communication:TransactionalPurpose`

Instance of: ekg-communication:CommunicationPurpose

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

IRI: `https://ekgf.org/ontology/communication#TransactionalPurpose`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:CommunicationPurpose
rdfs:isDefinedBy	ekg-communication:

9.4.36 Unsubscribed

Label: Unsubscribed

Identifier: `ekg-communication:Unsubscribed`

Instance of: ekg-communication:DeliveryStatus

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

IRI: `https://ekgf.org/ontology/communication#Unsubscribed`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-communication:DeliveryStatus
rdfs:isDefinedBy	ekg-communication:

9.4.37 Communication Validation Shapes

Label: Communication Validation Shapes

Identifier: `ekg-communication:ValidationShapeSet`

Instance of: ekg-shape:ShapeSet

Validates recipients, endpoints, programs, collections, memberships, legal classifications, consent evidence, attempts, and delivery events.

IRI: `https://ekgf.org/ontology/communication#ValidationShapeSet`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-shape:ShapeSet
`rdfs:isDefinedBy`	ekg-communication:
`ekg-shape:hasShape`	ekg-communication:CommunicationAttemptShape
`ekg-shape:hasShape`	ekg-communication:CommunicationEndpointShape
`ekg-shape:hasShape`	ekg-communication:CommunicationProgramShape
`ekg-shape:hasShape`	ekg-communication:DeliveryEventShape
`ekg-shape:hasShape`	ekg-communication:EmailRecipientShape
`ekg-shape:hasShape`	ekg-communication:MembershipShape
`ekg-shape:hasShape`	ekg-communication:RecipientCollectionShape
`ekg-shape:hasShape`	ekg-communication:RecipientConsentShape
`ekg-shape:hasShape`	ekg-communication:RecipientLegalClassificationShape
`ekg-shape:hasShape`	ekg-communication:RecipientShape
`ekg-shape:hasShapesGraph`	ekg-communication:
`ekg-shape:shapePurpose`	ekg-shape:Validation

9.4.38 Voice call channel

Label: Voice call channel

Identifier: `ekg-communication:VoiceCallChannel`

Instance of: ekg-communication:CommunicationChannel

The communication-channel value identifying voice-call transport.

IRI: `https://ekgf.org/ontology/communication#VoiceCallChannel`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-communication:CommunicationChannel
`rdfs:isDefinedBy`	ekg-communication:

9.4.39 Webhook channel

Label: Webhook channel

Identifier: `ekg-communication:WebhookChannel`

Instance of: ekg-communication:CommunicationChannel

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

IRI: `https://ekgf.org/ontology/communication#WebhookChannel`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-communication:CommunicationChannel
`rdfs:isDefinedBy`	ekg-communication:

9.5 Shapes

9.5.1 Communication Attempt Shape

Label: Communication Attempt Shape

Identifier: `ekg-communication:CommunicationAttemptShape`

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

IRI: `https://ekgf.org/ontology/communication#CommunicationAttemptShape`

RDF predicate	Value
`rdf:type`	`sh:NodeShape`
`rdfs:isDefinedBy`	ekg-communication:
`sh:or`	((`sh:property`: (`sh:minCount`: 1; `sh:path`: ekg-communication:attemptRecipient)), (`sh:property`: (`sh:minCount`: 1; `sh:path`: ekg-communication:attemptEndpoint)))
`sh:property`	(`sh:class`: ekg-communication:CommunicationProgram; `sh:maxCount`: 1; `sh:minCount`: 1; `sh:name`: Communication program; `sh:path`: ekg-communication:communicationProgram)
`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:CommunicationEndpoint; `sh:maxCount`: 1; `sh:name`: Attempt endpoint; `sh:path`: ekg-communication:attemptEndpoint)
`sh:targetClass`	ekg-communication:CommunicationAttempt

9.5.2 Communication Endpoint Shape

Label: Communication Endpoint Shape

Identifier: `ekg-communication:CommunicationEndpointShape`

Validates a communication endpoint's channel and address literals.

IRI: `https://ekgf.org/ontology/communication#CommunicationEndpointShape`

RDF predicate	Value
`rdf:type`	`sh:NodeShape`
`rdfs:isDefinedBy`	ekg-communication:
`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

9.5.3 Communication Program Shape

Label: Communication Program Shape

Identifier: `ekg-communication:CommunicationProgramShape`

Validates a communication program's declared channel and purpose.

IRI: `https://ekgf.org/ontology/communication#CommunicationProgramShape`

RDF predicate	Value
`rdf:type`	`sh:NodeShape`
`rdfs:isDefinedBy`	ekg-communication:

RDF predicate	Value
sh:property	(sh:class: ekg-communication:CommunicationPurpose; sh:minCount: 1; sh:name: Program purpose; sh:path: ekg-communication:programPurpose)
sh:property	(sh:class: ekg-communication:CommunicationChannel; sh:minCount: 1; sh:name: Program channel; sh:path: ekg-communication:programChannel)
sh:targetClass	ekg-communication:CommunicationProgram

9.5.4 Delivery Event Shape

Label: Delivery Event Shape

Identifier: `ekg-communication:DeliveryEventShape`

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

IRI: `https://ekgf.org/ontology/communication#DeliveryEventShape`

RDF predicate	Value
rdf:type	sh:NodeShape
rdfs:isDefinedBy	ekg-communication:
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:datatype: xsd:dateTime; sh:maxCount: 1; sh:minCount: 1; sh:name: Occurred at; sh:path: ekg-communication:occurredAt)
sh:property	(sh:class: ekg-communication:DeliveryStatus; sh:maxCount: 1; sh:minCount: 1; sh:name: Delivery status; sh:path: ekg-communication:deliveryStatus)
sh:targetClass	ekg-communication:DeliveryEvent

9.5.5 Email Recipient Shape

Label: Email Recipient Shape

Identifier: `ekg-communication:EmailRecipientShape`

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

IRI: `https://ekgf.org/ontology/communication#EmailRecipientShape`

RDF predicate	Value
rdf:type	sh:NodeShape
rdfs:isDefinedBy	ekg-communication:
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

9.5.6 Membership Shape

Label: Membership Shape

Identifier: `ekg-communication:MembershipShape`

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.

IRI: `https://ekgf.org/ontology/communication#MembershipShape`

RDF predicate	Value
rdf:type	sh:NodeShape
rdfs:isDefinedBy	ekg-communication:
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:MembershipStatus; sh:maxCount: 1; sh:name: Membership status; sh:path: ekg-communication:membershipStatus)
sh:property	(sh:class: ekg-communication:RecipientCollection; sh:maxCount: 1; sh:minCount: 1; sh:name: Membership collection; sh:path: ekg-communication:membershipCollection)
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:targetClass	ekg-communication:Membership

9.5.7 Recipient Collection Shape

Label: Recipient Collection Shape

Identifier: `ekg-communication:RecipientCollectionShape`

Validates a recipient collection's declared channels and purpose.

IRI: `https://ekgf.org/ontology/communication#RecipientCollectionShape`

RDF predicate	Value
rdf:type	sh:NodeShape
rdfs:isDefinedBy	ekg-communication:
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

9.5.8 Recipient Consent Shape

Label: Recipient Consent Shape

Identifier: `ekg-communication:RecipientConsentShape`

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

IRI: `https://ekgf.org/ontology/communication#RecipientConsentShape`

RDF predicate	Value
rdf:type	sh:NodeShape
rdfs:isDefinedBy	ekg-communication:

RDF predicate	Value
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:datatype: xsd:dateTime; sh:maxCount: 1; sh:minCount: 1; sh:name: Captured at; sh:path: ekg-communication:capturedAt)
sh:property	(sh:class: ekg-communication:LegalBasis; sh:minCount: 1; sh:name: Legal basis; sh:path: ekg-communication:legalBasis)
sh:targetClass	ekg-communication:RecipientConsent

9.5.9 Recipient Legal Classification Shape

Label: Recipient Legal Classification Shape

Identifier: `ekg-communication:RecipientLegalClassificationShape`

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

IRI: `https://ekgf.org/ontology/communication#RecipientLegalClassificationShape`

RDF predicate	Value
rdf:type	sh:NodeShape
rdfs:isDefinedBy	ekg-communication:
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:RecipientLegalCategory; sh:maxCount: 1; sh:minCount: 1; sh:name: Legal category; sh:path: ekg-communication:legalCategory)
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

9.5.10 Recipient Shape

Label: Recipient Shape

Identifier: `ekg-communication:RecipientShape`

Validates the resource represented by a communication recipient.

IRI: `https://ekgf.org/ontology/communication#RecipientShape`

RDF predicate	Value
rdf:type	sh:NodeShape
rdfs:isDefinedBy	ekg-communication:
sh:property	(sh:maxCount: 1; sh:name: Represented resource; sh:nodeKind: sh:IRI; sh:path: ekg-communication:representedResource)
sh:targetClass	ekg-communication:Recipient

10 Concept Ontology

10.1 Overview

Concept Ontology

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

10.1.1 Shared meanings, usage, and manifestations

A data element concept identifies reusable meaning. A story's concept usage records how that story uses the meaning. A manifestation expresses the meaning in business language or a technical system. These resources serve different purposes and retain separate identities.

The Story Ontology declares the input, output, and dependent usage model and its validation shapes. The Use Case Ontology describes concept-set ownership. Together these distinguish the story-owned usage from the shared concept and its vocabulary membership.

10.2 Classes

10.2.1 abbreviation

Label: abbreviation

Identifier: `ekg-concept:Abbreviation`

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

IRI: `https://ekgf.org/ontology/concept#Abbreviation`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-concept:
`rdfs:subClassOf`	ekg-concept:BusinessTerm

10.2.2 business term

Label: business term

Identifier: `ekg-concept:BusinessTerm`

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

IRI: `https://ekgf.org/ontology/concept#BusinessTerm`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-concept:
`rdfs:subClassOf`	ekg-concept:ConceptManifestation

10.2.3 concept manifestation

Label: concept manifestation

Identifier: `ekg-concept:ConceptManifestation`

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.

IRI: `https://ekgf.org/ontology/concept#ConceptManifestation`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-concept:

10.2.4 concept set

Label: concept set

Identifier: `ekg-concept:ConceptSet`

A governed set of related data element concepts.

IRI: `https://ekgf.org/ontology/concept#ConceptSet`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-concept:

10.2.5 Cypher identifier manifestation

Label: Cypher identifier manifestation

Identifier: `ekg-concept:CypherIdentifierManifestation`

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.

IRI: `https://ekgf.org/ontology/concept#CypherIdentifierManifestation`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-concept:
`rdfs:subClassOf`	ekg-concept:TechnicalManifestation

10.2.6 data element concept

Label: data element concept

Identifier: `ekg-concept:DataElementConcept`

10.2.6.1 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.

10.2.6.2 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.

10.2.6.3 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.

`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.

10.2.6.4 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.

IRI: `https://ekgf.org/ontology/concept#DataElementConcept`

RDF predicate	Value
`dct:created`	2020-04-22T17:56:44Z
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-concept:
`rdfs:seeAlso`	`https://www.iso.org/standard/78925.html`

10.2.7 description

Label: description

Identifier: `ekg-concept:Description`

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

IRI: `https://ekgf.org/ontology/concept#Description`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-concept:
`rdfs:subClassOf`	ekg-concept:BusinessTerm

10.2.8 full plural business term

Label: full plural business term

Identifier: `ekg-concept:FullPluralBusinessTerm`

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

IRI: `https://ekgf.org/ontology/concept#FullPluralBusinessTerm`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-concept:
`rdfs:subClassOf`	ekg-concept:BusinessTerm

10.2.9 full singular business term

Label: full singular business term

Identifier: `ekg-concept:FullSingularBusinessTerm`

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

IRI: `https://ekgf.org/ontology/concept#FullSingularBusinessTerm`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-concept:
`rdfs:subClassOf`	ekg-concept:BusinessTerm

10.2.10 JSON field manifestation

Label: JSON field manifestation

Identifier: `ekg-concept:JSONFieldManifestation`

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.

IRI: `https://ekgf.org/ontology/concept#JSONFieldManifestation`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-concept:
`rdfs:subClassOf`	ekg-concept:TechnicalManifestation

10.2.11 Java identifier manifestation

Label: Java identifier manifestation

Identifier: `ekg-concept:JavaIdentifierManifestation`

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.

IRI: `https://ekgf.org/ontology/concept#JavaIdentifierManifestation`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-concept:
`rdfs:subClassOf`	ekg-concept:SourceCodeIdentifierManifestation

10.2.12 JavaScript identifier manifestation

Label: JavaScript identifier manifestation

Identifier: `ekg-concept:JavaScriptIdentifierManifestation`

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.

IRI: `https://ekgf.org/ontology/concept#JavaScriptIdentifierManifestation`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-concept:

RDF predicate	Value
rdfs:subClassOf	ekg-concept:SourceCodeIdentifierManifestation

10.2.13 literal datatype manifestation

Label: literal datatype manifestation

Identifier: `ekg-concept:LiteralDatatypeManifestation`

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.

IRI: `https://ekgf.org/ontology/concept#LiteralDatatypeManifestation`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-concept:
rdfs:subClassOf	ekg-concept:TechnicalManifestation

10.2.14 manifestation occurrence

Label: manifestation occurrence

Identifier: `ekg-concept:ManifestationOccurrence`

A concrete sighting of a concept manifestation in a system, source repository, statement, RDF file, OWL file, SHACL file, SPARQL query, SQL statement, Cypher statement, or code file.

IRI: `https://ekgf.org/ontology/concept#ManifestationOccurrence`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-concept:

10.2.15 OWL axiom manifestation

Label: OWL axiom manifestation

Identifier: `ekg-concept:OWLAxiomManifestation`

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.

IRI: `https://ekgf.org/ontology/concept#OWLAxiomManifestation`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-concept:
rdfs:subClassOf	ekg-concept:TechnicalManifestation

10.2.16 OWL class manifestation

Label: OWL class manifestation

Identifier: `ekg-concept:OWLClassManifestation`

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.

IRI: `https://ekgf.org/ontology/concept#OWLClassManifestation`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-concept:

RDF predicate	Value
rdfs:subClassOf	ekg-concept:TechnicalManifestation

10.2.17 OWL named individual manifestation

Label: OWL named individual manifestation

Identifier: `ekg-concept:OWLNamedIndividualManifestation`

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.

IRI: `https://ekgf.org/ontology/concept#OWLNamedIndividualManifestation`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-concept:
rdfs:subClassOf	ekg-concept:TechnicalManifestation

10.2.18 OWL property manifestation

Label: OWL property manifestation

Identifier: `ekg-concept:OWLPropertyManifestation`

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.

IRI: `https://ekgf.org/ontology/concept#OWLPropertyManifestation`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-concept:
rdfs:subClassOf	ekg-concept:TechnicalManifestation

10.2.19 Python identifier manifestation

Label: Python identifier manifestation

Identifier: `ekg-concept:PythonIdentifierManifestation`

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.

IRI: `https://ekgf.org/ontology/concept#PythonIdentifierManifestation`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-concept:
rdfs:subClassOf	ekg-concept:SourceCodeIdentifierManifestation

10.2.20 RDF predicate manifestation

Label: RDF predicate manifestation

Identifier: `ekg-concept:RDFPredicateManifestation`

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.

IRI: `https://ekgf.org/ontology/concept#RDFPredicateManifestation`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-concept:
rdfs:subClassOf	ekg-concept:TechnicalManifestation

10.2.21 RDFS class manifestation

Label: RDFS class manifestation

Identifier: `ekg-concept:RDFSClassManifestation`

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.

IRI: `https://ekgf.org/ontology/concept#RDFSClassManifestation`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-concept:
rdfs:subClassOf	ekg-concept:TechnicalManifestation

10.2.22 raw source field manifestation

Label: raw source field manifestation

Identifier: `ekg-concept:RawSourceFieldManifestation`

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.

IRI: `https://ekgf.org/ontology/concept#RawSourceFieldManifestation`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-concept:
rdfs:subClassOf	ekg-concept:TechnicalManifestation

10.2.23 SHACL shape manifestation

Label: SHACL shape manifestation

Identifier: `ekg-concept:SHACLShapeManifestation`

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.

IRI: `https://ekgf.org/ontology/concept#SHACLShapeManifestation`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-concept:
rdfs:subClassOf	ekg-concept:TechnicalManifestation

10.2.24 SHACL shapeset manifestation

Label: SHACL shapeset manifestation

Identifier: `ekg-concept:SHACLShapeSetManifestation`

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.

IRI: `https://ekgf.org/ontology/concept#SHACLShapeSetManifestation`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-concept:
rdfs:subClassOf	ekg-concept:TechnicalManifestation

10.2.25 SKOS concept manifestation

Label: SKOS concept manifestation

Identifier: `ekg-concept:SKOSConceptManifestation`

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.

IRI: `https://ekgf.org/ontology/concept#SKOSConceptManifestation`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-concept:
rdfs:subClassOf	ekg-concept:TechnicalManifestation

10.2.26 SKOS concept scheme manifestation

Label: SKOS concept scheme manifestation

Identifier: `ekg-concept:SKOSConceptSchemeManifestation`

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.

IRI: `https://ekgf.org/ontology/concept#SKOSConceptSchemeManifestation`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-concept:
rdfs:subClassOf	ekg-concept:TechnicalManifestation

10.2.27 SPARQL result column manifestation

Label: SPARQL result column manifestation

Identifier: `ekg-concept:SPARQLResultColumnManifestation`

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.

IRI: `https://ekgf.org/ontology/concept#SPARQLResultColumnManifestation`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-concept:
rdfs:subClassOf	ekg-concept:TechnicalManifestation

10.2.28 SPARQL variable manifestation

Label: SPARQL variable manifestation

Identifier: `ekg-concept:SPARQLVariableManifestation`

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.

IRI: `https://ekgf.org/ontology/concept#SPARQLVariableManifestation`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-concept:
`rdfs:subClassOf`	ekg-concept:TechnicalManifestation

10.2.29 SQL column manifestation

Label: SQL column manifestation

Identifier: `ekg-concept:SQLColumnManifestation`

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.

IRI: `https://ekgf.org/ontology/concept#SQLColumnManifestation`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-concept:
`rdfs:subClassOf`	ekg-concept:TechnicalManifestation

10.2.30 SQL table manifestation

Label: SQL table manifestation

Identifier: `ekg-concept:SQLTableManifestation`

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.

IRI: `https://ekgf.org/ontology/concept#SQLTableManifestation`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-concept:
`rdfs:subClassOf`	ekg-concept:TechnicalManifestation

10.2.31 source code identifier manifestation

Label: source code identifier manifestation

Identifier: `ekg-concept:SourceCodeIdentifierManifestation`

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.

IRI: `https://ekgf.org/ontology/concept#SourceCodeIdentifierManifestation`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-concept:
`rdfs:subClassOf`	ekg-concept:TechnicalManifestation

10.2.32 tagline

Label: tagline

Identifier: `ekg-concept:Tagline`

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

IRI: `https://ekgf.org/ontology/concept#Tagline`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-concept:
`rdfs:subClassOf`	ekg-concept:BusinessTerm

10.2.33 technical manifestation

Label: technical manifestation

Identifier: `ekg-concept:TechnicalManifestation`

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

IRI: `https://ekgf.org/ontology/concept#TechnicalManifestation`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-concept:
`rdfs:subClassOf`	ekg-concept:ConceptManifestation

10.2.34 Template Language

Label: Template Language

Identifier: `ekg-concept:TemplateLanguage`

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.

IRI: `https://ekgf.org/ontology/concept#TemplateLanguage`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-concept:

10.2.35 template parameter manifestation

Label: template parameter manifestation

Identifier: `ekg-concept:TemplateParameterManifestation`

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.

IRI: `https://ekgf.org/ontology/concept#TemplateParameterManifestation`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-concept:
`rdfs:subClassOf`	ekg-concept:TechnicalManifestation
`rdfs:subClassOf`	(`rdf:type`: `owl:Restriction`; `owl:minCardinality`: 1; `owl:onProperty`: ekg-concept:templateLanguage)

RDF predicate	Value
rdfs:subClassOf	(rdf:type: owl:Restriction; owl:maxCardinality: 1; owl:onProperty: ekg-concept:templateLanguage)

10.2.36 URI parameter manifestation

Label: URI parameter manifestation

Identifier: `ekg-concept:URIParameterManifestation`

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.

IRI: `https://ekgf.org/ontology/concept#URIParameterManifestation`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-concept:
rdfs:subClassOf	ekg-concept:TechnicalManifestation

10.3 Properties

10.3.1 concept

Label: concept

Identifier: `ekg-concept:concept`

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

IRI: `https://ekgf.org/ontology/concept#concept`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-concept:ConceptManifestation
rdfs:isDefinedBy	ekg-concept:
rdfs:range	ekg-concept:DataElementConcept
owl:inverseOf	ekg-concept:hasManifestation

10.3.2 has concept

Label: has concept

Identifier: `ekg-concept:hasConcept`

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

IRI: `https://ekgf.org/ontology/concept#hasConcept`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:isDefinedBy	ekg-concept:
rdfs:range	ekg-concept:DataElementConcept

10.3.3 has manifestation

Label: has manifestation

Identifier: `ekg-concept:hasManifestation`

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.

IRI: `https://ekgf.org/ontology/concept#hasManifestation`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-concept:DataElementConcept
`rdfs:isDefinedBy`	ekg-concept:
`rdfs:range`	ekg-concept:ConceptManifestation

10.3.4 in concept set

Label: in concept set

Identifier: `ekg-concept:inConceptSet`

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

IRI: `https://ekgf.org/ontology/concept#inConceptSet`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-concept:DataElementConcept
`rdfs:isDefinedBy`	ekg-concept:
`rdfs:range`	ekg-concept:ConceptSet

10.3.5 manifestation resource

Label: manifestation resource

Identifier: `ekg-concept:manifestationResource`

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.

IRI: `https://ekgf.org/ontology/concept#manifestationResource`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-concept:TechnicalManifestation
`rdfs:isDefinedBy`	ekg-concept:
`rdfs:range`	`rdfs:Resource`

10.3.6 manifestation value

Label: manifestation value

Identifier: `ekg-concept:manifestationValue`

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.

IRI: `https://ekgf.org/ontology/concept#manifestationValue`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-concept:ConceptManifestation
`rdfs:isDefinedBy`	ekg-concept:
`rdfs:range`	`xsd:string`

10.3.7 occurrence of

Label: occurrence of

Identifier: `ekg-concept:occurrenceOf`
Identifies the concept manifestation that occurs at a particular source location.
IRI: `https://ekgf.org/ontology/concept#occurrenceOf`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-concept:ManifestationOccurrence
`rdfs:isDefinedBy`	ekg-concept:
`rdfs:range`	ekg-concept:ConceptManifestation

10.3.8 preferred business term
Label: preferred business term
Identifier: `ekg-concept:preferredBusinessTerm`
Identifies a preferred business-language term for a data element concept.
IRI: `https://ekgf.org/ontology/concept#preferredBusinessTerm`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-concept:DataElementConcept
`rdfs:isDefinedBy`	ekg-concept:
`rdfs:range`	ekg-concept:BusinessTerm
`rdfs:subPropertyOf`	ekg-concept:preferredManifestation

10.3.9 preferred manifestation
Label: preferred manifestation
Identifier: `ekg-concept:preferredManifestation`
Identifies a preferred manifestation of a data element concept, specializing hasManifestation.
IRI: `https://ekgf.org/ontology/concept#preferredManifestation`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-concept:DataElementConcept
`rdfs:isDefinedBy`	ekg-concept:
`rdfs:range`	ekg-concept:ConceptManifestation
`rdfs:subPropertyOf`	ekg-concept:hasManifestation

10.3.10 preferred query parameter manifestation
Label: preferred query parameter manifestation
Identifier: `ekg-concept:preferredQueryParameterManifestation`
Identifies a preferred URI-parameter manifestation for a data element concept.
IRI: `https://ekgf.org/ontology/concept#preferredQueryParameterManifestation`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-concept:DataElementConcept
`rdfs:isDefinedBy`	ekg-concept:
`rdfs:range`	ekg-concept:URIParameterManifestation

RDF predicate	Value
rdfs:subPropertyOf	ekg-concept:preferredManifestation

10.3.11 preferred result column manifestation

Label: preferred result column manifestation

Identifier: `ekg-concept:preferredResultColumnManifestation`

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).

IRI: `https://ekgf.org/ontology/concept#preferredResultColumnManifestation`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-concept:DataElementConcept
rdfs:isDefinedBy	ekg-concept:
rdfs:range	ekg-concept:SPARQLResultColumnManifestation
rdfs:subPropertyOf	ekg-concept:preferredManifestation

10.3.12 source column

Label: source column

Identifier: `ekg-concept:sourceColumn`

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.

IRI: `https://ekgf.org/ontology/concept#sourceColumn`

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-concept:ManifestationOccurrence
rdfs:isDefinedBy	ekg-concept:
rdfs:range	xsd:integer

10.3.13 source commit

Label: source commit

Identifier: `ekg-concept:sourceCommit`

Records the source repository commit associated with a manifestation occurrence.

IRI: `https://ekgf.org/ontology/concept#sourceCommit`

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-concept:ManifestationOccurrence
rdfs:isDefinedBy	ekg-concept:
rdfs:range	xsd:string

10.3.14 source line

Label: source line

Identifier: `ekg-concept:sourceLine`

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.

IRI: `https://ekgf.org/ontology/concept#sourceLine`

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-concept:ManifestationOccurrence
rdfs:isDefinedBy	ekg-concept:
rdfs:range	xsd:integer

10.3.15 source path

Label: source path

Identifier: `ekg-concept:sourcePath`

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

IRI: `https://ekgf.org/ontology/concept#sourcePath`

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-concept:ManifestationOccurrence
rdfs:isDefinedBy	ekg-concept:
rdfs:range	xsd:string

10.3.16 source repository

Label: source repository

Identifier: `ekg-concept:sourceRepository`

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

IRI: `https://ekgf.org/ontology/concept#sourceRepository`

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-concept:ManifestationOccurrence
rdfs:isDefinedBy	ekg-concept:
rdfs:range	xsd:anyURI

10.3.17 source system

Label: source system

Identifier: `ekg-concept:sourceSystem`

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.

IRI: `https://ekgf.org/ontology/concept#sourceSystem`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-concept:ManifestationOccurrence
rdfs:isDefinedBy	ekg-concept:
rdfs:range	rdfs:Resource

10.3.18 template language

Label: template language

Identifier: `ekg-concept:templateLanguage`

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.

IRI: `https://ekgf.org/ontology/concept#templateLanguage`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:isDefinedBy`	ekg-concept:
`rdfs:range`	ekg-concept:TemplateLanguage

10.4 Named Individuals

10.4.1 Concept Validation Shapes

Label: Concept Validation Shapes

Identifier: `ekg-concept:ValidationShapeSet`

Instance of: ekg-shape:ShapeSet

Validates template-placeholder tokens and their documented template-language profiles, and rejects retired template-placeholder classifications.

IRI: `https://ekgf.org/ontology/concept#ValidationShapeSet`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-shape:ShapeSet
`rdfs:isDefinedBy`	ekg-concept:
`ekg-shape:hasShape`	ekg-concept:LegacyTemplateManifestationShape
`ekg-shape:hasShape`	ekg-concept:TemplateLanguageShape
`ekg-shape:hasShape`	ekg-concept:TemplateParameterManifestationShape
`ekg-shape:hasShapesGraph`	ekg-concept:
`ekg-shape:shapePurpose`	ekg-shape:Validation

10.5 Shapes

10.5.1 Legacy Template Manifestation Shape

Label: Legacy Template Manifestation Shape

Identifier: `ekg-concept:LegacyTemplateManifestationShape`

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.

IRI: `https://ekgf.org/ontology/concept#LegacyTemplateManifestationShape`

RDF predicate	Value
`rdf:type`	`sh:NodeShape`
`rdfs:isDefinedBy`	ekg-concept:
`sh:message`	LegraDollarManifestation is retired. Replace that type with TemplateParameterManifestation and declare its documented templateLanguage profile.
`sh:not`	(`sh:class`: `ekg-concept:LegraDollarManifestation`)
`sh:targetClass`	`ekg-concept:LegraDollarManifestation`

10.5.2 Template Language Shape

Label: Template Language Shape

Identifier: `ekg-concept:TemplateLanguageShape`

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.

IRI: `https://ekgf.org/ontology/concept#TemplateLanguageShape`

RDF predicate	Value
`rdf:type`	`sh:NodeShape`
`rdfs:isDefinedBy`	ekg-concept:
`sh:nodeKind`	`sh:IRI`
`sh:property`	(`sh:minCount`: 1; `sh:nodeKind`: `sh:IRI`; `sh:path`: `rdfs:seeAlso`)
`sh:property`	(`sh:minCount`: 1; `sh:minLength`: 1; `sh:nodeKind`: `sh:Literal`; `sh:path`: `rdfs:label`)
`sh:targetClass`	ekg-concept:TemplateLanguage
`sh:targetObjectsOf`	ekg-concept:templateLanguage

10.5.3 Template Parameter Manifestation Shape

Label: Template Parameter Manifestation Shape

Identifier: `ekg-concept:TemplateParameterManifestationShape`

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.

IRI: `https://ekgf.org/ontology/concept#TemplateParameterManifestationShape`

RDF predicate	Value
`rdf:type`	`sh:NodeShape`
`rdfs:isDefinedBy`	ekg-concept:
`sh:nodeKind`	`sh:BlankNodeOrIRI`
`sh:property`	(`sh:datatype`: `xsd:string`; `sh:maxCount`: 1; `sh:minCount`: 1; `sh:minLength`: 1; `sh:path`: ekg-concept:manifestationValue)
`sh:property`	(`sh:class`: ekg-concept:TemplateLanguage; `sh:maxCount`: 1; `sh:minCount`: 1; `sh:nodeKind`: `sh:IRI`; `sh:path`: ekg-concept:templateLanguage)
`sh:targetClass`	ekg-concept:TemplateParameterManifestation

11 DataOps Ontology

11.1 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.

11.2 Classes

11.2.1 Failed RDF Load Request

Label: Failed RDF Load Request

Identifier: `ekg-dataops:FailedLoadRequest`

A load request classified as failed because its loading operation did not complete successfully.

IRI: `https://ekgf.org/ontology/dataops#FailedLoadRequest`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-dataops:
`rdfs:subClassOf`	ekg-dataops:LoadRequest

11.2.2 Finished RDF Load Request

Label: Finished RDF Load Request

Identifier: `ekg-dataops:FinishedLoadRequest`

A load request classified as finished after its loading operation has completed.

IRI: `https://ekgf.org/ontology/dataops#FinishedLoadRequest`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-dataops:
`rdfs:subClassOf`	ekg-dataops:LoadRequest

11.2.3 RDF Load Request

Label: RDF Load Request

Identifier: `ekg-dataops:LoadRequest`

A LoadRequest represents an RDF file being loaded into the current RDF Graph Database.

IRI: `https://ekgf.org/ontology/dataops#LoadRequest`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-dataops:

11.2.4 Loading RDF Load Request

Label: Loading RDF Load Request

Identifier: `ekg-dataops:LoadingLoadRequest`

A load request whose loading operation is in progress.

IRI: `https://ekgf.org/ontology/dataops#LoadingLoadRequest`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-dataops:
`rdfs:subClassOf`	ekg-dataops:LoadRequest

11.2.5 Logical RDF Dataset

Label: Logical RDF Dataset

Identifier: `ekg-dataops:MultiGraphDataset`

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,

IRI: `https://ekgf.org/ontology/dataops#MultiGraphDataset`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-dataops:
`rdfs:subClassOf`	ekg-dataset:Dataset

11.2.6 Pipeline

Label: Pipeline

Identifier: `ekg-dataops:Pipeline`

A DataOps process in which load requests bring source data into datasets and subsequent steps can assemble or transform that data.

IRI: `https://ekgf.org/ontology/dataops#Pipeline`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-dataops:

11.2.7 Queued RDF Load Request

Label: Queued RDF Load Request

Identifier: `ekg-dataops:QueuedLoadRequest`

A load request waiting to be processed by a pipeline.

IRI: `https://ekgf.org/ontology/dataops#QueuedLoadRequest`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-dataops:
`rdfs:subClassOf`	ekg-dataops:LoadRequest

11.2.8 Single-graph RDF Dataset

Label: Single-graph RDF Dataset

Identifier: `ekg-dataops:SingleGraphDataset`

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.

IRI: `https://ekgf.org/ontology/dataops#SingleGraphDataset`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-dataops:
`rdfs:subClassOf`	ekg-dataset:Dataset

11.3 Properties

11.3.1 Created By Pipeline

Label: Created By Pipeline

Identifier: `ekg-dataops:createdByPipeline`

Identifies the DataOps pipeline that produced a dataset.

IRI: `https://ekgf.org/ontology/dataops#createdByPipeline`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-dataset:Dataset
`rdfs:isDefinedBy`	ekg-dataops:
`rdfs:range`	ekg-dataops:Pipeline

11.3.2 Load Request Target Graph

Label: Load Request Target Graph

Identifier: `ekg-dataops:graph`

Records the IRI of the target graph for a load request as an xsd:anyURI literal.

IRI: `https://ekgf.org/ontology/dataops#graph`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdf:type`	`owl:FunctionalProperty`
`rdfs:domain`	ekg-dataops:LoadRequest
`rdfs:isDefinedBy`	ekg-dataops:
`rdfs:range`	`xsd:anyURI`

11.3.3 In Pipeline

Label: In Pipeline

Identifier: `ekg-dataops:inPipeline`

Identifies the pipeline to which a load request belongs. Each load request has at most one pipeline under the declared functional-property axiom.

IRI: `https://ekgf.org/ontology/dataops#inPipeline`

RDF predicate	Value
`rdf:type`	`owl:FunctionalProperty`
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-dataops:LoadRequest
`rdfs:isDefinedBy`	ekg-dataops:
`rdfs:range`	ekg-dataops:Pipeline

11.3.4 Loaded By Load Request

Label: Loaded By Load Request

Identifier: `ekg-dataops:loadedByLoadRequest`

Relates a single-graph dataset to a load request that loaded its data.

IRI: `https://ekgf.org/ontology/dataops#loadedByLoadRequest`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-dataops:SingleGraphDataset
`rdfs:isDefinedBy`	ekg-dataops:
`rdfs:range`	ekg-dataops:LoadRequest

11.3.5 Load Request Source IRI

Label: Load Request Source IRI

Identifier: `ekg-dataops:source`

Records the IRI of the source to be loaded by a load request as an xsd:anyURI literal.

IRI: `https://ekgf.org/ontology/dataops#source`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdf:type`	`owl:FunctionalProperty`
`rdfs:domain`	ekg-dataops:LoadRequest
`rdfs:isDefinedBy`	ekg-dataops:
`rdfs:range`	`xsd:anyURI`

12 Dataset Ontology

12.1 Overview

Dataset Ontology

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

A vocabulary for RDF datasets and their source files, named graphs, identifiers, and composition into static or compound datasets.

12.2 Classes

12.2.1 Compound Dataset

Label: Compound Dataset

Identifier: `ekg-dataset:CompoundDataset`

A CompoundDataset is a Dataset that is dynamically composed of multiple other Datasets

IRI: `https://ekgf.org/ontology/dataset#CompoundDataset`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-dataset:
`rdfs:subClassOf`	ekg-dataset:Dataset

12.2.2 Dataset

Label: Dataset

Identifier: `ekg-dataset:Dataset`

An organized collection of RDF data described with source-file and graph metadata. The class specializes dcat:Dataset and can be static or composed of other datasets.

IRI: `https://ekgf.org/ontology/dataset#Dataset`

RDF predicate	Value
`http://purl.org/dc/elements/1.1/creator`	`http://orcid.org/0000-0003-2309-5068`
`http://purl.org/dc/elements/1.1/date`	2020-04-24T12:34:37Z

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-dataset:
rdfs:subClassOf	http://www.w3.org/ns/dcat#Dataset

12.2.3 Named Graph

Label: Named Graph

Identifier: ekg-dataset:NamedGraph

An identified graph used to organize RDF data in a dataset. The class represents the graph as a resource, independently of the triples it contains.

IRI: https://ekgf.org/ontology/dataset#NamedGraph

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-dataset:

12.2.4 Static Dataset

Label: Static Dataset

Identifier: ekg-dataset:StaticDataset

A Static EKG Dataset is a git-based RDF Dataset that is not produced by a DataOps pipeline

IRI: https://ekgf.org/ontology/dataset#StaticDataset

RDF predicate	Value
http://purl.org/dc/elements/1.1/creator	http://orcid.org/0000-0003-2309-5068
http://purl.org/dc/elements/1.1/date	2020-04-24T12:35:33Z
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-dataset:
rdfs:subClassOf	ekg-dataset:Dataset

12.3 Properties

12.3.1 Data Source Code

Label: Data Source Code

Identifier: ekg-dataset:dataSourceCode

The dataSourceCode is a lowercase short unique name for the given data source that is being used as the default (local) name for the named graph that holds the data source for logging purposes to identify the data source.

IRI: https://ekgf.org/ontology/dataset#dataSourceCode

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:isDefinedBy	ekg-dataset:
rdfs:range	xsd:string

12.3.2 Dataset Code

Label: Dataset Code

Identifier: ekg-dataset:datasetCode

The datasetCode is a lowercase short unique name for the given dataset that is being used as the default (local) name for the named graph that holds the Dataset or the Pipeline that creates the dataset and for logging purposes to identify the Dataset.

IRI: `https://ekgf.org/ontology/dataset#datasetCode`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdf:type`	`owl:FunctionalProperty`
`rdfs:isDefinedBy`	ekg-dataset:
`rdfs:range`	`xsd:string`

12.3.3 default graph

Label: default graph

Identifier: `ekg-dataset:defaultGraph`

The named graph that the files of the dataset are loaded into.

IRI: `https://ekgf.org/ontology/dataset#defaultGraph`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-dataset:Dataset
`rdfs:isDefinedBy`	ekg-dataset:

12.3.4 include

Label: include

Identifier: `ekg-dataset:include`

Path of an RDF file that belongs to the dataset.

IRI: `https://ekgf.org/ontology/dataset#include`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-dataset:Dataset
`rdfs:isDefinedBy`	ekg-dataset:
`rdfs:range`	`xsd:string`

12.3.5 Includes Dataset

Label: Includes Dataset

Identifier: `ekg-dataset:includesDataset`

Relates a compound dataset to one of the datasets from which it is composed.

IRI: `https://ekgf.org/ontology/dataset#includesDataset`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-dataset:CompoundDataset
`rdfs:isDefinedBy`	ekg-dataset:
`rdfs:range`	ekg-dataset:Dataset

13 File System Ontology

13.1 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.

13.2 Classes

13.2.1 archive entry kind

Label: archive entry kind

Identifier: `ekg-file-system:ArchiveEntryKind`

The closed set of member entry kinds a decomposed container records.

IRI: `https://ekgf.org/ontology/file-system#ArchiveEntryKind`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-file-system:

13.2.2 archive format

Label: archive format

Identifier: `ekg-file-system:ArchiveFormat`

The closed set of container formats the archive adapter decomposes.

IRI: `https://ekgf.org/ontology/file-system#ArchiveFormat`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-file-system:

13.2.3 archive member

Label: archive member

Identifier: `ekg-file-system:ArchiveMember`

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.

IRI: `https://ekgf.org/ontology/file-system#ArchiveMember`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-file-system:

13.2.4 archive version

Label: archive version

Identifier: `ekg-file-system:ArchiveVersion`

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.

IRI: `https://ekgf.org/ontology/file-system#ArchiveVersion`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-file-system:

13.2.5 attachment

Label: attachment

Identifier: `ekg-file-system: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.

IRI: `https://ekgf.org/ontology/file-system#Attachment`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-file-system:

13.2.6 attachment mode

Label: attachment mode

Identifier: `ekg-file-system:AttachmentMode`

How an attachment resolves its target. The mode is mandatory and is one of the five declared individuals.

IRI: `https://ekgf.org/ontology/file-system#AttachmentMode`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-file-system:

13.2.7 attachment role

Label: attachment role

Identifier: `ekg-file-system:AttachmentRole`

The business role a file plays on the resource it is attached to (proposal, contract, invoice, evidence). Workspaces declare their own role individuals.

IRI: `https://ekgf.org/ontology/file-system#AttachmentRole`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-file-system:

13.2.8 capability profile

Label: capability profile

Identifier: `ekg-file-system:CapabilityProfile`

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.

IRI: `https://ekgf.org/ontology/file-system#CapabilityProfile`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-file-system:

13.2.9 directory

Label: directory

Identifier: `ekg-file-system:Directory`

A container resource. A directory may also attach to a business object; membership is asserted by directory entries, never by direct containment edges.

IRI: `https://ekgf.org/ontology/file-system#Directory`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-file-system:

13.2.10 directory entry

Label: directory entry

Identifier: `ekg-file-system:DirectoryEntry`

One named parent/target assertion in a rooted view. The same target may appear through many entries and names without copying content.

IRI: `https://ekgf.org/ontology/file-system#DirectoryEntry`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-file-system:

13.2.11 external object

Label: external object

Identifier: `ekg-file-system:ExternalObject`

A provider-stable file object identity (not its bytes). Connector workers map provider revisions of an external object onto new file versions.

IRI: `https://ekgf.org/ontology/file-system#ExternalObject`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-file-system:

13.2.12 file series

Label: file series

Identifier: `ekg-file-system:FileSeries`

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.

IRI: `https://ekgf.org/ontology/file-system#FileSeries`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-file-system:

13.2.13 file system view

Label: file system view

Identifier: `ekg-file-system:FileSystemView`

A branch/commit-scoped rooted projection consumed by Drive or another filesystem client. Paths are presentation addresses within one view and never resource identity.

IRI: `https://ekgf.org/ontology/file-system#FileSystemView`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-file-system:

13.2.14 file version

Label: file version

Identifier: `ekg-file-system:FileVersion`

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.

IRI: `https://ekgf.org/ontology/file-system#FileVersion`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-file-system:

13.2.15 name family rule

Label: name family rule

Identifier: `ekg-file-system:NameFamilyRule`

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.

IRI: `https://ekgf.org/ontology/file-system#NameFamilyRule`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-file-system:

13.2.16 path rule

Label: path rule

Identifier: `ekg-file-system:PathRule`

A path-tracking target: the file series currently occupying a path in a selected view. Zero or multiple matches fail resolution hard.

IRI: `https://ekgf.org/ontology/file-system#PathRule`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-file-system:

13.2.17 service shortcut

Label: service shortcut

Identifier: `ekg-file-system:ServiceShortcut`

An entry target that refers to an external provider object (a Google Drive or OneDrive shortcut) rather than to content in the workspace.

IRI: `https://ekgf.org/ontology/file-system#ServiceShortcut`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-file-system:

13.2.18 source kind

Label: source kind

Identifier: `ekg-file-system:SourceKind`

How a file version was observed.

IRI: `https://ekgf.org/ontology/file-system#SourceKind`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-file-system:

13.2.19 suffix pattern

Label: suffix pattern

Identifier: `ekg-file-system:SuffixPattern`

A frozen family of copy-suffix conventions a name family rule may allow.

IRI: `https://ekgf.org/ontology/file-system#SuffixPattern`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-file-system:

13.2.20 symbolic link

Label: symbolic link

Identifier: `ekg-file-system:SymbolicLink`

An entry target that names a path to resolve within the same view, POSIX-style. Resolution is bounded and loop-checked before projection.

IRI: `https://ekgf.org/ontology/file-system#SymbolicLink`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-file-system:

13.2.21 tombstone

Label: tombstone

Identifier: `ekg-file-system:Tombstone`

A recorded deletion: the target it retires and when. Trash/restore flows read tombstones; content survives while retention or references keep it alive.

IRI: `https://ekgf.org/ontology/file-system#Tombstone`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-file-system:

13.3 Properties

13.3.1 allowed suffix pattern

Label: allowed suffix pattern

Identifier: `ekg-file-system:allowedSuffixPattern`
Associates a name-family rule with a suffix pattern that the rule permits.
IRI: `https://ekgf.org/ontology/file-system#allowedSuffixPattern`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-file-system:NameFamilyRule
`rdfs:isDefinedBy`	ekg-file-system:
`rdfs:range`	ekg-file-system:SuffixPattern

13.3.2 also allow final/draft suffix
Label: also allow final/draft suffix
Identifier: `ekg-file-system:alsoAllowFinalDraftSuffix`
Records whether a name-family rule also permits a final or draft suffix.
IRI: `https://ekgf.org/ontology/file-system#alsoAllowFinalDraftSuffix`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-file-system:NameFamilyRule
`rdfs:isDefinedBy`	ekg-file-system:
`rdfs:range`	`xsd:boolean`

13.3.3 archive format
Label: archive format
Identifier: `ekg-file-system:archiveFormat`
The decomposed container's format.
IRI: `https://ekgf.org/ontology/file-system#archiveFormat`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-file-system:ArchiveVersion
`rdfs:isDefinedBy`	ekg-file-system:
`rdfs:range`	ekg-file-system:ArchiveFormat

13.3.4 archive version
Label: archive version
Identifier: `ekg-file-system:archiveVersion`
The decomposed container this member belongs to.
IRI: `https://ekgf.org/ontology/file-system#archiveVersion`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-file-system:ArchiveMember
`rdfs:isDefinedBy`	ekg-file-system:
`rdfs:range`	ekg-file-system:ArchiveVersion

13.3.5 attached to
Label: attached to

Identifier: `ekg-file-system:attachedTo`
The resource the attachment is asserted on.
IRI: `https://ekgf.org/ontology/file-system#attachedTo`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-file-system:Attachment
`rdfs:isDefinedBy`	ekg-file-system:

13.3.6 attachment mode
Label: attachment mode
Identifier: `ekg-file-system:attachmentMode`
Specifies the attachment mode selected for an attachment.
IRI: `https://ekgf.org/ontology/file-system#attachmentMode`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-file-system:Attachment
`rdfs:isDefinedBy`	ekg-file-system:
`rdfs:range`	ekg-file-system:AttachmentMode

13.3.7 attachment role
Label: attachment role
Identifier: `ekg-file-system:attachmentRole`
Specifies the role assigned to an attachment.
IRI: `https://ekgf.org/ontology/file-system#attachmentRole`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-file-system:Attachment
`rdfs:isDefinedBy`	ekg-file-system:
`rdfs:range`	ekg-file-system:AttachmentRole

13.3.8 base name
Label: base name
Identifier: `ekg-file-system:baseName`
Records the base-name component of a name-family rule, separately from its extension and folder.
IRI: `https://ekgf.org/ontology/file-system#baseName`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-file-system:NameFamilyRule
`rdfs:isDefinedBy`	ekg-file-system:
`rdfs:range`	`xsd:string`

13.3.9 byte size
Label: byte size
Identifier: `ekg-file-system:byteSize`

The stored value's length in bytes.

IRI: `https://ekgf.org/ontology/file-system#byteSize`

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:FileVersion
rdfs:isDefinedBy	ekg-file-system:
rdfs:range	xsd:integer

13.3.10 capability profile

Label: capability profile

Identifier: `ekg-file-system:capabilityProfile`

The declared capabilities of the namespace a view projects.

IRI: `https://ekgf.org/ontology/file-system#capabilityProfile`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-file-system:FileSystemView
rdfs:isDefinedBy	ekg-file-system:
rdfs:range	ekg-file-system:CapabilityProfile

13.3.11 case sensitive names

Label: case sensitive names

Identifier: `ekg-file-system:caseSensitiveNames`

Records whether a capability profile treats file and folder names as case-sensitive.

IRI: `https://ekgf.org/ontology/file-system#caseSensitiveNames`

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:CapabilityProfile
rdfs:isDefinedBy	ekg-file-system:
rdfs:range	xsd:boolean

13.3.12 content

Label: content

Identifier: `ekg-file-system: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.

IRI: `https://ekgf.org/ontology/file-system#content`

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:FileVersion
rdfs:isDefinedBy	ekg-file-system:

13.3.13 current version

Label: current version

Identifier: `ekg-file-system:currentVersion`
The series' current version on the branch being viewed. Time-travel reads resolve this pointer as of the commit.
IRI: `https://ekgf.org/ontology/file-system#currentVersion`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-file-system:FileSeries
`rdfs:isDefinedBy`	ekg-file-system:
`rdfs:range`	ekg-file-system:FileVersion

13.3.14 deleted at
Label: deleted at
Identifier: `ekg-file-system:deletedAt`
Records the deletion time represented by a tombstone.
IRI: `https://ekgf.org/ontology/file-system#deletedAt`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-file-system:Tombstone
`rdfs:isDefinedBy`	ekg-file-system:
`rdfs:range`	`xsd:dateTime`

13.3.15 entry count
Label: entry count
Identifier: `ekg-file-system:entryCount`
Non-member entries walked in this container.
IRI: `https://ekgf.org/ontology/file-system#entryCount`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-file-system:ArchiveVersion
`rdfs:isDefinedBy`	ekg-file-system:
`rdfs:range`	`xsd:integer`

13.3.16 entry kind
Label: entry kind
Identifier: `ekg-file-system:entryKind`
What this member entry is.
IRI: `https://ekgf.org/ontology/file-system#entryKind`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-file-system:ArchiveMember
`rdfs:isDefinedBy`	ekg-file-system:
`rdfs:range`	ekg-file-system:ArchiveEntryKind

13.3.17 extension

Label: extension

Identifier: `ekg-file-system:extension`

Records the filename-extension component of a name-family rule.

IRI: `https://ekgf.org/ontology/file-system#extension`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-file-system:NameFamilyRule
`rdfs:isDefinedBy`	ekg-file-system:
`rdfs:range`	`xsd:string`

13.3.18 external id

Label: external id

Identifier: `ekg-file-system:externalId`

The provider-stable object identifier. It identifies the external object, not its bytes.

IRI: `https://ekgf.org/ontology/file-system#externalId`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-file-system:ExternalObject
`rdfs:isDefinedBy`	ekg-file-system:
`rdfs:range`	`xsd:string`

13.3.19 external object

Label: external object

Identifier: `ekg-file-system:externalObject`

The provider object a shortcut or sync relationship refers to.

IRI: `https://ekgf.org/ontology/file-system#externalObject`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:isDefinedBy`	ekg-file-system:
`rdfs:range`	ekg-file-system:ExternalObject

13.3.20 external revision id

Label: external revision id

Identifier: `ekg-file-system:externalRevisionId`

The provider revision a file version was fetched from, when the provider has stable revision IDs.

IRI: `https://ekgf.org/ontology/file-system#externalRevisionId`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:isDefinedBy`	ekg-file-system:
`rdfs:range`	`xsd:string`

13.3.21 folder

Label: folder

Identifier: `ekg-file-system:folder`

Records the folder component of a name-family rule.

IRI: `https://ekgf.org/ontology/file-system#folder`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-file-system:NameFamilyRule
`rdfs:isDefinedBy`	ekg-file-system:
`rdfs:range`	`xsd:string`

13.3.22 has attachment

Label: has attachment

Identifier: `ekg-file-system:hasAttachment`

The resource's attachment assertions.

IRI: `https://ekgf.org/ontology/file-system#hasAttachment`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:isDefinedBy`	ekg-file-system:
`rdfs:range`	ekg-file-system:Attachment

13.3.23 has version

Label: has version

Identifier: `ekg-file-system:hasVersion`

Every observed version of the series.

IRI: `https://ekgf.org/ontology/file-system#hasVersion`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-file-system:FileSeries
`rdfs:isDefinedBy`	ekg-file-system:
`rdfs:range`	ekg-file-system:FileVersion

13.3.24 in view

Label: in view

Identifier: `ekg-file-system:inView`

The view a path rule resolves its path in.

IRI: `https://ekgf.org/ontology/file-system#inView`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-file-system:PathRule
`rdfs:isDefinedBy`	ekg-file-system:
`rdfs:range`	ekg-file-system:FileSystemView

13.3.25 legal name pattern

Label: legal name pattern

Identifier: `ekg-file-system:legalNamePattern`

A regular expression names must match to be legal in the namespace.

IRI: `https://ekgf.org/ontology/file-system#legalNamePattern`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-file-system:CapabilityProfile
`rdfs:isDefinedBy`	ekg-file-system:
`rdfs:range`	`xsd:string`

13.3.26 link target

Label: link target

Identifier: `ekg-file-system:linkTarget`

A link entry's verbatim target.

IRI: `https://ekgf.org/ontology/file-system#linkTarget`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-file-system:ArchiveMember
`rdfs:isDefinedBy`	ekg-file-system:
`rdfs:range`	`xsd:string`

13.3.27 member count

Label: member count

Identifier: `ekg-file-system:memberCount`

Regular members walked in this container.

IRI: `https://ekgf.org/ontology/file-system#memberCount`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-file-system:ArchiveVersion
`rdfs:isDefinedBy`	ekg-file-system:
`rdfs:range`	`xsd:integer`

13.3.28 member mode

Label: member mode

Identifier: `ekg-file-system:memberMode`

POSIX-style mode bits, when the format carries them — reconstruction metadata.

IRI: `https://ekgf.org/ontology/file-system#memberMode`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-file-system:ArchiveMember
`rdfs:isDefinedBy`	ekg-file-system:
`rdfs:range`	`xsd:integer`

13.3.29 member mtime

Label: member mtime

Identifier: `ekg-file-system:memberMtime`

Modification time in seconds since the epoch, when the format carries it — reconstruction metadata.

IRI: `https://ekgf.org/ontology/file-system#memberMtime`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-file-system:ArchiveMember
`rdfs:isDefinedBy`	ekg-file-system:
`rdfs:range`	`xsd:long`

13.3.30 member ordinal

Label: member ordinal

Identifier: `ekg-file-system:memberOrdinal`

The member's zero-based position in container order. Duplicate paths keep distinct ordinals; reconstruction replays this order.

IRI: `https://ekgf.org/ontology/file-system#memberOrdinal`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-file-system:ArchiveMember
`rdfs:isDefinedBy`	ekg-file-system:
`rdfs:range`	`xsd:integer`

13.3.31 member path

Label: member path

Identifier: `ekg-file-system:memberPath`

The member's safe relative path inside the container, verbatim: never absolute, never containing a parent traversal (such containers stay opaque).

IRI: `https://ekgf.org/ontology/file-system#memberPath`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-file-system:ArchiveMember
`rdfs:isDefinedBy`	ekg-file-system:
`rdfs:range`	`xsd:string`

13.3.32 member target

Label: member target

Identifier: `ekg-file-system:memberTarget`

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.

IRI: `https://ekgf.org/ontology/file-system#memberTarget`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-file-system:ArchiveMember
`rdfs:isDefinedBy`	ekg-file-system:

13.3.33 name

Label: name

Identifier: `ekg-file-system:name`

The entry's name in its parent. Names belong to entries, never to files.

IRI: `https://ekgf.org/ontology/file-system#name`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-file-system:DirectoryEntry
`rdfs:isDefinedBy`	ekg-file-system:
`rdfs:range`	`xsd:string`

13.3.34 normalized name

Label: normalized name

Identifier: `ekg-file-system:normalizedName`

The entry name after the view's declared normalization, for duplicate detection.

IRI: `https://ekgf.org/ontology/file-system#normalizedName`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-file-system:DirectoryEntry
`rdfs:isDefinedBy`	ekg-file-system:
`rdfs:range`	`xsd:string`

13.3.35 normalized source file name

Label: normalized source file name

Identifier: `ekg-file-system:normalizedSourceFileName`

Records the normalized source filename retained for a file version, separately from the originally observed filename.

IRI: `https://ekgf.org/ontology/file-system#normalizedSourceFileName`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-file-system:FileVersion
`rdfs:isDefinedBy`	ekg-file-system:
`rdfs:range`	`xsd:string`

13.3.36 observed at

Label: observed at

Identifier: `ekg-file-system:observedAt`

Records when a file version was observed by the source integration.

IRI: `https://ekgf.org/ontology/file-system#observedAt`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-file-system:FileVersion
`rdfs:isDefinedBy`	ekg-file-system:
`rdfs:range`	`xsd:dateTime`

13.3.37 office package kind

Label: office package kind

Identifier: `ekg-file-system:officePackageKind`
The Office Open XML package kind (word, excel, powerpoint) when the container is a recognized Office package.
IRI: `https://ekgf.org/ontology/file-system#officePackageKind`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-file-system:ArchiveVersion
`rdfs:isDefinedBy`	ekg-file-system:
`rdfs:range`	`xsd:string`

13.3.38 opaque reason
Label: opaque reason
Identifier: `ekg-file-system:opaqueReason`
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.
IRI: `https://ekgf.org/ontology/file-system#opaqueReason`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-file-system:FileVersion
`rdfs:isDefinedBy`	ekg-file-system:
`rdfs:range`	`xsd:string`

13.3.39 parent
Label: parent
Identifier: `ekg-file-system:parent`
The directory an entry places its target in. Parents belong to entries, so a target may have many parents.
IRI: `https://ekgf.org/ontology/file-system#parent`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-file-system:DirectoryEntry
`rdfs:isDefinedBy`	ekg-file-system:
`rdfs:range`	ekg-file-system:Directory

13.3.40 provider
Label: provider
Identifier: `ekg-file-system:provider`
The provider a connector profile resolves; vendor API fields and credentials stay in DRC connector profiles, never here.
IRI: `https://ekgf.org/ontology/file-system#provider`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-file-system:ExternalObject

RDF predicate	Value
rdfs:isDefinedBy	ekg-file-system:
rdfs:range	xsd:string

13.3.41 root

Label: root

Identifier: `ekg-file-system:root`

The view's root directory.

IRI: `https://ekgf.org/ontology/file-system#root`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-file-system:FileSystemView
rdfs:isDefinedBy	ekg-file-system:
rdfs:range	ekg-file-system:Directory

13.3.42 source file name

Label: source file name

Identifier: `ekg-file-system:sourceFileName`

Records the filename observed for a file version in its source.

IRI: `https://ekgf.org/ontology/file-system#sourceFileName`

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:FileVersion
rdfs:isDefinedBy	ekg-file-system:
rdfs:range	xsd:string

13.3.43 source kind

Label: source kind

Identifier: `ekg-file-system:sourceKind`

Classifies the source from which a file version was obtained using a SourceKind value.

IRI: `https://ekgf.org/ontology/file-system#sourceKind`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-file-system:FileVersion
rdfs:isDefinedBy	ekg-file-system:
rdfs:range	ekg-file-system:SourceKind

13.3.44 supports atomic move

Label: supports atomic move

Identifier: `ekg-file-system:supportsAtomicMove`

Records whether the source capability profile supports atomic move operations.

IRI: `https://ekgf.org/ontology/file-system#supportsAtomicMove`

RDF predicate	Value
rdf:type	owl:DatatypeProperty

RDF predicate	Value
rdfs:domain	ekg-file-system:CapabilityProfile
rdfs:isDefinedBy	ekg-file-system:
rdfs:range	xsd:boolean

13.3.45 supports links

Label: supports links

Identifier: `ekg-file-system:supportsLinks`

Records whether the source capability profile supports links.

IRI: `https://ekgf.org/ontology/file-system#supportsLinks`

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:CapabilityProfile
rdfs:isDefinedBy	ekg-file-system:
rdfs:range	xsd:boolean

13.3.46 supports multiple roots

Label: supports multiple roots

Identifier: `ekg-file-system:supportsMultipleRoots`

Records whether the source capability profile supports multiple roots.

IRI: `https://ekgf.org/ontology/file-system#supportsMultipleRoots`

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:CapabilityProfile
rdfs:isDefinedBy	ekg-file-system:
rdfs:range	xsd:boolean

13.3.47 supports range reads

Label: supports range reads

Identifier: `ekg-file-system:supportsRangeReads`

Records whether the source capability profile supports reading a selected byte range.

IRI: `https://ekgf.org/ontology/file-system#supportsRangeReads`

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:CapabilityProfile
rdfs:isDefinedBy	ekg-file-system:
rdfs:range	xsd:boolean

13.3.48 supports server-side copy

Label: supports server-side copy

Identifier: `ekg-file-system:supportsServerSideCopy`

Records whether the source capability profile supports copying within the source system without downloading and re-uploading the file.

IRI: `https://ekgf.org/ontology/file-system#supportsServerSideCopy`

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:CapabilityProfile
rdfs:isDefinedBy	ekg-file-system:
rdfs:range	xsd:boolean

13.3.49 supports shortcuts

Label: supports shortcuts

Identifier: `ekg-file-system:supportsShortcuts`

Records whether the source capability profile supports shortcut resources.

IRI: `https://ekgf.org/ontology/file-system#supportsShortcuts`

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:CapabilityProfile
rdfs:isDefinedBy	ekg-file-system:
rdfs:range	xsd:boolean

13.3.50 supports stable revision ids

Label: supports stable revision ids

Identifier: `ekg-file-system:supportsStableRevisionIds`

Records whether the source capability profile supplies stable revision identifiers.

IRI: `https://ekgf.org/ontology/file-system#supportsStableRevisionIds`

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:CapabilityProfile
rdfs:isDefinedBy	ekg-file-system:
rdfs:range	xsd:boolean

13.3.51 supports trash

Label: supports trash

Identifier: `ekg-file-system:supportsTrash`

Records whether the source capability profile supports a trash or recycle-bin facility.

IRI: `https://ekgf.org/ontology/file-system#supportsTrash`

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:CapabilityProfile
rdfs:isDefinedBy	ekg-file-system:
rdfs:range	xsd:boolean

13.3.52 supports versioning

Label: supports versioning

Identifier: `ekg-file-system:supportsVersioning`

Records whether the source capability profile supports file versions.

IRI: `https://ekgf.org/ontology/file-system#supportsVersioning`

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:CapabilityProfile
rdfs:isDefinedBy	ekg-file-system:
rdfs:range	xsd:boolean

13.3.53 symbolic target path

Label: symbolic target path

Identifier: `ekg-file-system:symbolicTargetPath`

Records the target path named by a symbolic link.

IRI: `https://ekgf.org/ontology/file-system#symbolicTargetPath`

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:SymbolicLink
rdfs:isDefinedBy	ekg-file-system:
rdfs:range	xsd:string

13.3.54 target

Label: target

Identifier: `ekg-file-system:target`

What the entry names: a directory, file series, symbolic link, or service shortcut.

IRI: `https://ekgf.org/ontology/file-system#target`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-file-system:DirectoryEntry
rdfs:isDefinedBy	ekg-file-system:

13.3.55 target external object

Label: target external object

Identifier: `ekg-file-system:targetExternalObject`

The provider-stable object a track-external-id attachment follows.

IRI: `https://ekgf.org/ontology/file-system#targetExternalObject`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-file-system:Attachment
rdfs:isDefinedBy	ekg-file-system:
rdfs:range	ekg-file-system:ExternalObject

13.3.56 target name family rule

Label: target name family rule

Identifier: `ekg-file-system:targetNameFamilyRule`

Identifies the rule governing the filename family of an attachment in its target view.

IRI: `https://ekgf.org/ontology/file-system#targetNameFamilyRule`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-file-system:Attachment
rdfs:isDefinedBy	ekg-file-system:
rdfs:range	ekg-file-system:NameFamilyRule

13.3.57 target path rule

Label: target path rule

Identifier: `ekg-file-system:targetPathRule`

Identifies the rule governing the target-view path of an attachment.

IRI: `https://ekgf.org/ontology/file-system#targetPathRule`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-file-system:Attachment
rdfs:isDefinedBy	ekg-file-system:
rdfs:range	ekg-file-system:PathRule

13.3.58 target series

Label: target series

Identifier: `ekg-file-system:targetSeries`

The series a track-series attachment follows.

IRI: `https://ekgf.org/ontology/file-system#targetSeries`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-file-system:Attachment
rdfs:isDefinedBy	ekg-file-system:
rdfs:range	ekg-file-system:FileSeries

13.3.59 target version

Label: target version

Identifier: `ekg-file-system:targetVersion`

The immutable version an exact-version attachment resolves to.

IRI: `https://ekgf.org/ontology/file-system#targetVersion`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-file-system:Attachment
rdfs:isDefinedBy	ekg-file-system:
rdfs:range	ekg-file-system:FileVersion

13.3.60 tombstone of

Label: tombstone of

Identifier: `ekg-file-system:tombstoneOf`

The entry, series, or directory the tombstone retires.

IRI: `https://ekgf.org/ontology/file-system#tombstoneOf`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-file-system:Tombstone
rdfs:isDefinedBy	ekg-file-system:

13.3.61 unicode normalization

Label: unicode normalization

Identifier: `ekg-file-system:unicodeNormalization`

The normalization form the namespace applies to names (NFC, NFD, none).

IRI: `https://ekgf.org/ontology/file-system#unicodeNormalization`

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:CapabilityProfile
rdfs:isDefinedBy	ekg-file-system:
rdfs:range	xsd:string

13.3.62 view path

Label: view path

Identifier: `ekg-file-system:viewPath`

Records the path expression used by a path rule in a target view.

IRI: `https://ekgf.org/ontology/file-system#viewPath`

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-file-system:PathRule
rdfs:isDefinedBy	ekg-file-system:
rdfs:range	xsd:string

13.4 Other Terms

13.4.1 archive directory entry

Label: archive directory entry

Identifier: `ekg-file-system:ArchiveDirectoryEntry`

An explicit directory entry inside the container's own tree — never a workspace directory.

IRI: `https://ekgf.org/ontology/file-system#ArchiveDirectoryEntry`

RDF predicate	Value
rdf:type	ekg-file-system:ArchiveEntryKind
rdfs:isDefinedBy	ekg-file-system:

13.4.2 exact version

Label: exact version

Identifier: `ekg-file-system:ExactVersion`

Always resolves to one immutable file version. The default for evidence: compliance, contracts, audit, decisions, invoices, signed documents.

IRI: `https://ekgf.org/ontology/file-system#ExactVersion`

RDF predicate	Value
rdf:type	ekg-file-system:AttachmentMode
rdfs:isDefinedBy	ekg-file-system:

13.4.3 gzip

Label: gzip

Identifier: `ekg-file-system:Gzip`

A gzip stream wrapping a single non-archive member.

IRI: `https://ekgf.org/ontology/file-system#Gzip`

RDF predicate	Value
rdf:type	ekg-file-system:ArchiveFormat
rdfs:isDefinedBy	ekg-file-system:

13.4.4 hardlink entry

Label: hardlink entry

Identifier: `ekg-file-system:HardlinkEntry`

A hard-link entry; its verbatim target is preserved for reconstruction.

IRI: `https://ekgf.org/ontology/file-system#HardlinkEntry`

RDF predicate	Value
rdf:type	ekg-file-system:ArchiveEntryKind
rdfs:isDefinedBy	ekg-file-system:

13.4.5 native upload

Label: native upload

Identifier: `ekg-file-system:NativeUpload`

The bytes arrived through a direct upload into the workspace.

IRI: `https://ekgf.org/ontology/file-system#NativeUpload`

RDF predicate	Value
rdf:type	ekg-file-system:SourceKind
rdfs:isDefinedBy	ekg-file-system:

13.4.6 nested archive

Label: nested archive

Identifier: `ekg-file-system:NestedArchive`

A member that is itself a recognized container, decomposed recursively into its own archive version.

IRI: `https://ekgf.org/ontology/file-system#NestedArchive`

RDF predicate	Value
rdf:type	ekg-file-system:ArchiveEntryKind
rdfs:isDefinedBy	ekg-file-system:

13.4.7 other entry

Label: other entry

Identifier: `ekg-file-system:OtherEntry`

A recognized entry carried as metadata only (device nodes, FIFOs), preserved by its format-specific tag.

IRI: `https://ekgf.org/ontology/file-system#OtherEntry`

RDF predicate	Value
rdf:type	ekg-file-system:ArchiveEntryKind
rdfs:isDefinedBy	ekg-file-system:

13.4.8 provider sync

Label: provider sync

Identifier: `ekg-file-system:ProviderSync`

The bytes were fetched from an external provider by a connector worker.

IRI: `https://ekgf.org/ontology/file-system#ProviderSync`

RDF predicate	Value
rdf:type	ekg-file-system:SourceKind
rdfs:isDefinedBy	ekg-file-system:

13.4.9 regular file

Label: regular file

Identifier: `ekg-file-system:RegularFile`

A regular member whose decompressed bytes are an ordinary member file version.

IRI: `https://ekgf.org/ontology/file-system#RegularFile`

RDF predicate	Value
rdf:type	ekg-file-system:ArchiveEntryKind
rdfs:isDefinedBy	ekg-file-system:

13.4.10 symlink entry

Label: symlink entry

Identifier: `ekg-file-system:SymlinkEntry`

A symbolic-link entry; its verbatim target is preserved for reconstruction.

IRI: `https://ekgf.org/ontology/file-system#SymlinkEntry`

RDF predicate	Value
rdf:type	ekg-file-system:ArchiveEntryKind
rdfs:isDefinedBy	ekg-file-system:

13.4.11 tar

Label: tar

Identifier: `ekg-file-system:Tar`

A TAR archive.

IRI: `https://ekgf.org/ontology/file-system#Tar`

RDF predicate	Value
rdf:type	ekg-file-system:ArchiveFormat
rdfs:isDefinedBy	ekg-file-system:

13.4.12 tar+gzip

Label: tar+gzip

Identifier: `ekg-file-system:TarGzip`

A gzip stream wrapping a TAR archive.

IRI: `https://ekgf.org/ontology/file-system#TarGzip`

RDF predicate	Value
rdf:type	ekg-file-system:ArchiveFormat
rdfs:isDefinedBy	ekg-file-system:

13.4.13 track external id

Label: track external id

Identifier: `ekg-file-system:TrackExternalId`

Follows a provider-stable file ID; every fetched provider revision still becomes a new file version in one series.

IRI: `https://ekgf.org/ontology/file-system#TrackExternalId`

RDF predicate	Value
rdf:type	ekg-file-system:AttachmentMode
rdfs:isDefinedBy	ekg-file-system:

13.4.14 track name family

Label: track name family

Identifier: `ekg-file-system:TrackNameFamily`

Follows an explicit name-family rule treating copied names as one document stream.

IRI: `https://ekgf.org/ontology/file-system#TrackNameFamily`

RDF predicate	Value
rdf:type	ekg-file-system:AttachmentMode
rdfs:isDefinedBy	ekg-file-system:

13.4.15 track path

Label: track path

Identifier: `ekg-file-system:TrackPath`

Follows whatever series currently occupies a path in a selected view. A view address, not identity; broken or ambiguous resolution fails hard.

IRI: `https://ekgf.org/ontology/file-system#TrackPath`

RDF predicate	Value
rdf:type	ekg-file-system:AttachmentMode
rdfs:isDefinedBy	ekg-file-system:

13.4.16 track series

Label: track series

Identifier: `ekg-file-system:TrackSeries`

Resolves the series' current version as of the branch/commit being viewed — the living document.

IRI: `https://ekgf.org/ontology/file-system#TrackSeries`

RDF predicate	Value
rdf:type	ekg-file-system:AttachmentMode
rdfs:isDefinedBy	ekg-file-system:

13.4.17 windows copy suffix

Label: windows copy suffix

Identifier: `ekg-file-system:WindowsCopySuffix`

The " (2)", " (3)" copy suffixes Windows appends to duplicated names.

IRI: `https://ekgf.org/ontology/file-system#WindowsCopySuffix`

RDF predicate	Value
`rdf:type`	ekg-file-system:SuffixPattern
`rdfs:isDefinedBy`	ekg-file-system:

13.4.18 zip

Label: zip

Identifier: `ekg-file-system:Zip`

A ZIP archive, including Office Open XML packages.

IRI: `https://ekgf.org/ontology/file-system#Zip`

RDF predicate	Value
`rdf:type`	ekg-file-system:ArchiveFormat
`rdfs:isDefinedBy`	ekg-file-system:

14 Lifecycle Ontology

14.1 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.

14.2 Classes

14.2.1 Lifecycle

Label: Lifecycle

Identifier: `ekg-lifecycle: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.

IRI: `https://ekgf.org/ontology/lifecycle#Lifecycle`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-lifecycle:

14.2.2 Phase

Label: Phase

Identifier: `ekg-lifecycle:Phase`

A high-level part of a lifecycle that groups states.

IRI: `https://ekgf.org/ontology/lifecycle#Phase`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-lifecycle:

14.2.3 State

Label: State

Identifier: `ekg-lifecycle:State`

One state of a lifecycle.

IRI: `https://ekgf.org/ontology/lifecycle#State`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-lifecycle:

14.2.4 State transition

Label: State transition

Identifier: `ekg-lifecycle:StateTransition`

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.

IRI: `https://ekgf.org/ontology/lifecycle#StateTransition`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-lifecycle:
owl:disjointWith	ekg-lifecycle:Transition

14.2.5 Transition

Label: Transition

Identifier: `ekg-lifecycle: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.

IRI: `https://ekgf.org/ontology/lifecycle#Transition`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-lifecycle:

14.3 Properties

14.3.1 applies to

Label: applies to

Identifier: `ekg-lifecycle:appliesTo`

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.

IRI: `https://ekgf.org/ontology/lifecycle#appliesTo`

RDF predicate	Value
`rdf:type`	`owl:AnnotationProperty`
`rdfs:domain`	ekg-lifecycle:Lifecycle
`rdfs:isDefinedBy`	ekg-lifecycle:
`rdfs:range`	`rdfs:Class`

14.3.2 current state

Label: current state

Identifier: `ekg-lifecycle:currentState`

The state entered by this occurrence. Its inLifecycle identifies the lifecycle; the occurrence does not duplicate that assignment.

IRI: `https://ekgf.org/ontology/lifecycle#currentState`

RDF predicate	Value
`rdf:type`	`owl:FunctionalProperty`
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-lifecycle:StateTransition
`rdfs:isDefinedBy`	ekg-lifecycle:
`rdfs:range`	ekg-lifecycle:State

14.3.3 follows lifecycle

Label: follows lifecycle

Identifier: `ekg-lifecycle:followsLifecycle`

A lifecycle this thing follows. When explicit assignments exist, their complete set replaces the class defaults from appliesTo. A thing can follow several lifecycles.

IRI: `https://ekgf.org/ontology/lifecycle#followsLifecycle`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:isDefinedBy`	ekg-lifecycle:
`rdfs:range`	ekg-lifecycle:Lifecycle

14.3.4 from state

Label: from state

Identifier: `ekg-lifecycle:fromState`

The source state of a permitted transition.

IRI: `https://ekgf.org/ontology/lifecycle#fromState`

RDF predicate	Value
rdf:type	owl:FunctionalProperty
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-lifecycle:Transition
rdfs:isDefinedBy	ekg-lifecycle:
rdfs:range	ekg-lifecycle:State

14.3.5 in lifecycle

Label: in lifecycle

Identifier: `ekg-lifecycle:inLifecycle`

The lifecycle that a phase, state or transition belongs to.

IRI: `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:isDefinedBy	ekg-lifecycle:
rdfs:range	ekg-lifecycle:Lifecycle

14.3.6 in state

Label: in state

Identifier: `ekg-lifecycle:inState`

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.

IRI: `https://ekgf.org/ontology/lifecycle#inState`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:isDefinedBy	ekg-lifecycle:
rdfs:range	ekg-lifecycle:StateTransition

14.3.7 is terminal

Label: is terminal

Identifier: `ekg-lifecycle:isTerminal`

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.

IRI: `https://ekgf.org/ontology/lifecycle#isTerminal`

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-lifecycle:State
rdfs:isDefinedBy	ekg-lifecycle:
rdfs:range	xsd:boolean

14.3.8 order

Label: order

Identifier: `ekg-lifecycle:order`

Display position of a state within its lifecycle. Order does not imply a permitted transition.

IRI: `https://ekgf.org/ontology/lifecycle#order`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdf:type`	`owl:FunctionalProperty`
`rdfs:domain`	ekg-lifecycle:State
`rdfs:isDefinedBy`	ekg-lifecycle:
`rdfs:range`	`xsd:integer`

14.3.9 phase

Label: phase

Identifier: `ekg-lifecycle:phase`

The phase that a state belongs to.

IRI: `https://ekgf.org/ontology/lifecycle#phase`

RDF predicate	Value
`rdf:type`	`owl:FunctionalProperty`
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-lifecycle:State
`rdfs:isDefinedBy`	ekg-lifecycle:
`rdfs:range`	ekg-lifecycle:Phase

14.3.10 prior state transition

Label: prior state transition

Identifier: `ekg-lifecycle:priorStateTransition`

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.

IRI: `https://ekgf.org/ontology/lifecycle#priorStateTransition`

RDF predicate	Value
`rdf:type`	`owl:FunctionalProperty`
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-lifecycle:StateTransition
`rdfs:isDefinedBy`	ekg-lifecycle:
`rdfs:range`	ekg-lifecycle:StateTransition

14.3.11 to state

Label: to state

Identifier: `ekg-lifecycle:toState`

The target state of a permitted transition.

IRI: `https://ekgf.org/ontology/lifecycle#toState`

RDF predicate	Value
rdf:type	owl:FunctionalProperty
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-lifecycle:Transition
rdfs:isDefinedBy	ekg-lifecycle:
rdfs:range	ekg-lifecycle:State

14.3.12 trigger

Label: trigger

Identifier: `ekg-lifecycle: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.

IRI: `https://ekgf.org/ontology/lifecycle#trigger`

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-lifecycle:Transition
rdfs:isDefinedBy	ekg-lifecycle:
rdfs:range	rdf:langString

14.4 Named Individuals

14.4.1 Lifecycle Validation Shapes

Label: Lifecycle Validation Shapes

Identifier: `ekg-lifecycle:ValidationShapeSet`

Instance of: ekg-shape:ShapeSet

Validates permitted lifecycle transitions, state-entry history, effective current heads, and migration from direct state assignments.

IRI: `https://ekgf.org/ontology/lifecycle#ValidationShapeSet`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-shape:ShapeSet
rdfs:isDefinedBy	ekg-lifecycle:
ekg-shape:hasShape	ekg-lifecycle:CurrentStateShape
ekg-shape:hasShape	ekg-lifecycle:HistoryStateShape
ekg-shape:hasShape	ekg-lifecycle:LegacyStateShape
ekg-shape:hasShape	ekg-lifecycle:StateTransitionShape
ekg-shape:hasShape	ekg-lifecycle:TransitionShape
ekg-shape:hasShapesGraph	ekg-lifecycle:
ekg-shape:shapePurpose	ekg-shape:Validation

14.5 Shapes

14.5.1 Current State Shape

Label: Current State Shape

Identifier: `ekg-lifecycle:CurrentStateShape`

Validates effective lifecycle selection and current history-head assignments, including explicitly selected lifecycles and class defaults.

IRI: `https://ekgf.org/ontology/lifecycle#CurrentStateShape`

RDF predicate	Value
`rdf:type`	`sh:NodeShape`
`rdfs:isDefinedBy`	ekg-lifecycle:
`sh:property`	(`sh:class`: ekg-lifecycle:StateTransition; `sh:path`: ekg-lifecycle:inState)
`sh:property`	(`sh:class`: ekg-lifecycle:Lifecycle; `sh:path`: ekg-lifecycle:followsLifecycle)
`sh:sparql`	(`sh:message`: A current history head cannot have a recorded successor.; `sh:select`: PREFIX lifecycle: <https://ekgf.org/ontology/lifecycle#> SELECT $this WHERE { $this lifecycle:inState ?head . ?successor lifecycle:priorStateTransition ?head . })
`sh:sparql`	(`sh:message`: Each effective lifecycle requires a current history head.; `sh:select`: 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 . } })
`sh:sparql`	(`sh:message`: A current history head must belong to an effective lifecycle.; `sh:select`: 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 . }) })
`sh:sparql`	(`sh:message`: An object must have only one current history head per lifecycle.; `sh:select`: 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) })
`sh:targetSubjectsOf`	`rdf:type`
`sh:targetSubjectsOf`	ekg-lifecycle:followsLifecycle
`sh:targetSubjectsOf`	ekg-lifecycle:inState

14.5.2 History State Shape

Label: History State Shape

Identifier: `ekg-lifecycle:HistoryStateShape`

Validates the State resources referenced by history occurrences, requiring each state to identify exactly one lifecycle.

IRI: `https://ekgf.org/ontology/lifecycle#HistoryStateShape`

RDF predicate	Value
`rdf:type`	`sh:NodeShape`
`rdfs:isDefinedBy`	ekg-lifecycle:
`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

14.5.3 Legacy State Shape

Label: Legacy State Shape

Identifier: `ekg-lifecycle:LegacyStateShape`

Rejects use of the removed hasState predicate. Current assignments use inState with StateTransition occurrences instead.

IRI: `https://ekgf.org/ontology/lifecycle#LegacyStateShape`

RDF predicate	Value
`rdf:type`	`sh:NodeShape`
`rdfs:isDefinedBy`	ekg-lifecycle:
`sh:property`	(`sh:maxCount`: 0; `sh:message`: hasState was removed. Use inState with a StateTransition occurrence.; `sh:path`: `ekg-lifecycle:hasState`)
`sh:targetSubjectsOf`	`ekg-lifecycle:hasState`

14.5.4 State Transition Shape

Label: State Transition Shape

Identifier: `ekg-lifecycle:StateTransitionShape`

Validates a recorded state-transition event, including its current state and predecessor transition where supplied.

IRI: `https://ekgf.org/ontology/lifecycle#StateTransitionShape`

RDF predicate	Value
`rdf:type`	`sh:NodeShape`
`rdfs:isDefinedBy`	ekg-lifecycle:
`sh:class`	ekg-lifecycle:StateTransition
`sh:not`	(`sh:class`: ekg-lifecycle:Transition)
`sh:property`	(`sh:class`: ekg-lifecycle:StateTransition; `sh:maxCount`: 1; `sh:path`: ekg-lifecycle:priorStateTransition)
`sh:property`	(`sh:class`: ekg-lifecycle:State; `sh:maxCount`: 1; `sh:minCount`: 1; `sh:path`: ekg-lifecycle:currentState)
`sh:sparql`	(`sh:message`: A state-entry occurrence cannot belong to histories of different objects.; `sh:select`:

RDF predicate	Value
	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) })
`sh:sparql`	(`sh:message`: State history must not contain predecessor cycles.; `sh:select`: PREFIX lifecycle: <https://ekgf.org/ontology/lifecycle#> SELECT $this WHERE { $this lifecycle:priorStateTransition+ $this . })
`sh:sparql`	(`sh:message`: Successive occurrences must follow a permitted transition within the same lifecycle.; `sh:select`: 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 . } })
`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

14.5.5 Transition Shape

Label: Transition Shape

Identifier: `ekg-lifecycle:TransitionShape`

Validates a lifecycle transition's lifecycle, source state, destination state, and trigger.

IRI: `https://ekgf.org/ontology/lifecycle#TransitionShape`

RDF predicate	Value
`rdf:type`	`sh:NodeShape`
`rdfs:isDefinedBy`	ekg-lifecycle:
`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: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:property`	(`sh:datatype`: `rdf:langString`; `sh:path`: ekg-lifecycle:trigger)
`sh:sparql`	(`sh:message`: A terminal state cannot be the source of a transition.; `sh:select`: PREFIX lifecycle:

RDF predicate	Value
	<https://ekgf.org/ontology/lifecycle#> SELECT $this WHERE { $this lifecycle:fromState ? state . ?state lifecycle:isTerminal true . })
`sh:sparql`	(`sh:message`: Both endpoint states must belong only to the transition's lifecycle.; `sh:select`: 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) }) })
`sh:targetClass`	ekg-lifecycle:Transition
`sh:targetSubjectsOf`	ekg-lifecycle:fromState
`sh:targetSubjectsOf`	ekg-lifecycle:toState
`sh:targetSubjectsOf`	ekg-lifecycle:trigger

15 Maturity Model Ontology

15.1 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.

15.2 Classes

15.2.1 Capability

Label: Capability

Identifier: `ekg-mm:Capability`

An ability assessed within the maturity model and associated with a capability area.

IRI: `https://ekgf.org/ontology/ekgf-maturity-model#Capability`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	https://ekgf.org/ontology/ekgf-maturity-model
`ekg-mm:iriLocalTypeName`	capability

15.2.2 Capability Area

Label: Capability Area

Identifier: `ekg-mm:CapabilityArea`

A subject area within a maturity-model pillar that organizes related capabilities.

IRI: `https://ekgf.org/ontology/ekgf-maturity-model#CapabilityArea`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	https://ekgf.org/ontology/ekgf-maturity-model
`ekg-mm:iriLocalTypeName`	capability-area

15.2.3 Level

Label: Level

Identifier: `ekg-mm:Level`

A maturity level defined within a particular maturity model.

IRI: `https://ekgf.org/ontology/ekgf-maturity-model#Level`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	https://ekgf.org/ontology/ekgf-maturity-model
`ekg-mm:iriLocalTypeName`	level

15.2.4 Model

Label: Model

Identifier: `ekg-mm:Model`

An identified maturity assessment model organizing pillars and levels.

IRI: `https://ekgf.org/ontology/ekgf-maturity-model#Model`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	https://ekgf.org/ontology/ekgf-maturity-model
`ekg-mm:iriLocalTypeName`	model

15.2.5 Pillar

Label: Pillar

Identifier: `ekg-mm:Pillar`

A dimension of a maturity model, such as business, data, organization, or technology.

IRI: `https://ekgf.org/ontology/ekgf-maturity-model#Pillar`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	https://ekgf.org/ontology/ekgf-maturity-model
`ekg-mm:iriLocalTypeName`	pillar

15.3 Properties

15.3.1 Capability Number

Label: Capability Number

Identifier: `ekg-mm:capabilityNumber`

A capability number in the format `<pillar>.<area>.<capability>` where `<pillar>` is an uppercase letter and `<area>` and `<capability>` are integer numbers.

IRI: `https://ekgf.org/ontology/ekgf-maturity-model#capabilityNumber`

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:isDefinedBy	https://ekgf.org/ontology/ekgf-maturity-model
rdfs:range	xsd:string

15.3.2 hasCapability

Label: hasCapability

Identifier: ekg-mm:hasCapability

Associates a pillar or capability area with a capability it includes.

IRI: 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:isDefinedBy	https://ekgf.org/ontology/ekgf-maturity-model
rdfs:range	ekg-mm:Capability

15.3.3 hasLevel

Label: hasLevel

Identifier: ekg-mm:hasLevel

Associates a maturity model with a level defined in that model; it is the inverse of levelInModel.

IRI: https://ekgf.org/ontology/ekgf-maturity-model#hasLevel

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-mm:Model
rdfs:isDefinedBy	https://ekgf.org/ontology/ekgf-maturity-model
rdfs:range	ekg-mm:Level
owl:inverseOf	ekg-mm:levelInModel

15.3.4 hasPillar

Label: hasPillar

Identifier: ekg-mm:hasPillar

Associates a maturity model with one of its pillars; it is the inverse of pillarInModel.

IRI: https://ekgf.org/ontology/ekgf-maturity-model#hasPillar

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-mm:Model
rdfs:isDefinedBy	https://ekgf.org/ontology/ekgf-maturity-model
rdfs:range	ekg-mm:Pillar
owl:inverseOf	ekg-mm:pillarInModel

15.3.5 inArea

Label: inArea

Identifier: `ekg-mm:inArea`

Identifies the capability area containing a maturity-model capability.

IRI: `https://ekgf.org/ontology/ekgf-maturity-model#inArea`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-mm:Capability
`rdfs:isDefinedBy`	https://ekgf.org/ontology/ekgf-maturity-model
`rdfs:range`	ekg-mm:CapabilityArea

15.3.6 inModel

Label: inModel

Identifier: `ekg-mm:inModel`

Relates a capability, capability area, level, or pillar to the maturity model in which it participates. The property is declared transitive.

IRI: `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:isDefinedBy`	https://ekgf.org/ontology/ekgf-maturity-model
`rdfs:range`	ekg-mm:Model

15.3.7 inPillar

Label: inPillar

Identifier: `ekg-mm:inPillar`

Identifies the pillar containing a capability area.

IRI: `https://ekgf.org/ontology/ekgf-maturity-model#inPillar`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-mm:CapabilityArea
`rdfs:isDefinedBy`	https://ekgf.org/ontology/ekgf-maturity-model
`rdfs:range`	ekg-mm:Pillar

15.3.8 IRI local name

Label: IRI local name

Identifier: `ekg-mm:iriLocalName`

The "local name" of the given object as it is used in IRIs or URIs

IRI: `https://ekgf.org/ontology/ekgf-maturity-model#iriLocalName`

RDF predicate	Value
`rdf:type`	`owl:AnnotationProperty`
`rdfs:isDefinedBy`	https://ekgf.org/ontology/ekgf-maturity-model

15.3.9 IRI local type name

Label: IRI local type name

Identifier: `ekg-mm:iriLocalTypeName`

The "local type name" of the given object as it is used in IRIs or URIs

IRI: `https://ekgf.org/ontology/ekgf-maturity-model#iriLocalTypeName`

RDF predicate	Value
`rdf:type`	`owl:AnnotationProperty`
`rdfs:isDefinedBy`	https://ekgf.org/ontology/ekgf-maturity-model

15.3.10 levelInModel

Label: levelInModel

Identifier: `ekg-mm:levelInModel`

Identifies the maturity model defining a level. The functional-property axiom limits each level resource to one model.

IRI: `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:isDefinedBy`	https://ekgf.org/ontology/ekgf-maturity-model
`rdfs:range`	ekg-mm:Model

15.3.11 pillarInModel

Label: pillarInModel

Identifier: `ekg-mm:pillarInModel`

Identifies the maturity model containing a pillar. The functional-property axiom limits each pillar resource to one model.

IRI: `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:isDefinedBy`	https://ekgf.org/ontology/ekgf-maturity-model
`rdfs:range`	ekg-mm:Model

15.4 Named Individuals

15.4.1 Business Pillar

Label: Business Pillar

Identifier: `ekg-mm:BusinessPillar`

Instance of: ekg-mm:Pillar

The business dimension of the example EKGF maturity model.

IRI: `https://ekgf.org/ontology/ekgf-maturity-model#BusinessPillar`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`

RDF predicate	Value
rdf:type	ekg-mm:Pillar
rdfs:isDefinedBy	https://ekgf.org/ontology/ekgf-maturity-model
ekg-mm:inModel	ekg-mm:EKGFMaturityModel
ekg-mm:iriLocalName	business

15.4.2 Data Pillar

Label: Data Pillar

Identifier: `ekg-mm:DataPillar`

Instance of: ekg-mm:Pillar

The data dimension of the example EKGF maturity model.

IRI: `https://ekgf.org/ontology/ekgf-maturity-model#DataPillar`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-mm:Pillar
rdfs:isDefinedBy	https://ekgf.org/ontology/ekgf-maturity-model
ekg-mm:iriLocalName	data

15.4.3 Test EKG/MM

Label: Test EKG/MM

Identifier: `ekg-mm:EKGFMaturityModel`

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.

IRI: `https://ekgf.org/ontology/ekgf-maturity-model#EKGFMaturityModel`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-mm:Model
rdfs:isDefinedBy	https://ekgf.org/ontology/ekgf-maturity-model
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

15.4.4 Level 1

Label: Level 1

Identifier: `ekg-mm:Level1`

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.

IRI: `https://ekgf.org/ontology/ekgf-maturity-model#Level1`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-mm:Level
`rdfs:isDefinedBy`	https://ekgf.org/ontology/ekgf-maturity-model
`ekg-mm:iriLocalName`	1

15.4.5 Level 2

Label: Level 2

Identifier: `ekg-mm:Level2`

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.

IRI: `https://ekgf.org/ontology/ekgf-maturity-model#Level2`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-mm:Level
`rdfs:isDefinedBy`	https://ekgf.org/ontology/ekgf-maturity-model
`ekg-mm:iriLocalName`	2

15.4.6 Level 3

Label: Level 3

Identifier: `ekg-mm:Level3`

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.

IRI: `https://ekgf.org/ontology/ekgf-maturity-model#Level3`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-mm:Level
`rdfs:isDefinedBy`	https://ekgf.org/ontology/ekgf-maturity-model
`ekg-mm:iriLocalName`	3

15.4.7 Level 4

Label: Level 4

Identifier: `ekg-mm:Level4`

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.

IRI: `https://ekgf.org/ontology/ekgf-maturity-model#Level4`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-mm:Level
rdfs:isDefinedBy	https://ekgf.org/ontology/ekgf-maturity-model
ekg-mm:iriLocalName	4

15.4.8 Level 5

Label: Level 5

Identifier: `ekg-mm:Level5`

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.

IRI: `https://ekgf.org/ontology/ekgf-maturity-model#Level5`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-mm:Level
rdfs:isDefinedBy	https://ekgf.org/ontology/ekgf-maturity-model
ekg-mm:iriLocalName	5

15.4.9 Organization Pillar

Label: Organization Pillar

Identifier: `ekg-mm:OrganizationPillar`

Instance of: ekg-mm:Pillar

The organizational dimension of the example EKGF maturity model.

IRI: `https://ekgf.org/ontology/ekgf-maturity-model#OrganizationPillar`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-mm:Pillar
rdfs:isDefinedBy	https://ekgf.org/ontology/ekgf-maturity-model
ekg-mm:iriLocalName	organization

15.4.10 Technology Pillar

Label: Technology Pillar

Identifier: `ekg-mm:TechnologyPillar`

Instance of: ekg-mm:Pillar

The technology dimension of the example EKGF maturity model.

IRI: `https://ekgf.org/ontology/ekgf-maturity-model#TechnologyPillar`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-mm:Pillar
rdfs:isDefinedBy	https://ekgf.org/ontology/ekgf-maturity-model
ekg-mm:iriLocalName	technology

16 Outcome Ontology

16.1 Overview

Outcome Ontology

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

The outcomes that use cases and stories contribute to. An outcome has a stereotype and a priority. Its lifecycle is described with the Lifecycle Ontology.

16.2 Classes

16.2.1 Outcome

Label: Outcome

Identifier: `ekg-outcome:Outcome`

A result, goal, benefit, or theme to which an EKG use case or story contributes, classified by an outcome stereotype and optionally prioritized.

IRI: `https://ekgf.org/ontology/outcome#Outcome`

RDF predicate	Value
`http://purl.org/dc/elements/1.1/creator`	`http://orcid.org/0000-0003-2309-5068`
`http://purl.org/dc/elements/1.1/date`	2020-04-01T18:12:06Z
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-outcome:

16.2.2 Outcome Stereotype

Label: Outcome Stereotype

Identifier: `ekg-outcome:OutcomeStereoType`

The kinds of outcome.

IRI: `https://ekgf.org/ontology/outcome#OutcomeStereoType`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-outcome:

16.3 Properties

16.3.1 Has Outcome

Label: Has Outcome

Identifier: `ekg-outcome:hasOutcome`

Relates a resource, such as a use case or story, to an outcome it contributes to. No domain restriction is declared.

IRI: `https://ekgf.org/ontology/outcome#hasOutcome`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:isDefinedBy`	ekg-outcome:
`rdfs:range`	ekg-outcome:Outcome

16.3.2 priority

Label: priority

Identifier: `ekg-outcome:priority`

The priority of the outcome, as an integer.

IRI: `https://ekgf.org/ontology/outcome#priority`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-outcome:Outcome
`rdfs:isDefinedBy`	ekg-outcome:
`rdfs:range`	`xsd:integer`

16.3.3 stereotype

Label: stereotype

Identifier: `ekg-outcome:stereotype`

The kind of outcome: theme, goal, objective or benefit.

IRI: `https://ekgf.org/ontology/outcome#stereotype`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-outcome:Outcome
`rdfs:isDefinedBy`	ekg-outcome:
`rdfs:range`	ekg-outcome:OutcomeStereoType

16.3.4 updated at

Label: updated at

Identifier: `ekg-outcome:updatedAt`

When the outcome was last changed.

IRI: `https://ekgf.org/ontology/outcome#updatedAt`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-outcome:Outcome
`rdfs:isDefinedBy`	ekg-outcome:
`rdfs:range`	`xsd:dateTime`

16.4 Named Individuals

16.4.1 Benefit

Label: Benefit

Identifier: `ekg-outcome:OutcomeStereoTypeBenefit`

Instance of: ekg-outcome:OutcomeStereoType

The outcome stereotype used to classify an outcome as a benefit.

IRI: `https://ekgf.org/ontology/outcome#OutcomeStereoTypeBenefit`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-outcome:OutcomeStereoType

RDF predicate	Value
rdfs:isDefinedBy	ekg-outcome:

16.4.2 Goal

Label: Goal

Identifier: `ekg-outcome:OutcomeStereoTypeGoal`

Instance of: ekg-outcome:OutcomeStereoType

The outcome stereotype used to classify an outcome as a goal.

IRI: `https://ekgf.org/ontology/outcome#OutcomeStereoTypeGoal`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-outcome:OutcomeStereoType
rdfs:isDefinedBy	ekg-outcome:

16.4.3 Objective

Label: Objective

Identifier: `ekg-outcome:OutcomeStereoTypeObjective`

Instance of: ekg-outcome:OutcomeStereoType

The outcome stereotype used to classify an outcome as an objective.

IRI: `https://ekgf.org/ontology/outcome#OutcomeStereoTypeObjective`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-outcome:OutcomeStereoType
rdfs:isDefinedBy	ekg-outcome:

16.4.4 Theme

Label: Theme

Identifier: `ekg-outcome:OutcomeStereoTypeTheme`

Instance of: ekg-outcome:OutcomeStereoType

The outcome stereotype used to classify an outcome as a guiding theme.

IRI: `https://ekgf.org/ontology/outcome#OutcomeStereoTypeTheme`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-outcome:OutcomeStereoType
rdfs:isDefinedBy	ekg-outcome:

16.4.5 Accountability

Label: Accountability

Identifier: `ekg-outcome:ThemeAccountability`

Instance of: ekg-outcome:Outcome

A theme outcome for recording goals and benefits concerning accountability, linked to the corresponding EKG principle for further explanation.

IRI: `https://ekgf.org/ontology/outcome#ThemeAccountability`

RDF predicate	Value
rdf:type	owl:NamedIndividual

RDF predicate	Value
rdf:type	ekg-outcome:Outcome
rdfs:isDefinedBy	ekg-outcome:
rdfs:seeAlso	https://principles.ekgf.org/theme/accountability/
ekg-outcome:stereotype	ekg-outcome:OutcomeStereoTypeTheme

16.4.6 Composable Business

Label: Composable Business

Identifier: `ekg-outcome:ThemeComposableBusiness`

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.

IRI: `https://ekgf.org/ontology/outcome#ThemeComposableBusiness`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-outcome:Outcome
rdfs:isDefinedBy	ekg-outcome:
rdfs:seeAlso	https://principles.ekgf.org/theme/composable-business/
ekg-outcome:stereotype	ekg-outcome:OutcomeStereoTypeTheme

16.4.7 Contextual Truth

Label: Contextual Truth

Identifier: `ekg-outcome:ThemeContextualTruth`

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.

IRI: `https://ekgf.org/ontology/outcome#ThemeContextualTruth`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-outcome:Outcome
rdfs:isDefinedBy	ekg-outcome:
rdfs:seeAlso	https://principles.ekgf.org/theme/contextual-truth/
ekg-outcome:stereotype	ekg-outcome:OutcomeStereoTypeTheme

16.4.8 Digital Assets

Label: Digital Assets

Identifier: `ekg-outcome:ThemeDigitalAssets`

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.

IRI: `https://ekgf.org/ontology/outcome#ThemeDigitalAssets`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-outcome:Outcome

RDF predicate	Value
rdfs:isDefinedBy	ekg-outcome:
rdfs:seeAlso	https://principles.ekgf.org/theme/digital-assets/
ekg-outcome:stereotype	ekg-outcome:OutcomeStereoTypeTheme

16.4.9 Environmental Sustainability

Label: Environmental Sustainability

Identifier: `ekg-outcome:ThemeEnvironmentalSustainability`

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.

IRI: `https://ekgf.org/ontology/outcome#ThemeEnvironmentalSustainability`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-outcome:Outcome
rdfs:isDefinedBy	ekg-outcome:
rdfs:seeAlso	https://principles.ekgf.org/theme/environmental-sustainability/
ekg-outcome:stereotype	ekg-outcome:OutcomeStereoTypeTheme

16.4.10 Fairness

Label: Fairness

Identifier: `ekg-outcome:ThemeFairness`

Instance of: ekg-outcome:Outcome

A theme outcome for recording goals and benefits concerning fairness, linked to the corresponding EKG principle for further explanation.

IRI: `https://ekgf.org/ontology/outcome#ThemeFairness`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-outcome:Outcome
rdfs:isDefinedBy	ekg-outcome:
rdfs:seeAlso	https://principles.ekgf.org/theme/fairness/
ekg-outcome:stereotype	ekg-outcome:OutcomeStereoTypeTheme

16.4.11 Manage Profitability

Label: Manage Profitability

Identifier: `ekg-outcome:ThemeManageProfitability`

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.

IRI: `https://ekgf.org/ontology/outcome#ThemeManageProfitability`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-outcome:Outcome

RDF predicate	Value
rdfs:isDefinedBy	ekg-outcome:
rdfs:seeAlso	https://principles.ekgf.org/theme/manage-profitability/
ekg-outcome:stereotype	ekg-outcome:OutcomeStereoTypeTheme

16.4.12 Manage Risk & Compliance

Label: Manage Risk & Compliance

Identifier: `ekg-outcome:ThemeManageRiskAndCompliance`

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.

IRI: `https://ekgf.org/ontology/outcome#ThemeManageRiskAndCompliance`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-outcome:Outcome
rdfs:isDefinedBy	ekg-outcome:
rdfs:seeAlso	https://principles.ekgf.org/theme/manage-risk-and-compliance/
ekg-outcome:stereotype	ekg-outcome:OutcomeStereoTypeTheme

16.4.13 Openness

Label: Openness

Identifier: `ekg-outcome:ThemeOpenness`

Instance of: ekg-outcome:Outcome

A theme outcome for recording goals and benefits concerning openness, linked to the corresponding EKG principle for further explanation.

IRI: `https://ekgf.org/ontology/outcome#ThemeOpenness`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-outcome:Outcome
rdfs:isDefinedBy	ekg-outcome:
rdfs:seeAlso	https://principles.ekgf.org/theme/openness/
ekg-outcome:stereotype	ekg-outcome:OutcomeStereoTypeTheme

16.4.14 Power of Abstraction

Label: Power of Abstraction

Identifier: `ekg-outcome:ThemePowerOfAbstraction`

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.

IRI: `https://ekgf.org/ontology/outcome#ThemePowerOfAbstraction`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-outcome:Outcome

RDF predicate	Value
rdfs:isDefinedBy	ekg-outcome:
rdfs:seeAlso	https://principles.ekgf.org/theme/power-of-abstraction/
ekg-outcome:stereotype	ekg-outcome:OutcomeStereoTypeTheme

16.4.15 Resilience

Label: Resilience

Identifier: `ekg-outcome:ThemeResilience`

Instance of: ekg-outcome:Outcome

A theme outcome for recording goals and benefits concerning resilience, linked to the corresponding EKG principle for further explanation.

IRI: `https://ekgf.org/ontology/outcome#ThemeResilience`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-outcome:Outcome
rdfs:isDefinedBy	ekg-outcome:
rdfs:seeAlso	https://principles.ekgf.org/theme/resilience/
ekg-outcome:stereotype	ekg-outcome:OutcomeStereoTypeTheme

16.4.16 Reusability

Label: Reusability

Identifier: `ekg-outcome:ThemeReusability`

Instance of: ekg-outcome:Outcome

A theme outcome for recording goals and benefits concerning reusability, linked to the corresponding EKG principle for further explanation.

IRI: `https://ekgf.org/ontology/outcome#ThemeReusability`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-outcome:Outcome
rdfs:isDefinedBy	ekg-outcome:
rdfs:seeAlso	https://principles.ekgf.org/theme/reusability/
ekg-outcome:stereotype	ekg-outcome:OutcomeStereoTypeTheme

16.4.17 Sustainability

Label: Sustainability

Identifier: `ekg-outcome:ThemeSustainability`

Instance of: ekg-outcome:Outcome

A theme outcome for recording goals and benefits concerning sustainability, linked to the corresponding EKG principle for further explanation.

IRI: `https://ekgf.org/ontology/outcome#ThemeSustainability`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-outcome:Outcome
rdfs:isDefinedBy	ekg-outcome:

RDF predicate	Value
`rdfs:seeAlso`	https://principles.ekgf.org/theme/sustainability/
`ekg-outcome:stereotype`	ekg-outcome:OutcomeStereoTypeTheme

16.4.18 Transparency

Label: Transparency

Identifier: `ekg-outcome:ThemeTransparency`

Instance of: ekg-outcome:Outcome

A theme outcome for recording goals and benefits concerning transparency, linked to the corresponding EKG principle for further explanation.

IRI: `https://ekgf.org/ontology/outcome#ThemeTransparency`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-outcome:Outcome
`rdfs:isDefinedBy`	ekg-outcome:
`rdfs:seeAlso`	https://principles.ekgf.org/theme/transparency/
`ekg-outcome:stereotype`	ekg-outcome:OutcomeStereoTypeTheme

16.4.19 Unknown Outcome

Label: Unknown Outcome

Identifier: `ekg-outcome:UnknownOutcome`

Instance of: ekg-outcome:Outcome

The default outcome. Think about what your use case or story accomplishes.

IRI: `https://ekgf.org/ontology/outcome#UnknownOutcome`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-outcome:Outcome
`rdfs:isDefinedBy`	ekg-outcome:
`ekg-outcome:stereotype`	ekg-outcome:OutcomeStereoTypeTheme

17 Persona Ontology

17.1 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.

17.2 Classes

17.2.1 Accountability

Label: Accountability

Identifier: `ekg-persona:Accountability`

A responsibility or expected area of ownership assigned to a persona.

IRI: `https://ekgf.org/ontology/persona#Accountability`

RDF predicate	Value
`http://purl.org/dc/elements/1.1/date`	2023-08-18T17:12:00Z
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-persona:

17.2.2 Persona

Label: Persona

Identifier: `ekg-persona: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.

IRI: `https://ekgf.org/ontology/persona#Persona`

RDF predicate	Value
`http://purl.org/dc/elements/1.1/creator`	`http://orcid.org/0000-0003-2309-5068`
`http://purl.org/dc/elements/1.1/date`	2020-04-01T18:10:23Z
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-persona:

17.3 Properties

17.3.1 has accountability

Label: has accountability

Identifier: `ekg-persona:accountability`

Relates a persona to an accountability assigned to it.

IRI: `https://ekgf.org/ontology/persona#accountability`

RDF predicate	Value
`http://purl.org/dc/elements/1.1/date`	2023-08-18T17:12:22Z
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-persona:Persona
`rdfs:isDefinedBy`	ekg-persona:
`rdfs:range`	ekg-persona:Accountability

17.3.2 Is Sub Persona Of

Label: Is Sub Persona Of

Identifier: `ekg-persona:isSubPersonaOf`

Relates a more specific persona to a broader persona. Its irreflexive axiom excludes a persona being its own sub-persona.

IRI: `https://ekgf.org/ontology/persona#isSubPersonaOf`

RDF predicate	Value
`http://purl.org/dc/elements/1.1/creator`	`http://orcid.org/0000-0003-2309-5068`
`http://purl.org/dc/elements/1.1/date`	2020-06-20T14:34:46Z
`rdf:type`	`owl:IrreflexiveProperty`
`rdf:type`	`owl:ObjectProperty`

RDF predicate	Value
rdfs:domain	ekg-persona:Persona
rdfs:isDefinedBy	ekg-persona:
rdfs:range	ekg-persona:Persona

17.3.3 Persona key

Label: Persona key

Identifier: `ekg-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.

IRI: `https://ekgf.org/ontology/persona#key`

RDF predicate	Value
http://purl.org/dc/elements/1.1/creator	http://orcid.org/0000-0003-2309-5068
http://purl.org/dc/elements/1.1/date	2020-05-01T15:41:28Z
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-persona:Persona
rdfs:isDefinedBy	ekg-persona:
rdfs:range	xsd:string

18 Raw Data Ontology

18.1 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.

18.2 Classes

18.2.1 Column

Label: Column

Identifier: `ekg-raw: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.

IRI: `https://ekgf.org/ontology/raw#Column`

RDF predicate	Value
http://purl.org/dc/elements/1.1/creator	http://orcid.org/0000-0003-2309-5068
http://purl.org/dc/elements/1.1/date	2020-06-13T09:09:19Z

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-raw:
rdfs:subClassOf	(rdf:type: owl:Restriction; owl:minCardinality: 1; owl:onProperty: ekg-raw:isColumnInView)

18.2.2 File

Label: File

Identifier: `ekg-raw:File`

A file used as a raw-data source during the capture step of a DataOps pipeline.

IRI: `https://ekgf.org/ontology/raw#File`

RDF predicate	Value
http://purl.org/dc/elements/1.1/creator	http://orcid.org/0000-0003-2309-5068
http://purl.org/dc/elements/1.1/date	2020-06-13T09:26:36Z
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-raw:
rdfs:subClassOf	ekg-raw:Source

18.2.3 Null value

Label: Null value

Identifier: `ekg-raw:NullValue`

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.

IRI: `https://ekgf.org/ontology/raw#NullValue`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-raw:
rdfs:subClassOf	ekg-raw:Value
rdfs:subClassOf	(rdf:type: owl:Restriction; owl:maxCardinality: 0; owl:onProperty: ekg-raw:literalValue)

18.2.4 Row

Label: Row

Identifier: `ekg-raw: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.

IRI: `https://ekgf.org/ontology/raw#Row`

RDF predicate	Value
http://purl.org/dc/elements/1.1/creator	http://orcid.org/0000-0003-2309-5068
http://purl.org/dc/elements/1.1/date	2020-06-13T09:09:08Z
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-raw:

RDF predicate	Value
`rdfs:subClassOf`	(`rdf:type`: `owl:Restriction`; `owl:minCardinality`: 1; `owl:onProperty`: ekg-raw:isRowInView)

18.2.5 Source

Label: Source

Identifier: `ekg-raw:Source`

An input data source captured as RDF before transformation into higher-level domain ontologies. A source may expose several views.

IRI: `https://ekgf.org/ontology/raw#Source`

RDF predicate	Value
`http://purl.org/dc/elements/1.1/creator`	`http://orcid.org/0000-0003-2309-5068`
`http://purl.org/dc/elements/1.1/date`	2020-06-13T09:26:09Z
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-raw:

18.2.6 Value

Label: Value

Identifier: `ekg-raw:Value`

18.2.6.1 Captured content

`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`, registered in `ValidationShapeSet`, checks this contract. OWL models the alternatives and cardinalities; it does not supply missing content or classify an incomplete occurrence as null.

18.2.6.2 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 `ValidationShapeSet`.

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.

IRI: `https://ekgf.org/ontology/raw#Value`

RDF predicate	Value
`http://purl.org/dc/elements/1.1/creator`	`http://orcid.org/0000-0003-2309-5068`
`http://purl.org/dc/elements/1.1/date`	2020-06-13T09:14:07Z
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-raw:
`rdfs:subClassOf`	(`rdf:type`: `owl:Restriction`; `owl:minCardinality`: 1; `owl:onProperty`: ekg-raw:valueOf)
`rdfs:subClassOf`	(`rdf:type`: `owl:Restriction`; `owl:minCardinality`: 1; `owl:onProperty`: ekg-raw:forColumn)
`rdfs:subClassOf`	(`rdf:type`: `owl:Class`; `owl:unionOf`: (ekg-raw:NullValue, (`rdf:type`: `owl:Restriction`; `owl:minCardinality`: 1; `owl:onProperty`: ekg-raw:literalValue)))

18.2.7 View

Label: View

Identifier: `ekg-raw:View`

A tabular or record-oriented view within a captured source, organizing the rows and columns exposed by that source.

IRI: `https://ekgf.org/ontology/raw#View`

RDF predicate	Value
`http://purl.org/dc/elements/1.1/creator`	`http://orcid.org/0000-0003-2309-5068`
`http://purl.org/dc/elements/1.1/date`	2020-06-13T09:25:24Z
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-raw:

18.2.8 XLSX file

Label: XLSX file

Identifier: `ekg-raw:XlsxFile`

A raw-data file source in the Excel XLSX workbook format.

IRI: `https://ekgf.org/ontology/raw#XlsxFile`

RDF predicate	Value
`http://purl.org/dc/elements/1.1/creator`	`http://orcid.org/0000-0003-2309-5068`
`http://purl.org/dc/elements/1.1/date`	2020-06-13T09:26:15Z
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-raw:
`rdfs:subClassOf`	ekg-raw:File

18.3 Properties

18.3.1 25th percentile

Label: 25th percentile

Identifier: `ekg-raw:25pct`

Records the 25th-percentile summary value for a captured column.

IRI: `https://ekgf.org/ontology/raw#25pct`

RDF predicate	Value
`http://purl.org/dc/elements/1.1/creator`	`http://orcid.org/0000-0003-2309-5068`
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-raw:Column
`rdfs:isDefinedBy`	ekg-raw:

18.3.2 50th percentile

Label: 50th percentile

Identifier: `ekg-raw:50pct`

Records the 50th-percentile summary value, or median, for a captured column.

IRI: `https://ekgf.org/ontology/raw#50pct`

RDF predicate	Value
`http://purl.org/dc/elements/1.1/creator`	`http://orcid.org/0000-0003-2309-5068`
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-raw:Column
`rdfs:isDefinedBy`	ekg-raw:

18.3.3 75th percentile

Label: 75th percentile

Identifier: `ekg-raw:75pct`

Records the 75th-percentile summary value for a captured column.

IRI: `https://ekgf.org/ontology/raw#75pct`

RDF predicate	Value
`http://purl.org/dc/elements/1.1/creator`	`http://orcid.org/0000-0003-2309-5068`
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-raw:Column
`rdfs:isDefinedBy`	ekg-raw:

18.3.4 Column number

Label: Column number

Identifier: `ekg-raw:columnNumber`

1-based column index

IRI: `https://ekgf.org/ontology/raw#columnNumber`

RDF predicate	Value
`http://purl.org/dc/elements/1.1/creator`	`http://orcid.org/0000-0003-2309-5068`
`http://purl.org/dc/elements/1.1/date`	2020-06-13T09:31:29Z
`rdf:type`	`owl:DatatypeProperty`
`rdf:type`	`owl:FunctionalProperty`

RDF predicate	Value
rdfs:domain	ekg-raw:Column
rdfs:isDefinedBy	ekg-raw:
rdfs:range	xsd:integer

18.3.5 Count

Label: Count

Identifier: `ekg-raw:count`

Count non-NA cells for each column or row.

IRI: `https://ekgf.org/ontology/raw#count`

RDF predicate	Value
http://purl.org/dc/elements/1.1/creator	http://orcid.org/0000-0003-2309-5068
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-raw:Column
rdfs:isDefinedBy	ekg-raw:
rdfs:seeAlso	https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.count.html#pandas.DataFrame.count

18.3.6 File name

Label: File name

Identifier: `ekg-raw:fileName`

The name of the file as it was passed to the capture step

IRI: `https://ekgf.org/ontology/raw#fileName`

RDF predicate	Value
http://purl.org/dc/elements/1.1/creator	http://orcid.org/0000-0003-2309-5068
http://purl.org/dc/elements/1.1/date	2020-06-13T09:26:52Z
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-raw:File
rdfs:isDefinedBy	ekg-raw:
rdfs:range	xsd:string

18.3.7 For column

Label: For column

Identifier: `ekg-raw:forColumn`

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.

IRI: `https://ekgf.org/ontology/raw#forColumn`

RDF predicate	Value
rdf:type	owl:FunctionalProperty
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-raw:Value

RDF predicate	Value
rdfs:isDefinedBy	ekg-raw:
rdfs:range	ekg-raw:Column

18.3.8 For Predicate

Label: For Predicate

Identifier: `ekg-raw:forPredicate`

The predicate for which the given value was encountered

IRI: `https://ekgf.org/ontology/raw#forPredicate`

RDF predicate	Value
http://purl.org/dc/elements/1.1/creator	http://orcid.org/0000-0003-2309-5068
http://purl.org/dc/elements/1.1/date	2020-06-13T09:14:28Z
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-raw:Value
rdfs:isDefinedBy	ekg-raw:

18.3.9 The most common value's frequency

Label: The most common value's frequency

Identifier: `ekg-raw:freq`

Records the occurrence count of the most frequently occurring value in a captured column.

IRI: `https://ekgf.org/ontology/raw#freq`

RDF predicate	Value
http://purl.org/dc/elements/1.1/creator	http://orcid.org/0000-0003-2309-5068
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-raw:Column
rdfs:isDefinedBy	ekg-raw:
rdfs:seeAlso	https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.describe.html

18.3.10 Has column

Label: Has column

Identifier: `ekg-raw:hasColumn`

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.

IRI: `https://ekgf.org/ontology/raw#hasColumn`

RDF predicate	Value
rdf:type	owl:InverseFunctionalProperty
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-raw:View
rdfs:isDefinedBy	ekg-raw:
rdfs:range	ekg-raw:Column
owl:inverseOf	ekg-raw:isColumnInView

18.3.11 Has View

Label: Has View

Identifier: `ekg-raw:hasView`

One Source can have multiple Views

IRI: `https://ekgf.org/ontology/raw#hasView`

RDF predicate	Value
`http://purl.org/dc/elements/1.1/creator`	`http://orcid.org/0000-0003-2309-5068`
`http://purl.org/dc/elements/1.1/date`	2020-06-13T09:28:29Z
`rdf:type`	`owl:InverseFunctionalProperty`
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-raw:Source
`rdfs:isDefinedBy`	ekg-raw:
`rdfs:range`	ekg-raw:View
`owl:inverseOf`	ekg-raw:isViewIn

18.3.12 Is column in view

Label: Is column in view

Identifier: `ekg-raw:isColumnInView`

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.

IRI: `https://ekgf.org/ontology/raw#isColumnInView`

RDF predicate	Value
`rdf:type`	`owl:FunctionalProperty`
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-raw:Column
`rdfs:isDefinedBy`	ekg-raw:
`rdfs:range`	ekg-raw:View

18.3.13 Is Row In View

Label: Is Row In View

Identifier: `ekg-raw:isRowInView`

The current row occurs in the given view.

IRI: `https://ekgf.org/ontology/raw#isRowInView`

RDF predicate	Value
`http://purl.org/dc/elements/1.1/creator`	`http://orcid.org/0000-0003-2309-5068`
`http://purl.org/dc/elements/1.1/date`	2020-06-13T09:36:41Z
`rdf:type`	`owl:FunctionalProperty`
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-raw:Row
`rdfs:isDefinedBy`	ekg-raw:
`rdfs:range`	ekg-raw:View

18.3.14 Is View In

Label: Is View In

Identifier: `ekg-raw:isViewIn`

This View occurs in the given Source

IRI: `https://ekgf.org/ontology/raw#isViewIn`

RDF predicate	Value
http://purl.org/dc/elements/1.1/creator	http://orcid.org/0000-0003-2309-5068
http://purl.org/dc/elements/1.1/date	2020-06-13T09:30:00Z
rdf:type	owl:FunctionalProperty
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-raw:View
rdfs:isDefinedBy	ekg-raw:
rdfs:range	ekg-raw:Source

18.3.15 Legacy ID

Label: Legacy ID

Identifier: `ekg-raw:legacyId`

The values in the key-column in the given view will always be mapped to raw:legacyId

IRI: `https://ekgf.org/ontology/raw#legacyId`

RDF predicate	Value
http://purl.org/dc/elements/1.1/creator	http://orcid.org/0000-0003-2309-5068
http://purl.org/dc/elements/1.1/date	2020-06-13T09:38:33Z
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-raw:Row
rdfs:isDefinedBy	ekg-raw:
rdfs:range	xsd:string

18.3.16 Legacy Id In Iri

Label: Legacy Id In Iri

Identifier: `ekg-raw:legacyIdInIri`

The value of legacyId translated to an IRI-compliant lower kebab-case value

IRI: `https://ekgf.org/ontology/raw#legacyIdInIri`

RDF predicate	Value
http://purl.org/dc/elements/1.1/creator	http://orcid.org/0000-0003-2309-5068
http://purl.org/dc/elements/1.1/date	2020-06-13T09:40:31Z
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:isDefinedBy	ekg-raw:

18.3.17 Literal value

Label: Literal value

Identifier: `ekg-raw:literalValue`

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.

IRI: `https://ekgf.org/ontology/raw#literalValue`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdf:type`	`owl:FunctionalProperty`
`rdfs:domain`	ekg-raw:Value
`rdfs:isDefinedBy`	ekg-raw:
`rdfs:range`	`rdfs:Literal`

18.3.18 Local name

Label: Local name

Identifier: `ekg-raw:localName`

The local name of the property that this Column is mapped to

IRI: `https://ekgf.org/ontology/raw#localName`

RDF predicate	Value
`http://purl.org/dc/elements/1.1/creator`	`http://orcid.org/0000-0003-2309-5068`
`http://purl.org/dc/elements/1.1/date`	2020-06-13T09:32:26Z
`rdf:type`	`owl:DatatypeProperty`
`rdf:type`	`owl:FunctionalProperty`
`rdfs:domain`	ekg-raw:Column
`rdfs:isDefinedBy`	ekg-raw:
`rdfs:range`	`xsd:string`

18.3.19 Maximum

Label: Maximum

Identifier: `ekg-raw:max`

The maximum of the values for the given axis.

IRI: `https://ekgf.org/ontology/raw#max`

RDF predicate	Value
`http://purl.org/dc/elements/1.1/creator`	`http://orcid.org/0000-0003-2309-5068`
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-raw:Column
`rdfs:isDefinedBy`	ekg-raw:
`rdfs:seeAlso`	https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.max.html#pandas.DataFrame.max

18.3.20 Mean

Label: Mean

Identifier: `ekg-raw:mean`

The mean of the values for the given axis.

IRI: `https://ekgf.org/ontology/raw#mean`

RDF predicate	Value
http://purl.org/dc/elements/1.1/creator	http://orcid.org/0000-0003-2309-5068
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-raw:Column
rdfs:isDefinedBy	ekg-raw:
rdfs:seeAlso	https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.mean.html#pandas.DataFrame.mean

18.3.21 Minimum

Label: Minimum

Identifier: ekg-raw:min

The minimum of the values for the requested axis.

IRI: https://ekgf.org/ontology/raw#min

RDF predicate	Value
http://purl.org/dc/elements/1.1/creator	http://orcid.org/0000-0003-2309-5068
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-raw:Column
rdfs:isDefinedBy	ekg-raw:
rdfs:seeAlso	https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.min.html#pandas.DataFrame.min

18.3.22 Predicate

Label: Predicate

Identifier: ekg-raw: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.

IRI: https://ekgf.org/ontology/raw#predicate

RDF predicate	Value
http://purl.org/dc/elements/1.1/creator	http://orcid.org/0000-0003-2309-5068
http://purl.org/dc/elements/1.1/date	2020-06-13T09:33:37Z
rdf:type	owl:AnnotationProperty
rdfs:domain	ekg-raw:Column
rdfs:isDefinedBy	ekg-raw:
rdfs:range	rdf:Property

18.3.23 Source text

Label: Source text

Identifier: ekg-raw:sourceText

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.

IRI: https://ekgf.org/ontology/raw#sourceText

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-raw:Value
rdfs:isDefinedBy	ekg-raw:
rdfs:range	xsd:string

18.3.24 Standard deviation

Label: Standard deviation

Identifier: `ekg-raw:std`

The sample standard deviation over given axis.

IRI: `https://ekgf.org/ontology/raw#std`

RDF predicate	Value
http://purl.org/dc/elements/1.1/creator	http://orcid.org/0000-0003-2309-5068
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-raw:Column
rdfs:isDefinedBy	ekg-raw:
rdfs:seeAlso	https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.std.html#pandas.DataFrame.std

18.3.25 Most common value

Label: Most common value

Identifier: `ekg-raw:top`

Records the most frequently occurring value observed in a captured column.

IRI: `https://ekgf.org/ontology/raw#top`

RDF predicate	Value
http://purl.org/dc/elements/1.1/creator	http://orcid.org/0000-0003-2309-5068
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-raw:Column
rdfs:isDefinedBy	ekg-raw:
rdfs:seeAlso	https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.describe.html

18.3.26 Number of unique values

Label: Number of unique values

Identifier: `ekg-raw:unique`

Records the number of distinct values observed in a captured column.

IRI: `https://ekgf.org/ontology/raw#unique`

RDF predicate	Value
http://purl.org/dc/elements/1.1/creator	http://orcid.org/0000-0003-2309-5068
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-raw:Column
rdfs:isDefinedBy	ekg-raw:

RDF predicate	Value
rdfs:seeAlso	https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.describe.html

18.3.27 Value of

Label: Value of

Identifier: `ekg-raw:valueOf`

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.

IRI: `https://ekgf.org/ontology/raw#valueOf`

RDF predicate	Value
http://purl.org/dc/elements/1.1/creator	http://orcid.org/0000-0003-2309-5068
http://purl.org/dc/elements/1.1/date	2020-06-13T09:17:59Z
rdf:type	owl:FunctionalProperty
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-raw:Value
rdfs:isDefinedBy	ekg-raw:
rdfs:range	ekg-raw:Row

18.3.28 View Row Number

Label: View Row Number

Identifier: `ekg-raw:viewRowNumber`

The 0-based sequence number of the Row in the given View as it was encountered in the Source

IRI: `https://ekgf.org/ontology/raw#viewRowNumber`

RDF predicate	Value
http://purl.org/dc/elements/1.1/creator	http://orcid.org/0000-0003-2309-5068
http://purl.org/dc/elements/1.1/date	2020-06-13T09:41:42Z
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-raw:Row
rdfs:isDefinedBy	ekg-raw:
rdfs:range	xsd:integer

18.4 Named Individuals

18.4.1 Raw Data Validation Shapes

Label: Raw Data Validation Shapes

Identifier: `ekg-raw:ValidationShapeSet`

Instance of: ekg-shape:ShapeSet

Validates captured content, explicit source nulls, optional source text, row and column membership, and agreement on the containing view. View-to-source ownership and coordinate constraints require separate validation.

IRI: `https://ekgf.org/ontology/raw#ValidationShapeSet`

RDF predicate	Value
rdf:type	owl:NamedIndividual

RDF predicate	Value
rdf:type	ekg-shape:ShapeSet
rdfs:isDefinedBy	ekg-raw:
ekg-shape:hasShape	ekg-raw:ColumnLocationShape
ekg-shape:hasShape	ekg-raw:RowLocationShape
ekg-shape:hasShape	ekg-raw:ValueContentShape
ekg-shape:hasShape	ekg-raw:ValueLocationShape
ekg-shape:hasShapesGraph	ekg-raw:
ekg-shape:shapePurpose	ekg-shape:Validation

18.5 Shapes

18.5.1 Column Location Shape

Label: Column Location Shape

Identifier: `ekg-raw:ColumnLocationShape`

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.

IRI: `https://ekgf.org/ontology/raw#ColumnLocationShape`

RDF predicate	Value
rdf:type	sh:NodeShape
rdfs:isDefinedBy	ekg-raw:
sh:nodeKind	sh:BlankNodeOrIRI
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

18.5.2 Row Location Shape

Label: Row Location Shape

Identifier: `ekg-raw:RowLocationShape`

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.

IRI: `https://ekgf.org/ontology/raw#RowLocationShape`

RDF predicate	Value
rdf:type	sh:NodeShape
rdfs:isDefinedBy	ekg-raw:
sh:nodeKind	sh:BlankNodeOrIRI

RDF predicate	Value
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

18.5.3 Value Content Shape

Label: Value Content Shape

Identifier: `ekg-raw:ValueContentShape`

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.

IRI: `https://ekgf.org/ontology/raw#ValueContentShape`

RDF predicate	Value
rdf:type	sh:NodeShape
rdfs:isDefinedBy	ekg-raw:
sh:message	A captured value must have exactly one literalValue or be explicitly typed NullValue 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)))

18.5.4 Value Location Shape

Label: Value Location Shape

Identifier: `ekg-raw:ValueLocationShape`

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.

IRI: `https://ekgf.org/ontology/raw#ValueLocationShape`

RDF predicate	Value
`rdf:type`	`sh:NodeShape`
`rdfs:isDefinedBy`	ekg-raw:
`sh:nodeKind`	`sh:BlankNodeOrIRI`
`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: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:sparql`	(`sh:message`: A captured value's row and column must belong to the same view.; `sh:select`: 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) })
`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

19 Specification by Example Ontology

19.1 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.

19.2 Classes

19.2.1 Concept Value

Label: Concept Value

Identifier: `ekg-sbe:ConceptValue`

This is the value for a test concept

IRI: `https://ekgf.org/ontology/specification-by-example#ConceptValue`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-sbe:

19.2.2 Graph

Label: Graph

Identifier: `ekg-sbe:Graph`

A Graph in a Scenario State, every Scenario State (either the Given-state or the Then-state) can have one or more Graphs.

IRI: `https://ekgf.org/ontology/specification-by-example#Graph`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-sbe:

19.2.3 Scenario

Label: Scenario

Identifier: `ekg-sbe:Scenario`

An executable example specifying the request, initial state, and expected state or results for a story. Scenarios can be chained through follows.

IRI: `https://ekgf.org/ontology/specification-by-example#Scenario`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-sbe:

19.2.4 State

Label: State

Identifier: `ekg-sbe:State`

A set of graph and dataset conditions used to describe an initial or expected situation in a scenario.

IRI: `https://ekgf.org/ontology/specification-by-example#State`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-sbe:

19.2.5 Story Request

Label: Story Request

Identifier: `ekg-sbe:StoryRequest`

The inputs to the story that is being tested in the given Scenario

IRI: `https://ekgf.org/ontology/specification-by-example#StoryRequest`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-sbe:

19.2.6 capture

Label: capture

Identifier: `ekg-sbe: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.

IRI: `https://ekgf.org/ontology/specification-by-example#capture`

RDF predicate	Value
`rdf:type`	`rdfs:Datatype`
`rdfs:isDefinedBy`	ekg-sbe:

19.3 Properties

19.3.1 Concept Value

Label: Concept Value

Identifier: `ekg-sbe:conceptValue`

Relates a story request in a scenario to a value supplied for one of the concepts involved in that request.

IRI: `https://ekgf.org/ontology/specification-by-example#conceptValue`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-sbe:StoryRequest
`rdfs:isDefinedBy`	ekg-sbe:
`rdfs:range`	ekg-sbe:ConceptValue

19.3.2 Dataset

Label: Dataset

Identifier: `ekg-sbe:dataset`

Identifies a dataset on which a scenario state depends; it is the inverse of isDependedOnInState.

IRI: `https://ekgf.org/ontology/specification-by-example#dataset`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-sbe:State
`rdfs:isDefinedBy`	ekg-sbe:
`rdfs:range`	ekg-dataset:Dataset
`owl:inverseOf`	ekg-sbe:isDependedOnInState

19.3.3 follows

Label: follows

Identifier: `ekg-sbe: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.

IRI: `https://ekgf.org/ontology/specification-by-example#follows`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-sbe:Scenario

RDF predicate	Value
rdfs:isDefinedBy	ekg-sbe:
rdfs:range	ekg-sbe:Scenario

19.3.4 Given

Label: Given

Identifier: `ekg-sbe:given`

Relates a scenario to its initial state, before the story request is executed.

IRI: `https://ekgf.org/ontology/specification-by-example#given`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-sbe:Scenario
rdfs:isDefinedBy	ekg-sbe:
rdfs:range	ekg-sbe:State
owl:inverseOf	ekg-sbe:isGivenStateOf

19.3.5 has Dataset

Label: has Dataset

Identifier: `ekg-sbe:hasDataset`

a State can have one or more Graphs where each Graph can have one or more Datasets

IRI: `https://ekgf.org/ontology/specification-by-example#hasDataset`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-sbe:Graph
rdfs:isDefinedBy	ekg-sbe:
rdfs:range	ekg-dataset:Dataset
owl:inverseOf	ekg-sbe:isUsedInGraph

19.3.6 has Graph

Label: has Graph

Identifier: `ekg-sbe:hasGraph`

Any Scenario State can have one or more Graphs

IRI: `https://ekgf.org/ontology/specification-by-example#hasGraph`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-sbe:State
rdfs:isDefinedBy	ekg-sbe:
rdfs:range	ekg-sbe:Graph
owl:inverseOf	ekg-sbe:isGraphInState

19.3.7 Has Scenario

Label: Has Scenario

Identifier: `ekg-sbe:hasScenario`

Relates a story to an example scenario that validates its behavior; it is the inverse of validates.

IRI: `https://ekgf.org/ontology/specification-by-example#hasScenario`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-story:Story
`rdfs:isDefinedBy`	ekg-sbe:
`rdfs:range`	ekg-sbe:Scenario
`owl:inverseOf`	ekg-sbe:validates

19.3.8 ignore

Label: ignore

Identifier: `ekg-sbe: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.

IRI: `https://ekgf.org/ontology/specification-by-example#ignore`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-sbe:Scenario
`rdfs:isDefinedBy`	ekg-sbe:
`rdfs:range`	`xsd:boolean`

19.3.9 is depended on in state

Label: is depended on in state

Identifier: `ekg-sbe:isDependedOnInState`

Usually inferred from the inverse `sbe:dataset`

IRI: `https://ekgf.org/ontology/specification-by-example#isDependedOnInState`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-dataset:Dataset
`rdfs:isDefinedBy`	ekg-sbe:
`rdfs:range`	ekg-sbe:State

19.3.10 Is Given State Of

Label: Is Given State Of

Identifier: `ekg-sbe:isGivenStateOf`

Usually inferred from inverse `sbe:given`

IRI: `https://ekgf.org/ontology/specification-by-example#isGivenStateOf`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-sbe:State
`rdfs:isDefinedBy`	ekg-sbe:
`rdfs:range`	ekg-sbe:Scenario

19.3.11 is Graph in State

Label: is Graph in State

Identifier: `ekg-sbe:isGraphInState`
Usually inferred, a Graph can be part of one or more States
IRI: `https://ekgf.org/ontology/specification-by-example#isGraphInState`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-sbe:Graph
rdfs:isDefinedBy	ekg-sbe:
rdfs:range	ekg-sbe:State

19.3.12 Is Used In Graph

Label: Is Used In Graph
Identifier: `ekg-sbe:isUsedInGraph`
Relates a dataset to a scenario graph that uses it; it is the inverse of hasDataset.
IRI: `https://ekgf.org/ontology/specification-by-example#isUsedInGraph`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-dataset:Dataset
rdfs:isDefinedBy	ekg-sbe:
rdfs:range	ekg-sbe:Graph

19.3.13 Named Graph

Label: Named Graph
Identifier: `ekg-sbe:namedGraph`
Records the IRI naming a graph in a scenario state as an xsd:anyURI literal.
IRI: `https://ekgf.org/ontology/specification-by-example#namedGraph`

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-sbe:Graph
rdfs:isDefinedBy	ekg-sbe:
rdfs:range	xsd:anyURI

19.3.14 Story Parameters For Scenario

Label: Story Parameters For Scenario
Identifier: `ekg-sbe:storyParametersForScenario`
Usually inferred from inverse `sbe:when`
IRI: `https://ekgf.org/ontology/specification-by-example#storyParametersForScenario`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-sbe:State
rdfs:isDefinedBy	ekg-sbe:
rdfs:range	ekg-sbe:Scenario
owl:inverseOf	ekg-sbe:when

19.3.15 Then

Label: Then

Identifier: `ekg-sbe:then`

Relates a scenario to the state or results expected after its story request is executed.

IRI: `https://ekgf.org/ontology/specification-by-example#then`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-sbe:Scenario
`rdfs:isDefinedBy`	ekg-sbe:
`rdfs:range`	ekg-sbe:State

19.3.16 Validates

Label: Validates

Identifier: `ekg-sbe:validates`

Identifies the story whose behavior a scenario validates. The functional-property axiom assigns at most one story to a scenario.

IRI: `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:isDefinedBy`	ekg-sbe:
`rdfs:range`	ekg-story:Story

19.3.17 Value

Label: Value

Identifier: `ekg-sbe:value`

Contains the literal value represented by a scenario concept-value resource. No literal datatype restriction is declared.

IRI: `https://ekgf.org/ontology/specification-by-example#value`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-sbe:ConceptValue
`rdfs:isDefinedBy`	ekg-sbe:

19.3.18 When

Label: When

Identifier: `ekg-sbe:when`

Relates a scenario to the story request that is executed between the initial and expected states.

IRI: `https://ekgf.org/ontology/specification-by-example#when`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-sbe:Scenario

RDF predicate	Value
`rdfs:isDefinedBy`	ekg-sbe:
`rdfs:range`	ekg-sbe:StoryRequest

19.4 Named Individuals

19.4.1 any

Label: any

Identifier: `ekg-sbe: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.

IRI: `https://ekgf.org/ontology/specification-by-example#any`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdfs:isDefinedBy`	ekg-sbe:

20 SHACL Shape Metadata Ontology

20.1 Overview

SHACL Shape Metadata Ontology

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

20.1.1 Shapesets and purpose

A `ShapeSet` groups SHACL shape definitions. Its declared entry shapes can be node shapes or property shapes, named or anonymous. Their nested definitions are part of the set as well: follow `sh:property`, `sh:node`, `sh:not`, `sh:qualifiedValueShape`, and the members of `sh:and`, `sh:or`, and `sh:xone` lists. This includes inline property shapes and empty logical branches without assigning them artificial global identities.

Record entry membership using `hasShape` on the set or `inShapeSet` on the shape. These are annotation properties. Reading membership counts both directions; no OWL inverse inference is required. A reusable shape can belong to several sets, and a nested shape can be reached from several entries.

20.1.1.1 Validation and purpose values

Each shapeset has an IRI, a label, at least one entry shape, and at least one purpose. A purpose is a labeled IRI explicitly typed as `ShapePurpose`. `Validation` is the purpose used by this repository. Applications can define additional purpose individuals. Purposes can also be attached directly to individual shapes.

The included shapes validate membership endpoints, purpose values, and coverage of the supplied shape definitions. Every shape must be an entry or reachable from an entry. Undefined members, unregistered shapesets, empty sets, missing purposes, and orphan shapes are errors. Shapes can be identified through their types, targets, paths, SHACL Core constraints, SHACL-SPARQL constraints, and shape-reference parameters.

This validates shapeset metadata. SHACL meta-validation supplies the complete SHACL syntax checks; ordinary data validation executes the selected constraints. Shapeset membership itself does not change which targets a SHACL engine runs.

20.1.1.2 Concrete validation graphs

`hasShapesGraph` links a shapeset to an explicitly identified `sh:ShapesGraph`. The set records collection membership; the graph identifies an actual RDF validation input. Their identities are separate, and their classes are disjoint. A set may reference several graphs, and one graph may serve several sets. No graph is selected by default. A set may also exist without a graph link.

Each linked graph must be an IRI with the Shapes Graph role. The caller selects the intended graph, resolves it, checks that the required shape definitions and dependencies are present, and supplies that concrete graph to the validator. The link does not fetch data, copy definitions, enact imports, or prove conformance. Metadata validation alone cannot verify the graph's availability or contents.

The graph role uses `sh:ShapesGraph` from the 25 October 2026 SHACL 1.2 Core Working Draft. This is a referenced external term, not a class owned by this ontology. Our metadata constraints use SHACL Core features already supported by the repository's pinned validator; this reference does not claim full SHACL 1.2 implementation support.

Use-case parentage and reuse do not imply graph selection, profile specialization, inherited constraints, conformance, or warnings. Child use cases may choose stricter, looser, or different constraints. Inheritance policy would require a separate explicit decision.

20.1.1.3 Using our own metadata

Every ontology containing shapes declares a `ValidationShapeSet` that lists all its named entry shapes and assigns the `Validation` purpose. Inline definitions are included through their entries. This ontology groups its own metadata validators in exactly the same way.

Each repository validation set links to its module's ontology IRI through `hasShapesGraph`, and the module identifies that IRI as `sh:ShapesGraph`. Load the corresponding local file as the graph under that IRI. The ontology IRI identifies the module graph; its distinct `ValidationShapeSet` IRI identifies the collection. This explicit binding also applies to the validators in this metadata ontology.

Validate a module's metadata using this ontology as the shapes graph and as the ontology graph supplying the purpose definitions:

```
pyshacl -s ekgf-shape.ttl -e ekgf-shape.ttl ekgf-lifecycle.ttl
```

The repository test suite also validates the combined ontology sources and checks entry and nested-shape coverage independently.

20.1.1.4 Migration

`hasShape` replaces the node-only `hasNodeShape` predicate. Migrate those assertions; `LegacyShapeMembershipShape` rejects the old predicate. `inShapeSet` now covers both node and property shapes. The ontology IRI, namespace, preferred prefix, and registered artifact identity are retained.

20.2 Classes

20.2.1 Shape or shapeset

Label: Shape or shapeset

Identifier: `ekg-shape:ShapeOrShapeSet`

A SHACL node shape, property shape, or shapeset. This named union supplies the domain of the shapePurpose annotation property.

IRI: `https://ekgf.org/ontology/shape#ShapeOrShapeSet`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-shape:
`owl:equivalentClass`	(`rdf:type`: `owl:Class`; `owl:unionOf`: (`sh:Shape`, ekg-shape:ShapeSet))

20.2.2 Shape purpose

Label: Shape purpose

Identifier: `ekg-shape:ShapePurpose`

What a SHACL shape or shapeset is for. Purposes are extensible controlled values, each a labeled individual of this class.

IRI: `https://ekgf.org/ontology/shape#ShapePurpose`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-shape:

20.2.3 Shapeset

Label: Shapeset

Identifier: `ekg-shape:ShapeSet`

20.2.3.1 Collection and concrete graph

A ShapeSet groups entry shapes and the nested definitions they reference. Its membership and purpose describe a collection independently of how those definitions are packaged for validation. `hasShapesGraph` identifies concrete RDF graph inputs associated with the collection. A set and a shapes graph have separate identities; their classes are disjoint.

A set can reference zero or more graphs, and one graph can serve multiple sets. For an actual validation run, select the intended graph explicitly. Resolve its IRI, load the required definitions and dependencies, and validate using that fixed graph. Do not select an arbitrary graph from several links or treat an unresolved reference as an empty validation graph.

For example, `ex:personShapeSet` can link to `ex:personShapesGraph`; the former records membership and the latter identifies the RDF input given to the validator. The graph may contain additional sets, shapes, and supporting data, so passing it to a validator can run more than the one set's entry shapes. Collection membership is not an execution filter.

Use-case parentage and reuse imply no profile inheritance or graph selection. Child use cases may select stricter, looser, or different constraints. No conformance statement or positional warning is inferred from the use-case tree. Any future inheritance policy requires a separate explicit decision.

IRI: `https://ekgf.org/ontology/shape#ShapeSet`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-shape:
`owl:disjointWith`	`sh:ShapesGraph`

20.3 Properties

20.3.1 Has shape

Label: Has shape

Identifier: `ekg-shape:hasShape`

Registers an entry shape in a shapeset. Both node and property shapes are supported; their nested definitions are included through SHACL shape-reference paths. As an annotation, it can describe shapes also used as OWL classes.

IRI: `https://ekgf.org/ontology/shape#hasShape`

RDF predicate	Value
`rdf:type`	`owl:AnnotationProperty`

RDF predicate	Value
`rdfs:domain`	ekg-shape:ShapeSet
`rdfs:isDefinedBy`	ekg-shape:
`rdfs:range`	`sh:Shape`

20.3.2 Has shapes graph

Label: Has shapes graph

Identifier: `ekg-shape:hasShapesGraph`

20.3.2.1 Explicit validation input

Links a ShapeSet to an IRI identifying a concrete shapes graph containing definitions for validation with that set. Its domain is ShapeSet and its range is the external `sh:ShapesGraph` role. The property is neither functional nor inverse-functional: several graph references and shared graphs are allowed. It does not equate a set with a graph or materialize a default graph.

```
@prefix shape: <https://ekgf.org/ontology/shape#> .
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix ex: <https://example.test/> .

ex:personShapeSet shape:hasShapesGraph ex:personShapesGraph .
ex:personShapesGraph a sh:ShapesGraph .
```

This fragment illustrates the link. A complete set also needs its type, label, purpose, and entry shapes. The referenced graph must be resolved and loaded explicitly; typing its IRI does not prove availability, completeness, or a successful validation. The application checks that required definitions and dependencies are supplied and records the actual graph used in a run.

This is not `sh:shapesGraph`, the SHACL data-to-validation-graph link. It does not generate that property, assert conformance, load imports, select targets, or introduce use-case profile inheritance. These behaviors require their own explicit declarations and processing contracts.

The graph role is referenced from the SHACL 1.2 Core Working Draft dated 25 October 2026. The reference is metadata vocabulary use, not an assertion that the execution engine implements every feature of that draft.

See the shapes-graph guide for a complete example, validation, and graph selection.

IRI: `https://ekgf.org/ontology/shape#hasShapesGraph`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-shape:ShapeSet
`rdfs:isDefinedBy`	ekg-shape:
`rdfs:range`	`sh:ShapesGraph`

20.3.3 In shapeset

Label: In shapeset

Identifier: `ekg-shape:inShapeSet`

Registers a SHACL node or property shape as an entry in a shapeset. Membership can instead be asserted with hasShape in the other direction; readers count both directions explicitly.

IRI: `https://ekgf.org/ontology/shape#inShapeSet`

RDF predicate	Value
`rdf:type`	`owl:AnnotationProperty`

RDF predicate	Value
rdfs:domain	sh:Shape
rdfs:isDefinedBy	ekg-shape:
rdfs:range	ekg-shape:ShapeSet

20.3.4 Shape purpose

Label: Shape purpose

Identifier: `ekg-shape:shapePurpose`

Relates a SHACL shape or shapeset to a purpose individual. Purposes describe the intended use of constraints and do not execute them.

IRI: `https://ekgf.org/ontology/shape#shapePurpose`

RDF predicate	Value
rdf:type	owl:AnnotationProperty
rdfs:domain	ekg-shape:ShapeOrShapeSet
rdfs:isDefinedBy	ekg-shape:
rdfs:range	ekg-shape:ShapePurpose

20.4 Named Individuals

20.4.1 Validation

Label: Validation

Identifier: `ekg-shape:Validation`

Instance of: ekg-shape:ShapePurpose

The purpose of checking supplied RDF data or metadata against declared constraints.

IRI: `https://ekgf.org/ontology/shape#Validation`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-shape:ShapePurpose
rdfs:isDefinedBy	ekg-shape:

20.4.2 SHACL Shape Metadata Validation Shapes

Label: SHACL Shape Metadata Validation Shapes

Identifier: `ekg-shape:ValidationShapeSet`

Instance of: ekg-shape:ShapeSet

Validates shapeset membership, purpose annotations, explicit shapes-graph links, and complete coverage of shape definitions, including this ontology's own validators.

IRI: `https://ekgf.org/ontology/shape#ValidationShapeSet`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-shape:ShapeSet
rdfs:isDefinedBy	ekg-shape:
ekg-shape:hasShape	ekg-shape:LegacyShapeMembershipShape
ekg-shape:hasShape	ekg-shape:ShapeDefinitionShape
ekg-shape:hasShape	ekg-shape:ShapePurposeShape

RDF predicate	Value
ekg-shape:hasShape	ekg-shape:ShapePurposeTargetShape
ekg-shape:hasShape	ekg-shape:ShapeSetCoverageShape
ekg-shape:hasShape	ekg-shape:ShapeSetShape
ekg-shape:hasShape	ekg-shape:ShapesGraphLinkShape
ekg-shape:hasShapesGraph	ekg-shape:
ekg-shape:shapePurpose	ekg-shape:Validation

20.5 Shapes

20.5.1 Legacy Shape Membership Shape

Label: Legacy Shape Membership Shape

Identifier: `ekg-shape:LegacyShapeMembershipShape`

Rejects the retired node-only hasNodeShape predicate. Generic membership uses hasShape and inShapeSet.

IRI: `https://ekgf.org/ontology/shape#LegacyShapeMembershipShape`

RDF predicate	Value
rdf:type	sh:NodeShape
rdfs:isDefinedBy	ekg-shape:
sh:property	(sh:maxCount: 0; sh:message: hasNodeShape was removed. Use hasShape or inShapeSet.; sh:path: ekg-shape:hasNodeShape)
sh:targetSubjectsOf	ekg-shape:hasNodeShape

20.5.2 Shape Definition Shape

Label: Shape Definition Shape

Identifier: `ekg-shape:ShapeDefinitionShape`

Checks that membership and purpose targets identify supplied SHACL shape definitions, including implicit and nested shapes. Complete SHACL syntax validation is a separate meta-validation step.

IRI: `https://ekgf.org/ontology/shape#ShapeDefinitionShape`

RDF predicate	Value
rdf:type	sh:NodeShape
rdfs:isDefinedBy	ekg-shape:
sh:message	A shapeset member must be a supplied SHACL shape definition.
sh:nodeKind	sh:BlankNodeOrIRI
sh:or	((sh:class: sh:NodeShape), (sh:class: sh:PropertyShape), (sh:property: (sh:minCount: 1; sh:path: (sh:alternativePath: (sh:path, sh:targetClass, sh:targetNode, sh:targetSubjectsOf, sh:targetObjectsOf, sh:and, sh:class, sh:closed, sh:datatype, sh:disjoint, sh:equals, sh:flags, sh:hasValue, sh:ignoredProperties, sh:in, sh:languageIn, sh:lessThan, sh:lessThanOrEquals, sh:maxCount, sh:maxExclusive, sh:maxInclusive, sh:maxLength, sh:minCount, sh:minExclusive, sh:minInclusive, sh:minLength, sh:node, sh:nodeKind, sh:not, sh:or, sh:pattern,

RDF predicate	Value
	sh:property, sh:qualifiedMaxCount, sh:qualifiedMinCount, sh:qualifiedValueShape, sh:qualifiedValueShapesDisjoint, sh:uniqueLang, sh:xone, sh:sparql)))), (sh:property: (sh:minCount: 1; sh:path: (sh:inversePath: (sh:alternativePath: (sh:property, sh:node, sh:not, sh:qualifiedValueShape))))), (sh:property: (sh:minCount: 1; sh:path: (sh:inversePath: ((sh:alternativePath: (sh:and, sh:or, sh:xone)), (sh:zeroOrMorePath: rdf:rest), rdf:first)))))
sh:property	(sh:maxCount: 1; sh:nodeKind: sh:BlankNodeOrIRI; sh:path: sh:path)
sh:targetObjectsOf	ekg-shape:hasShape
sh:targetSubjectsOf	ekg-shape:inShapeSet

20.5.3 Shape Purpose Shape

Label: Shape Purpose Shape

Identifier: `ekg-shape:ShapePurposeShape`

Requires each purpose value to be an explicitly typed, labeled ShapePurpose IRI while permitting additional purpose individuals.

IRI: `https://ekgf.org/ontology/shape#ShapePurposeShape`

RDF predicate	Value
rdf:type	sh:NodeShape
rdfs:isDefinedBy	ekg-shape:
sh:class	ekg-shape:ShapePurpose
sh:nodeKind	sh:IRI
sh:property	(sh:minCount: 1; sh:minLength: 1; sh:or: ((sh:datatype: xsd:string), (sh:datatype: rdf:langString)); sh:path: rdfs:label)
sh:targetClass	ekg-shape:ShapePurpose
sh:targetObjectsOf	ekg-shape:shapePurpose

20.5.4 Shape Purpose Target Shape

Label: Shape Purpose Target Shape

Identifier: `ekg-shape:ShapePurposeTargetShape`

Restricts purpose annotations to shapesets and supplied SHACL shape definitions.

IRI: `https://ekgf.org/ontology/shape#ShapePurposeTargetShape`

RDF predicate	Value
rdf:type	sh:NodeShape
rdfs:isDefinedBy	ekg-shape:
sh:or	((sh:class: ekg-shape:ShapeSet), (sh:node: ekg-shape:ShapeDefinitionShape))
sh:targetSubjectsOf	ekg-shape:shapePurpose

20.5.5 Shapeset Coverage Shape

Label: Shapeset Coverage Shape

Identifier: `ekg-shape:ShapeSetCoverageShape`

Rejects orphan shape definitions anywhere in the supplied graph. Coverage includes both membership directions and all nested SHACL Core shape-reference paths.

IRI: `https://ekgf.org/ontology/shape#ShapeSetCoverageShape`

RDF predicate	Value
`rdf:type`	`sh:NodeShape`
`rdfs:isDefinedBy`	ekg-shape:
`sh:sparql`	(`sh:message`: Shape {?value} is outside every declared shapeset.; `sh:select`: PREFIX shape: <https://ekgf.org/ontology/shape#> PREFIX sh: <http://www.w3.org/ns/shacl#> PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> SELECT DISTINCT $this (?candidate AS ?value) WHERE { { ?candidate rdf:type/rdfs:subClassOf* ?type . FILTER (?type IN (sh:NodeShape, sh:PropertyShape)) } UNION { ?candidate ?parameter ?argument . FILTER (?parameter IN (sh:path, sh:targetClass, sh:targetNode, sh:targetSubjectsOf, sh:targetObjectsOf, sh:and, sh:class, sh:closed, sh:datatype, sh:disjoint, sh:equals, sh:flags, sh:hasValue, sh:ignoredProperties, sh:in, sh:languageIn, sh:lessThan, sh:lessThanOrEquals, sh:maxCount, sh:maxExclusive, sh:maxInclusive, sh:maxLength, sh:minCount, sh:minExclusive, sh:minInclusive, sh:minLength, sh:node, sh:nodeKind, sh:not, sh:or, sh:pattern, sh:property, sh:qualifiedMaxCount, sh:qualifiedMinCount, sh:qualifiedValueShape, sh:qualifiedValueShapesDisjoint, sh:uniqueLang, sh:xone, sh:sparql)) } UNION { ?parent (sh:property|sh:node|sh:not|sh:qualifiedValueShape) ?candidate . } UNION { ?parent (sh:and|sh:or|sh:xone)/rdf:rest*/rdf:first ?candidate . } FILTER NOT EXISTS { ?set a shape:ShapeSet . { ?set shape:hasShape ?entry . } UNION { ?entry shape:inShapeSet ?set . } ?entry (sh:property|sh:node|sh:not|sh:qualifiedValueShape| (sh:and|sh:or|sh:xone)/rdf:rest*/rdf:first)* ?candidate . } })
`sh:targetNode`	ekg-shape:ShapeSetCoverageShape

20.5.6 Shapeset Shape

Label: Shapeset Shape

Identifier: `ekg-shape:ShapeSetShape`

Validates a labeled shapeset IRI distinct from a shapes graph, at least one defined entry shape, and at least one typed purpose. Membership is counted in both directions and can be shared across sets. Graph-link subjects must satisfy the same collection metadata requirements.

IRI: `https://ekgf.org/ontology/shape#ShapeSetShape`

RDF predicate	Value
`rdf:type`	`sh:NodeShape`

RDF predicate	Value
rdfs:isDefinedBy	ekg-shape:
sh:class	ekg-shape:ShapeSet
sh:nodeKind	sh:IRI
sh:not	(sh:class: sh:ShapesGraph)
sh:property	(sh:class: ekg-shape:ShapePurpose; sh:minCount: 1; sh:nodeKind: sh:IRI; sh:path: ekg-shape:shapePurpose)
sh:property	(sh:minCount: 1; sh:minLength: 1; sh:or: ((sh:datatype: xsd:string), (sh:datatype: rdf:langString)); sh:path: rdfs:label)
sh:property	(sh:minCount: 1; sh:node: ekg-shape:ShapeDefinitionShape; sh:path: (sh:alternativePath: (ekg-shape:hasShape, (sh:inversePath: ekg-shape:inShapeSet))))
sh:targetClass	ekg-shape:ShapeSet
sh:targetObjectsOf	ekg-shape:inShapeSet
sh:targetSubjectsOf	ekg-shape:hasShape
sh:targetSubjectsOf	ekg-shape:hasShapesGraph

20.5.7 Shapes Graph Link Shape

Label: Shapes Graph Link Shape

Identifier: `ekg-shape:ShapesGraphLinkShape`

20.5.7.1 What this validator establishes

The subject of `hasShapesGraph` must identify a ShapeSet. Each object must be a ShapesGraph IRI and must not also be a ShapeSet. `ShapeSetShape` checks the collection's label, purpose, and entry definitions. Both validators are registered in this ontology's ValidationShapeSet.

The documented metadata-validation command uses no entailment, so endpoint types are supplied explicitly. RDFS or OWL reasoning can infer these types from the link's domain and range, but inferred typing is still not evidence that a graph has been retrieved or used successfully.

This validator does not dereference IRIs, compare named-graph contents against collection membership, check import closure, or validate application data. The repository tests load every linked local module as its named graph and exercise real data validation with the executable example's concrete graph. An application must reject unavailable or incomplete graph inputs before treating a validation result as conformance evidence.

IRI: `https://ekgf.org/ontology/shape#ShapesGraphLinkShape`

RDF predicate	Value
rdf:type	sh:NodeShape
rdfs:isDefinedBy	ekg-shape:
sh:class	ekg-shape:ShapeSet
sh:nodeKind	sh:IRI
sh:property	(sh:class: sh:ShapesGraph; sh:message: A shapeset must link to a ShapesGraph IRI with a separate identity; a collection cannot itself be the concrete graph.; sh:nodeKind: sh:IRI; sh:not: (sh:class: ekg-shape:ShapeSet); sh:path: ekg-shape:hasShapesGraph)

RDF predicate	Value
`sh:targetSubjectsOf`	ekg-shape:hasShapesGraph

21 Story Ontology

21.1 Overview

Story Ontology

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

21.1.1 Stories and concept usage

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.

21.1.1.1 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.

21.1.1.1.1 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.

21.1.1.2 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.

21.1.1.3 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.

21.1.1.4 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.

EKGF Editor's Draft

`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.

21.1.1.5 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 `ValidationShapeSet`; 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.

21.2 Classes

21.2.1 Agent Job Implementation

Label: Agent Job Implementation

Identifier: `ekg-story:AgentJobImplementation`

A story implementation expressed as a job performed by an agent.

IRI: `https://ekgf.org/ontology/story#AgentJobImplementation`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-story:
`rdfs:subClassOf`	ekg-story:StoryImplementation

21.2.2 API Proxy Implementation

Label: API Proxy Implementation

Identifier: `ekg-story:ApiProxyImplementation`

A story implementation that delegates the interaction to an API through a proxy.

IRI: `https://ekgf.org/ontology/story#ApiProxyImplementation`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-story:
rdfs:subClassOf	ekg-story:StoryImplementation

21.2.3 Named Statement Implementation

Label: Named Statement Implementation

Identifier: `ekg-story:NamedStatementImplementation`

A story implementation identified through a named executable statement.

IRI: `https://ekgf.org/ontology/story#NamedStatementImplementation`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-story:
rdfs:subClassOf	ekg-story:StoryImplementation

21.2.4 Story

Label: Story

Identifier: `ekg-story:Story`

A description of an interaction in a use case, identifying its participating persona, inputs, outputs, implementations, and the outcomes it helps achieve.

IRI: `https://ekgf.org/ontology/story#Story`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-story:
owl:disjointWith	ekg-story:StoryImplementation

21.2.5 Story Concept Usage

Label: Story Concept Usage

Identifier: `ekg-story:StoryConceptUsage`

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.

IRI: `https://ekgf.org/ontology/story#StoryConceptUsage`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-story:
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: (owl:inverseOf: ekg-story:hasConceptUsage))
rdfs:subClassOf	(rdf:type: owl:Restriction; owl:maxCardinality: 1; owl:onProperty: ekg-concept:hasConcept)
rdfs:subClassOf	(rdf:type: owl:Restriction; owl:minCardinality: 1; owl:onProperty: (owl:inverseOf: ekg-story:hasConceptUsage))

RDF predicate	Value
owl:disjointWith	ekg-concept:ConceptManifestation
owl:disjointWith	ekg-concept:DataElementConcept
owl:disjointWith	ekg-story:Story

21.2.6 Story Dependent

Label: Story Dependent

Identifier: `ekg-story:StoryDependent`

21.2.6.1 A story's dependent-concept usage

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.

IRI: `https://ekgf.org/ontology/story#StoryDependent`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-story:
rdfs:subClassOf	ekg-story:StoryConceptUsage

21.2.7 Story Implementation

Label: Story Implementation

Identifier: `ekg-story:StoryImplementation`

21.2.7.1 One owning story

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.

IRI: `https://ekgf.org/ontology/story#StoryImplementation`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-story:
rdfs:subClassOf	(rdf:type: owl:Restriction; owl:cardinality: 1; owl:onProperty: ekg-story:implementationOf)

21.2.8 Story Input

Label: Story Input

Identifier: `ekg-story:StoryInput`

21.2.8.1 A story's input-concept usage

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.

IRI: `https://ekgf.org/ontology/story#StoryInput`

RDF predicate	Value
`http://purl.org/dc/elements/1.1/creator`	`http://orcid.org/0000-0003-2309-5068`
`http://purl.org/dc/elements/1.1/date`	2020-04-01T17:24:36Z
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-story:
`rdfs:subClassOf`	ekg-story:StoryConceptUsage
`owl:disjointWith`	ekg-story:StoryDependent
`owl:disjointWith`	ekg-story:StoryOutput

21.2.9 Story Output

Label: Story Output

Identifier: `ekg-story:StoryOutput`

21.2.9.1 A story's output-concept usage

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.

IRI: `https://ekgf.org/ontology/story#StoryOutput`

RDF predicate	Value
`http://purl.org/dc/elements/1.1/creator`	`http://orcid.org/0000-0003-2309-5068`
`http://purl.org/dc/elements/1.1/date`	2020-04-07T20:40:12Z
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-story:
`rdfs:subClassOf`	ekg-story:StoryConceptUsage
`owl:disjointWith`	ekg-story:StoryDependent

21.2.10 Story Stereotype

Label: Story Stereotype

Identifier: `ekg-story:StoryStereoType`

21.2.10.1 Primary purpose, independent of execution

`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 `ValidationShapeSet`. 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.

IRI: `https://ekgf.org/ontology/story#StoryStereoType`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-story:
`owl:disjointWith`	ekg-story:Story
`owl:disjointWith`	ekg-story:StoryConceptUsage
`owl:disjointWith`	ekg-story:StoryImplementation

21.2.11 Workflow Implementation

Label: Workflow Implementation

Identifier: `ekg-story:WorkflowImplementation`

A story implementation expressed as a workflow of execution steps.

IRI: `https://ekgf.org/ontology/story#WorkflowImplementation`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-story:
`rdfs:subClassOf`	ekg-story:StoryImplementation

21.3 Properties

21.3.1 contributes to

Label: contributes to

Identifier: `ekg-story:contributesTo`

The business outcome this story contributes to.

IRI: `https://ekgf.org/ontology/story#contributesTo`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-story:Story
`rdfs:isDefinedBy`	ekg-story:
`rdfs:range`	ekg-outcome:Outcome

21.3.2 has concept usage

Label: has concept usage

Identifier: `ekg-story:hasConceptUsage`

Links a story to a concept-usage object owned by that story. Different stories use separate usage objects when referencing the same reusable concept.

IRI: `https://ekgf.org/ontology/story#hasConceptUsage`

RDF predicate	Value
`rdf:type`	`owl:InverseFunctionalProperty`
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-story:Story
`rdfs:isDefinedBy`	ekg-story:
`rdfs:range`	ekg-story:StoryConceptUsage

21.3.3 has dependent concept

Label: has dependent concept

Identifier: `ekg-story:hasDependentConcept`

Links a story to a dependent-concept usage referencing a shared data element concept used internally rather than exposed as an input or output.

IRI: `https://ekgf.org/ontology/story#hasDependentConcept`

RDF predicate	Value
`http://purl.org/dc/elements/1.1/creator`	`http://orcid.org/0000-0003-2309-5068`
`http://purl.org/dc/elements/1.1/date`	2026-02-23T00:00:00Z
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-story:Story
`rdfs:isDefinedBy`	ekg-story:
`rdfs:range`	ekg-story:StoryDependent
`rdfs:subPropertyOf`	ekg-story:hasConceptUsage

21.3.4 has input concept

Label: has input concept

Identifier: `ekg-story:hasInputConcept`

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.

IRI: `https://ekgf.org/ontology/story#hasInputConcept`

RDF predicate	Value
`http://purl.org/dc/elements/1.1/creator`	`http://orcid.org/0000-0003-2309-5068`
`http://purl.org/dc/elements/1.1/date`	2026-02-23T00:00:00Z
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-story:Story
`rdfs:isDefinedBy`	ekg-story:
`rdfs:range`	ekg-story:StoryInput
`rdfs:subPropertyOf`	ekg-story:hasConceptUsage

21.3.5 has output concept

Label: has output concept

Identifier: `ekg-story:hasOutputConcept`

Links a story to an output-concept usage referencing the shared data element concept produced by the story.

IRI: `https://ekgf.org/ontology/story#hasOutputConcept`

RDF predicate	Value
`http://purl.org/dc/elements/1.1/creator`	`http://orcid.org/0000-0003-2309-5068`
`http://purl.org/dc/elements/1.1/date`	2026-02-23T00:00:00Z
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-story:Story
`rdfs:isDefinedBy`	ekg-story:
`rdfs:range`	ekg-story:StoryOutput
`rdfs:subPropertyOf`	ekg-story:hasConceptUsage

21.3.6 has stereotype

Label: has stereotype

Identifier: `ekg-story:hasStereoType`

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.

IRI: `https://ekgf.org/ontology/story#hasStereoType`

RDF predicate	Value
`rdf:type`	`owl:FunctionalProperty`
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-story:Story
`rdfs:isDefinedBy`	ekg-story:
`rdfs:range`	ekg-story:StoryStereoType

21.3.7 implementation of

Label: implementation of

Identifier: `ekg-story:implementationOf`

21.3.7.1 Implementation to its single story

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.

IRI: `https://ekgf.org/ontology/story#implementationOf`

RDF predicate	Value
`http://purl.org/dc/elements/1.1/creator`	`http://orcid.org/0000-0003-2309-5068`
`http://purl.org/dc/elements/1.1/date`	2026-02-23T00:00:00Z
`rdf:type`	`owl:FunctionalProperty`
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-story:StoryImplementation

RDF predicate	Value
rdfs:isDefinedBy	ekg-story:
rdfs:range	ekg-story:Story
owl:inverseOf	ekg-story:implementedBy

21.3.8 Implementation Type

Label: Implementation Type

Identifier: `ekg-story:implementationType`

Coarse implementation type. Supported values: NamedStatement, ApiProxy, Workflow, AgentJob, Pipeline, UiAction.

IRI: `https://ekgf.org/ontology/story#implementationType`

RDF predicate	Value
http://purl.org/dc/elements/1.1/creator	http://orcid.org/0000-0003-2309-5068
http://purl.org/dc/elements/1.1/date	2026-02-23T00:00:00Z
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-story:StoryImplementation
rdfs:isDefinedBy	ekg-story:
rdfs:range	xsd:token

21.3.9 implemented by

Label: implemented by

Identifier: `ekg-story:implementedBy`

21.3.9.1 Story to implementation

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.

IRI: `https://ekgf.org/ontology/story#implementedBy`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-story:Story
rdfs:isDefinedBy	ekg-story:
rdfs:range	ekg-story:StoryImplementation

21.3.10 is deletion

Label: is deletion

Identifier: `ekg-story:isDeletion`

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.

IRI: `https://ekgf.org/ontology/story#isDeletion`

RDF predicate	Value
rdf:type	owl:DatatypeProperty

RDF predicate	Value
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-story:Story
rdfs:isDefinedBy	ekg-story:
rdfs:range	xsd:boolean

21.3.11 is mandatory

Label: is mandatory

Identifier: `ekg-story:isMandatory`

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.

IRI: `https://ekgf.org/ontology/story#isMandatory`

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-story:StoryInput
rdfs:isDefinedBy	ekg-story:
rdfs:range	xsd:boolean

21.3.12 Is Plural

Label: Is Plural

Identifier: `ekg-story:isPlural`

Do not use, use isSingular

IRI: `https://ekgf.org/ontology/story#isPlural`

RDF predicate	Value
http://purl.org/dc/elements/1.1/creator	http://orcid.org/0000-0003-2309-5068
http://purl.org/dc/elements/1.1/date	2020-05-06T09:38:46Z
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-story:Story
rdfs:isDefinedBy	ekg-story:
rdfs:range	xsd:boolean
owl:propertyDisjointWith	ekg-story:isSingular

21.3.13 is server provided

Label: is server provided

Identifier: `ekg-story:isServerProvided`

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.

IRI: `https://ekgf.org/ontology/story#isServerProvided`

RDF predicate	Value
rdf:type	owl:DatatypeProperty

RDF predicate	Value
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-story:StoryInput
rdfs:isDefinedBy	ekg-story:
rdfs:range	xsd:boolean

21.3.14 Is Singular

Label: Is Singular

Identifier: `ekg-story:isSingular`

Means that the user story results in a singular Thing as its output

IRI: `https://ekgf.org/ontology/story#isSingular`

RDF predicate	Value
http://purl.org/dc/elements/1.1/creator	http://orcid.org/0000-0003-2309-5068
http://purl.org/dc/elements/1.1/date	2020-05-06T09:37:47Z
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-story:Story
rdfs:isDefinedBy	ekg-story:
rdfs:range	xsd:boolean

21.3.15 Is Update

Label: Is Update

Identifier: `ekg-story:isUpdate`

Means that the user story mutates the database by insert or delete statements

IRI: `https://ekgf.org/ontology/story#isUpdate`

RDF predicate	Value
http://purl.org/dc/elements/1.1/creator	http://orcid.org/0000-0003-2309-5068
http://purl.org/dc/elements/1.1/date	2020-05-06T09:37:47Z
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-story:Story
rdfs:isDefinedBy	ekg-story:
rdfs:range	xsd:boolean

21.3.16 not found on empty

Label: not found on empty

Identifier: `ekg-story:notFoundOnEmpty`

A singular story that returns no rows answers 404 rather than an empty object.

IRI: `https://ekgf.org/ontology/story#notFoundOnEmpty`

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty

RDF predicate	Value
rdfs:domain	ekg-story:Story
rdfs:isDefinedBy	ekg-story:
rdfs:range	xsd:boolean

21.3.17 Played By

Label: Played By

Identifier: `ekg-story:playedBy`

Identifies the persona playing the actor role in a story. The functional-property axiom limits a story to one persona through this property.

IRI: `https://ekgf.org/ontology/story#playedBy`

RDF predicate	Value
http://purl.org/dc/elements/1.1/creator	http://orcid.org/0000-0003-2309-5068
http://purl.org/dc/elements/1.1/date	2020-04-01T18:12:45Z
rdf:type	owl:FunctionalProperty
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-story:Story
rdfs:isDefinedBy	ekg-story:
rdfs:range	ekg-persona:Persona

21.3.18 provided by

Label: provided by

Identifier: `ekg-story:providedBy`

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.

IRI: `https://ekgf.org/ontology/story#providedBy`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-story:Story
rdfs:isDefinedBy	ekg-story:

21.4 Named Individuals

21.4.1 Query

Label: Query

Identifier: `ekg-story:StoryStereoTypeQuery`

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.

IRI: `https://ekgf.org/ontology/story#StoryStereoTypeQuery`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-story:StoryStereoType
rdfs:isDefinedBy	ekg-story:

RDF predicate	Value
owl:differentFrom	ekg-story:StoryStereoTypeTransformationRule
owl:differentFrom	ekg-story:StoryStereoTypeUpdate
owl:differentFrom	ekg-story:StoryStereoTypeValidationRule

21.4.2 Transformation Rule

Label: Transformation Rule

Identifier: `ekg-story:StoryStereoTypeTransformationRule`

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.

IRI: `https://ekgf.org/ontology/story#StoryStereoTypeTransformationRule`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-story:StoryStereoType
rdfs:isDefinedBy	ekg-story:
owl:differentFrom	ekg-story:StoryStereoTypeValidationRule

21.4.3 Update

Label: Update

Identifier: `ekg-story:StoryStereoTypeUpdate`

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.

IRI: `https://ekgf.org/ontology/story#StoryStereoTypeUpdate`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-story:StoryStereoType
rdfs:isDefinedBy	ekg-story:
owl:differentFrom	ekg-story:StoryStereoTypeTransformationRule
owl:differentFrom	ekg-story:StoryStereoTypeValidationRule

21.4.4 Validation Rule

Label: Validation Rule

Identifier: `ekg-story:StoryStereoTypeValidationRule`

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.

IRI: `https://ekgf.org/ontology/story#StoryStereoTypeValidationRule`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-story:StoryStereoType
rdfs:isDefinedBy	ekg-story:

21.4.5 Story Validation Shapes

Label: Story Validation Shapes

Identifier: `ekg-story:ValidationShapeSet`

Instance of: ekg-shape:ShapeSet

Validates optional story stereotypes and their definitions, story-owned concept usages, input flags, single-story implementation ownership, and migration from retired vocabulary.

IRI: `https://ekgf.org/ontology/story#ValidationShapeSet`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-shape:ShapeSet
`rdfs:isDefinedBy`	ekg-story:
`ekg-shape:hasShape`	ekg-story:LegacyConceptUsageShape
`ekg-shape:hasShape`	ekg-story:LegacyImplementationShape
`ekg-shape:hasShape`	ekg-story:StoryClassificationShape
`ekg-shape:hasShape`	ekg-story:StoryConceptUsageShape
`ekg-shape:hasShape`	ekg-story:StoryImplementationShape
`ekg-shape:hasShape`	ekg-story:StoryInputShape
`ekg-shape:hasShape`	ekg-story:StoryStereoTypeShape
`ekg-shape:hasShapesGraph`	ekg-story:
`ekg-shape:shapePurpose`	ekg-shape:Validation

21.5 Shapes

21.5.1 Legacy Concept Usage Shape

Label: Legacy Concept Usage Shape

Identifier: `ekg-story:LegacyConceptUsageShape`

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.

IRI: `https://ekgf.org/ontology/story#LegacyConceptUsageShape`

RDF predicate	Value
`rdf:type`	`sh:NodeShape`
`rdfs:isDefinedBy`	ekg-story:
`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`: serverProvidedIRI was removed. Identify the input usage and set isServerProvided on that usage.; `sh:path`: `ekg-story:serverProvidedIRI`)
`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`: hasInput was removed. Use hasInputConcept to link to a StoryInput usage object.; `sh:path`: `ekg-story:hasInput`)

RDF predicate	Value
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

21.5.2 Legacy Implementation Shape

Label: Legacy Implementation Shape

Identifier: `ekg-story:LegacyImplementationShape`

Rejects the retired hasImplementation predicate. Story implementation links use implementedBy and its inverse implementationOf.

IRI: `https://ekgf.org/ontology/story#LegacyImplementationShape`

RDF predicate	Value
rdf:type	sh:NodeShape
rdfs:isDefinedBy	ekg-story:
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

21.5.3 Story Classification Shape

Label: Story Classification Shape

Identifier: `ekg-story:StoryClassificationShape`

Validates optional primary-purpose classification on stories, rejecting multiple stereotype terms, nonresource stories, and classification assigned to implementations, concept usages, or stereotype resources.

IRI: `https://ekgf.org/ontology/story#StoryClassificationShape`

RDF predicate	Value
rdf:type	sh:NodeShape
rdfs:isDefinedBy	ekg-story:
sh:nodeKind	sh:BlankNodeOrIRI
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

21.5.4 Story Concept Usage Shape

Label: Story Concept Usage Shape

Identifier: `ekg-story:StoryConceptUsageShape`

Validates one owning story, one referenced concept, and exactly one input, output, or dependent role. Usage objects are distinct from stories, concepts, and manifestations.

IRI: `https://ekgf.org/ontology/story#StoryConceptUsageShape`

RDF predicate	Value
`rdf:type`	`sh:NodeShape`
`rdfs:isDefinedBy`	ekg-story:
`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))

21.5.5 Story Implementation Shape

Label: Story Implementation Shape

Identifier: `ekg-story:StoryImplementationShape`

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.

IRI: `https://ekgf.org/ontology/story#StoryImplementationShape`

RDF predicate	Value
`rdf:type`	`sh:NodeShape`
`rdfs:isDefinedBy`	ekg-story:
`sh:nodeKind`	`sh:BlankNodeOrIRI`

RDF predicate	Value
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

21.5.6 Story Input Shape

Label: Story Input Shape

Identifier: `ekg-story:StoryInputShape`

Validates optional, single-valued boolean input flags and records their false application defaults without materializing them.

IRI: `https://ekgf.org/ontology/story#StoryInputShape`

RDF predicate	Value
rdf:type	sh:NodeShape
rdfs:isDefinedBy	ekg-story:
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

21.5.7 Story Stereotype Shape

Label: Story Stereotype Shape

Identifier: `ekg-story:StoryStereoTypeShape`

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.

IRI: `https://ekgf.org/ontology/story#StoryStereoTypeShape`

RDF predicate	Value
rdf:type	sh:NodeShape
rdfs:isDefinedBy	ekg-story:
sh:nodeKind	sh:IRI

RDF predicate	Value
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: 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:targetClass	ekg-story:StoryStereoType
sh:targetObjectsOf	ekg-story:hasStereoType

22 SPARQL Story Implementation Ontology

22.1 Overview

SPARQL Story Implementation Ontology

Identifier	Value
Ontology IRI / Namespace	https://ekgf.org/ontology/story-impl-sparql#
Prefix	ekg-story-impl-sparql

22.1.1 Source language, renderer, and generated 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 `ValidationShapeSet`.

See the repository's template processing guide for input serialization requirements and an executable metadata example.

22.2 Classes

22.2.1 ASK

Label: ASK

Identifier: `ekg-story-impl-sparql:ASK`

A SPARQL query story implementation using the ASK query form to produce a boolean result.

IRI: `https://ekgf.org/ontology/story-impl-sparql#ASK`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-story-impl-sparql:
`rdfs:seeAlso`	`https://www.w3.org/TR/sparql11-query/#ask`
`rdfs:subClassOf`	ekg-story-impl-sparql:StorySPARQLQueryImplementation

22.2.2 CONSTRUCT

Label: CONSTRUCT

Identifier: `ekg-story-impl-sparql:CONSTRUCT`

A SPARQL query story implementation using the CONSTRUCT query form to produce an RDF graph.

IRI: `https://ekgf.org/ontology/story-impl-sparql#CONSTRUCT`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-story-impl-sparql:
`rdfs:seeAlso`	`https://www.w3.org/TR/sparql11-query/#construct`
`rdfs:subClassOf`	ekg-story-impl-sparql:StorySPARQLQueryImplementation

22.2.3 DELETE

Label: DELETE

Identifier: `ekg-story-impl-sparql:DELETE`

A SPARQL update story implementation classified as deleting data.

IRI: `https://ekgf.org/ontology/story-impl-sparql#DELETE`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-story-impl-sparql:
`rdfs:seeAlso`	`https://www.w3.org/TR/sparql11-update/#deleteInsert`
`rdfs:subClassOf`	ekg-story-impl-sparql:StorySPARQLUpdateImplementation

22.2.4 DESCRIBE

Label: DESCRIBE

Identifier: `ekg-story-impl-sparql:DESCRIBE`

A SPARQL query story implementation using the DESCRIBE query form to obtain an RDF description of resources.

IRI: `https://ekgf.org/ontology/story-impl-sparql#DESCRIBE`

RDF predicate	Value
`rdf:type`	`owl:Class`

RDF predicate	Value
rdfs:isDefinedBy	ekg-story-impl-sparql:
rdfs:seeAlso	https://www.w3.org/TR/sparql11-query/#describe
rdfs:subClassOf	ekg-story-impl-sparql:StorySPARQLQueryImplementation

22.2.5 DROP

Label: DROP

Identifier: `ekg-story-impl-sparql:DROP`

A SPARQL update story implementation classified as dropping a graph.

IRI: `https://ekgf.org/ontology/story-impl-sparql#DROP`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-story-impl-sparql:
rdfs:seeAlso	https://www.w3.org/TR/sparql11-update/#drop
rdfs:subClassOf	ekg-story-impl-sparql:StorySPARQLUpdateImplementation

22.2.6 RDFox SPARQL flavor

Label: RDFox SPARQL flavor

Identifier: `ekg-story-impl-sparql:RDFoxSPARQL`

The RDFox-specific SPARQL flavor represented in this vocabulary as a specialization of SPARQL11.

IRI: `https://ekgf.org/ontology/story-impl-sparql#RDFoxSPARQL`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-story-impl-sparql:
rdfs:subClassOf	ekg-story-impl-sparql:SPARQL11

22.2.7 SELECT

Label: SELECT

Identifier: `ekg-story-impl-sparql:SELECT`

A SPARQL query story implementation using the SELECT query form to return variable bindings.

IRI: `https://ekgf.org/ontology/story-impl-sparql#SELECT`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-story-impl-sparql:
rdfs:seeAlso	https://www.w3.org/TR/sparql11-query/#select
rdfs:subClassOf	ekg-story-impl-sparql:StorySPARQLQueryImplementation

22.2.8 SPARQL 1.0

Label: SPARQL 1.0

Identifier: `ekg-story-impl-sparql:SPARQL10`

The SPARQL 1.0 flavor in the story implementation vocabulary.

IRI: `https://ekgf.org/ontology/story-impl-sparql#SPARQL10`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-story-impl-sparql:
rdfs:subClassOf	ekg-story-impl-sparql:SPARQLFlavor

22.2.9 SPARQL 1.1

Label: SPARQL 1.1

Identifier: `ekg-story-impl-sparql:SPARQL11`

The SPARQL 1.1 flavor in the story implementation vocabulary.

IRI: `https://ekgf.org/ontology/story-impl-sparql#SPARQL11`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-story-impl-sparql:
rdfs:subClassOf	ekg-story-impl-sparql:SPARQLFlavor

22.2.10 SPARQL database product

Label: SPARQL database product

Identifier: `ekg-story-impl-sparql:SPARQLDatabaseProduct`

A database product entry used to identify a platform for SPARQL story implementations.

IRI: `https://ekgf.org/ontology/story-impl-sparql#SPARQLDatabaseProduct`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-story-impl-sparql:

22.2.11 SPARQL database vendor

Label: SPARQL database vendor

Identifier: `ekg-story-impl-sparql:SPARQLDatabaseVendor`

A vendor entry in the SPARQL database implementation vocabulary, distinct from a database product.

IRI: `https://ekgf.org/ontology/story-impl-sparql#SPARQLDatabaseVendor`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-story-impl-sparql:

22.2.12 SPARQL Flavor

Label: SPARQL Flavor

Identifier: `ekg-story-impl-sparql:SPARQLFlavor`

A SPARQL language or implementation-specific dialect used to classify story implementations.

IRI: `https://ekgf.org/ontology/story-impl-sparql#SPARQLFlavor`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-story-impl-sparql:

22.2.13 Stardog SPARQL flavor

Label: Stardog SPARQL flavor

Identifier: `ekg-story-impl-sparql:StardogSPARQL`

The Stardog-specific SPARQL flavor represented in this vocabulary as a specialization of SPARQL11.

IRI: `https://ekgf.org/ontology/story-impl-sparql#StardogSPARQL`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-story-impl-sparql:
`rdfs:subClassOf`	ekg-story-impl-sparql:SPARQL11

22.2.14 Statement Template

Label: Statement Template

Identifier: `ekg-story-impl-sparql:StatementTemplate`

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.

IRI: `https://ekgf.org/ontology/story-impl-sparql#StatementTemplate`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-story-impl-sparql:
`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)

22.2.15 SPARQL story implementation

Label: SPARQL story implementation

Identifier: `ekg-story-impl-sparql:StorySPARQLImplementation`

A story implementation expressed using a SPARQL statement, with optional file and language-flavor metadata.

IRI: `https://ekgf.org/ontology/story-impl-sparql#StorySPARQLImplementation`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-story-impl-sparql:
`rdfs:subClassOf`	ekg-story:StoryImplementation

22.2.16 SPARQL query story implementation

Label: SPARQL query story implementation

Identifier: `ekg-story-impl-sparql:StorySPARQLQueryImplementation`

Story SPARQL Implementation for query statements like `SELECT`, `CONSTRUCT`, `ASK` and `DESCRIBE`

IRI: `https://ekgf.org/ontology/story-impl-sparql#StorySPARQLQueryImplementation`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-story-impl-sparql:
`rdfs:subClassOf`	ekg-story-impl-sparql:StorySPARQLImplementation

22.2.17 Story SPARQL Template Implementation

Label: Story SPARQL Template Implementation

Identifier: `ekg-story-impl-sparql:StorySPARQLTemplateImplementation`

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.

IRI: `https://ekgf.org/ontology/story-impl-sparql#StorySPARQLTemplateImplementation`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-story-impl-sparql:
`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)

22.2.18 SPARQL update story implementation

Label: SPARQL update story implementation

Identifier: `ekg-story-impl-sparql:StorySPARQLUpdateImplementation`

Story SPARQL Implementation for update statements like `INSERT/DELETE/WHERE` etc

IRI: `https://ekgf.org/ontology/story-impl-sparql#StorySPARQLUpdateImplementation`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-story-impl-sparql:
`rdfs:subClassOf`	ekg-story-impl-sparql:StorySPARQLImplementation

22.2.19 Template Renderer

Label: Template Renderer

Identifier: `ekg-story-impl-sparql:TemplateRenderer`

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.

IRI: `https://ekgf.org/ontology/story-impl-sparql#TemplateRenderer`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-story-impl-sparql:
`rdfs:subClassOf`	(`rdf:type`: `owl:Restriction`; `owl:minCardinality`: 1; `owl:onProperty`: ekg-story-impl-sparql:supportsTemplateLanguage)

22.2.20 UPDATE

Label: UPDATE

Identifier: `ekg-story-impl-sparql:UPDATE`

A story implementation classified as a SPARQL update. This class does not distinguish the individual SPARQL update forms.

IRI: `https://ekgf.org/ontology/story-impl-sparql#UPDATE`

RDF predicate	Value
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-story-impl-sparql:
rdfs:seeAlso	https://www.w3.org/TR/sparql11-update/
rdfs:subClassOf	ekg-story-impl-sparql:StorySPARQLUpdateImplementation

22.3 Properties

22.3.1 File name

Label: File name

Identifier: `ekg-story-impl-sparql:fileName`

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.

IRI: `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:isDefinedBy	ekg-story-impl-sparql:

22.3.2 Supports SPARQL Flavour

Label: Supports SPARQL Flavour

Identifier: `ekg-story-impl-sparql:flavor`

Identifies the SPARQL language or implementation-specific flavor used by a SPARQL story implementation.

IRI: `https://ekgf.org/ontology/story-impl-sparql#flavor`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-story-impl-sparql:StorySPARQLImplementation
rdfs:isDefinedBy	ekg-story-impl-sparql:
rdfs:range	ekg-story-impl-sparql:SPARQLFlavor

22.3.3 has template

Label: has template

Identifier: `ekg-story-impl-sparql:hasTemplate`

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.

IRI: `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:isDefinedBy	ekg-story-impl-sparql:

RDF predicate	Value
rdfs:range	ekg-story-impl-sparql:StatementTemplate

22.3.4 SPARQL statement

Label: SPARQL statement

Identifier: `ekg-story-impl-sparql:sparql`

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.

IRI: `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:isDefinedBy	ekg-story-impl-sparql:

22.3.5 supports template language

Label: supports template language

Identifier: `ekg-story-impl-sparql:supportsTemplateLanguage`

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.

IRI: `https://ekgf.org/ontology/story-impl-sparql#supportsTemplateLanguage`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-story-impl-sparql:TemplateRenderer
rdfs:isDefinedBy	ekg-story-impl-sparql:
rdfs:range	ekg-concept:TemplateLanguage

22.3.6 template file name

Label: template file name

Identifier: `ekg-story-impl-sparql:templateFileName`

The file containing a template document, relative to the document's source context. It identifies template source rather than a file of executable SPARQL.

IRI: `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:isDefinedBy	ekg-story-impl-sparql:
rdfs:range	xsd:string

22.3.7 template text

Label: template text

Identifier: `ekg-story-impl-sparql:templateText`

The unrendered template document's source text, interpreted in its declared template language. Parsing this text as SPARQL does not replace template rendering.

IRI: `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:isDefinedBy`	ekg-story-impl-sparql:
`rdfs:range`	`xsd:string`

22.3.8 uses renderer

Label: uses renderer

Identifier: `ekg-story-impl-sparql:usesRenderer`

Identifies the renderer selected for this template-based implementation. It must explicitly support the template document's declared language/profile.

IRI: `https://ekgf.org/ontology/story-impl-sparql#usesRenderer`

RDF predicate	Value
`rdf:type`	`owl:FunctionalProperty`
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-story-impl-sparql:StorySPARQLTemplateImplementation
`rdfs:isDefinedBy`	ekg-story-impl-sparql:
`rdfs:range`	ekg-story-impl-sparql:TemplateRenderer

22.4 Named Individuals

22.4.1 Allegro Graph

Label: Allegro Graph

Identifier: `ekg-story-impl-sparql:AllegroGraph`

Instance of: ekg-story-impl-sparql:SPARQLDatabaseProduct

The AllegroGraph product entry used to identify a database platform for SPARQL story implementations.

IRI: `https://ekgf.org/ontology/story-impl-sparql#AllegroGraph`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-story-impl-sparql:SPARQLDatabaseProduct
`rdfs:isDefinedBy`	ekg-story-impl-sparql:

22.4.2 Amazon Web Services

Label: Amazon Web Services

Identifier: `ekg-story-impl-sparql:AmazonWebServicesInc`

Instance of: ekg-story-impl-sparql:SPARQLDatabaseVendor

The vendor entry named Amazon Web Services in the SPARQL database implementation vocabulary.

IRI: `https://ekgf.org/ontology/story-impl-sparql#AmazonWebServicesInc`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-story-impl-sparql:SPARQLDatabaseVendor

RDF predicate	Value
rdfs:isDefinedBy	ekg-story-impl-sparql:

22.4.3 Anzo Graph

Label: Anzo Graph

Identifier: `ekg-story-impl-sparql:AnzoGraph`

Instance of: ekg-story-impl-sparql:SPARQLDatabaseProduct

The AnzoGraph product entry used to identify a database platform for SPARQL story implementations.

IRI: `https://ekgf.org/ontology/story-impl-sparql#AnzoGraph`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-story-impl-sparql:SPARQLDatabaseProduct
rdfs:isDefinedBy	ekg-story-impl-sparql:

22.4.4 Blazegraph

Label: Blazegraph

Identifier: `ekg-story-impl-sparql:BlazeGraph`

Instance of: ekg-story-impl-sparql:SPARQLDatabaseProduct

The Blazegraph product entry used to identify a database platform for SPARQL story implementations.

IRI: `https://ekgf.org/ontology/story-impl-sparql#BlazeGraph`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-story-impl-sparql:SPARQLDatabaseProduct
rdfs:isDefinedBy	ekg-story-impl-sparql:

22.4.5 Cambridge Semantics

Label: Cambridge Semantics

Identifier: `ekg-story-impl-sparql:CambridgeSemantics`

Instance of: ekg-story-impl-sparql:SPARQLDatabaseVendor

The vendor entry named Cambridge Semantics in the SPARQL database implementation vocabulary.

IRI: `https://ekgf.org/ontology/story-impl-sparql#CambridgeSemantics`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-story-impl-sparql:SPARQLDatabaseVendor
rdfs:isDefinedBy	ekg-story-impl-sparql:

22.4.6 GraphDB

Label: GraphDB

Identifier: `ekg-story-impl-sparql:GraphDb`

Instance of: ekg-story-impl-sparql:SPARQLDatabaseProduct

The GraphDB product entry used to identify a database platform for SPARQL story implementations.

IRI: `https://ekgf.org/ontology/story-impl-sparql#GraphDb`

RDF predicate	Value
rdf:type	owl:NamedIndividual

RDF predicate	Value
rdf:type	ekg-story-impl-sparql:SPARQLDatabaseProduct
rdfs:isDefinedBy	ekg-story-impl-sparql:

22.4.7 Neptune

Label: Neptune

Identifier: `ekg-story-impl-sparql:Neptune`

Instance of: ekg-story-impl-sparql:SPARQLDatabaseProduct

The Neptune product entry used to identify a database platform for SPARQL story implementations.

IRI: `https://ekgf.org/ontology/story-impl-sparql#Neptune`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-story-impl-sparql:SPARQLDatabaseProduct
rdfs:isDefinedBy	ekg-story-impl-sparql:

22.4.8 Ontotext

Label: Ontotext

Identifier: `ekg-story-impl-sparql:Ontotext`

Instance of: ekg-story-impl-sparql:SPARQLDatabaseVendor

The vendor entry named Ontotext in the SPARQL database implementation vocabulary.

IRI: `https://ekgf.org/ontology/story-impl-sparql#Ontotext`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-story-impl-sparql:SPARQLDatabaseVendor
rdfs:isDefinedBy	ekg-story-impl-sparql:

22.4.9 Oxford Semantic Technologies

Label: Oxford Semantic Technologies

Identifier: `ekg-story-impl-sparql:OxfordSemanticTechnologies`

Instance of: ekg-story-impl-sparql:SPARQLDatabaseVendor

The vendor entry named Oxford Semantic Technologies in the SPARQL database implementation vocabulary.

IRI: `https://ekgf.org/ontology/story-impl-sparql#OxfordSemanticTechnologies`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-story-impl-sparql:SPARQLDatabaseVendor
rdfs:isDefinedBy	ekg-story-impl-sparql:

22.4.10 RD Fox

Label: RD Fox

Identifier: `ekg-story-impl-sparql:RDFox`

Instance of: ekg-story-impl-sparql:SPARQLDatabaseProduct

The RDFox product entry used to identify a database platform for SPARQL story implementations.

IRI: `https://ekgf.org/ontology/story-impl-sparql#RDFox`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-story-impl-sparql:SPARQLDatabaseProduct
rdfs:isDefinedBy	ekg-story-impl-sparql:

22.4.11 Stardog

Label: Stardog

Identifier: `ekg-story-impl-sparql:Stardog`

Instance of: ekg-story-impl-sparql:SPARQLDatabaseProduct

The Stardog product entry used to identify a database platform for SPARQL story implementations.

IRI: `https://ekgf.org/ontology/story-impl-sparql#Stardog`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-story-impl-sparql:SPARQLDatabaseProduct
rdfs:isDefinedBy	ekg-story-impl-sparql:

22.4.12 Stardog Union

Label: Stardog Union

Identifier: `ekg-story-impl-sparql:StardogUnion`

Instance of: ekg-story-impl-sparql:SPARQLDatabaseVendor

The vendor entry named Stardog Union in the SPARQL database implementation vocabulary.

IRI: `https://ekgf.org/ontology/story-impl-sparql#StardogUnion`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-story-impl-sparql:SPARQLDatabaseVendor
rdfs:isDefinedBy	ekg-story-impl-sparql:

22.4.13 SPARQL Story Implementation Validation Shapes

Label: SPARQL Story Implementation Validation Shapes

Identifier: `ekg-story-impl-sparql:ValidationShapeSet`

Instance of: ekg-shape:ShapeSet

Validates template documents, renderer language capabilities, and template-based SPARQL story implementations.

IRI: `https://ekgf.org/ontology/story-impl-sparql#ValidationShapeSet`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-shape:ShapeSet
rdfs:isDefinedBy	ekg-story-impl-sparql:
ekg-shape:hasShape	ekg-story-impl-sparql:StatementTemplateShape
ekg-shape:hasShape	ekg-story-impl-sparql:StorySPARQLTemplateImplementationShape
ekg-shape:hasShape	ekg-story-impl-sparql:TemplateRendererShape
ekg-shape:hasShapesGraph	ekg-story-impl-sparql:

RDF predicate	Value
ekg-shape:shapePurpose	ekg-shape:Validation

22.4.14 Virtuoso

Label: Virtuoso

Identifier: ekg-story-impl-sparql:Virtuoso

Instance of: ekg-story-impl-sparql:SPARQLDatabaseProduct

The Virtuoso product entry used to identify a database platform for SPARQL story implementations.

IRI: https://ekgf.org/ontology/story-impl-sparql#Virtuoso

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-story-impl-sparql:SPARQLDatabaseProduct
rdfs:isDefinedBy	ekg-story-impl-sparql:

22.5 Shapes

22.5.1 Statement Template Shape

Label: Statement Template Shape

Identifier: ekg-story-impl-sparql:StatementTemplateShape

Requires one documented template-language profile and nonempty inline or file-based template source. Source content and its file location may both be supplied.

IRI: https://ekgf.org/ontology/story-impl-sparql#StatementTemplateShape

RDF predicate	Value
rdf:type	sh:NodeShape
rdfs:isDefinedBy	ekg-story-impl-sparql:
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:templateText)
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: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

22.5.2 Story SPARQL Template Implementation Shape

Label: Story SPARQL Template Implementation Shape

Identifier: ekg-story-impl-sparql:StorySPARQLTemplateImplementationShape

Requires one template and one renderer with matching language support, and rejects mixing template source with executable-SPARQL source fields.

IRI: `https://ekgf.org/ontology/story-impl-sparql#StorySPARQLTemplateImplementationShape`

RDF predicate	Value
`rdf:type`	`sh:NodeShape`
`rdfs:isDefinedBy`	ekg-story-impl-sparql:
`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:class`: ekg-story-impl-sparql:TemplateRenderer; `sh:maxCount`: 1; `sh:minCount`: 1; `sh:nodeKind`: `sh:BlankNodeOrIRI`; `sh:path`: ekg-story-impl-sparql:usesRenderer)
`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:maxCount`: 0; `sh:message`: A template source file belongs to the linked StatementTemplate through templateFileName.; `sh:path`: ekg-story-impl-sparql:fileName)
`sh:sparql`	(`sh:message`: The selected renderer must explicitly support the template document's declared language/profile.; `sh:select`: 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 . } })
`sh:targetClass`	ekg-story-impl-sparql:StorySPARQLTemplateImplementation
`sh:targetSubjectsOf`	ekg-story-impl-sparql:hasTemplate
`sh:targetSubjectsOf`	ekg-story-impl-sparql:usesRenderer

22.5.3 Template Renderer Shape

Label: Template Renderer Shape

Identifier: `ekg-story-impl-sparql:TemplateRendererShape`

Requires a renderer to explicitly identify the template-language profiles it supports. Execution approval and safe substitution remain application responsibilities.

IRI: `https://ekgf.org/ontology/story-impl-sparql#TemplateRendererShape`

RDF predicate	Value
`rdf:type`	`sh:NodeShape`
`rdfs:isDefinedBy`	ekg-story-impl-sparql:
`sh:nodeKind`	`sh:BlankNodeOrIRI`

RDF predicate	Value
`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

23 Story Service Ontology

23.1 Overview

Story Service Ontology

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

23.1.1 Executing stories

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.

23.1.1.1 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.

23.1.1.2 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 language. The service renders that document into a separate SPARQL statement for each invocation; the generated text is parsed before it is sent for execution.

23.2 Classes

23.2.1 Parameter

Label: Parameter

Identifier: `ekg-story-service:Parameter`

A parameter description used by an executable statement in the story service. Parameters are distinct from statements and service instances.

IRI: `https://ekgf.org/ontology/story-service#Parameter`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-story-service:
`owl:disjointWith`	ekg-story-service:Statement
`owl:disjointWith`	ekg-story-service:StoryService

23.2.2 Result

Label: Result

Identifier: `ekg-story-service:Result`

A description of an output supplied by a story-service execution.

IRI: `https://ekgf.org/ontology/story-service#Result`

RDF predicate	Value
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-story-service:

23.2.3 SHACL Statement

Label: SHACL Statement

Identifier: `ekg-story-service:SHACLStatement`

A story-service statement expressed using SHACL.

IRI: `https://ekgf.org/ontology/story-service#SHACLStatement`

RDF predicate	Value
`http://purl.org/dc/elements/1.1/creator`	`http://orcid.org/0000-0003-2309-5068`
`http://purl.org/dc/elements/1.1/date`	2020-04-01T18:14:13Z
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-story-service:
`rdfs:subClassOf`	ekg-story-service:Statement

23.2.4 SPARQL Parameter

Label: SPARQL Parameter

Identifier: `ekg-story-service:SPARQLParameter`

A statement parameter used by a SPARQL story implementation.

IRI: `https://ekgf.org/ontology/story-service#SPARQLParameter`

RDF predicate	Value
`http://purl.org/dc/elements/1.1/creator`	`http://orcid.org/0000-0003-2309-5068`
`http://purl.org/dc/elements/1.1/date`	2020-04-01T17:25:05Z
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-story-service:
`rdfs:subClassOf`	ekg-story-service:Parameter

23.2.5 SPARQL Results

Label: SPARQL Results

Identifier: `ekg-story-service:SPARQLResults`

A story-service result description for the output of a SPARQL execution.

IRI: `https://ekgf.org/ontology/story-service#SPARQLResults`

RDF predicate	Value
`http://purl.org/dc/elements/1.1/creator`	`http://orcid.org/0000-0003-2309-5068`
`http://purl.org/dc/elements/1.1/date`	2020-04-07T20:40:18Z
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-story-service:
`rdfs:subClassOf`	ekg-story-service:Result

23.2.6 SPARQL Statement

Label: SPARQL Statement

Identifier: `ekg-story-service:SPARQLStatement`

A story-service statement expressed using SPARQL.

IRI: `https://ekgf.org/ontology/story-service#SPARQLStatement`

RDF predicate	Value
`http://purl.org/dc/elements/1.1/creator`	`http://orcid.org/0000-0003-2309-5068`
`http://purl.org/dc/elements/1.1/date`	2020-04-01T17:24:10Z
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-story-service:
`rdfs:subClassOf`	ekg-story-service:Statement

23.2.7 Statement

Label: Statement

Identifier: `ekg-story-service: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.

IRI: `https://ekgf.org/ontology/story-service#Statement`

RDF predicate	Value
`http://purl.org/dc/elements/1.1/creator`	`http://orcid.org/0000-0003-2309-5068`
`http://purl.org/dc/elements/1.1/date`	2020-04-01T17:24:43Z
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-story-service:

RDF predicate	Value
owl:disjointWith	ekg-story-service:StoryService

23.2.8 Story Service

Label: Story Service

Identifier: `ekg-story-service:StoryService`

23.2.8.1 Execution and defaults

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.

IRI: `https://ekgf.org/ontology/story-service#StoryService`

RDF predicate	Value
http://purl.org/dc/elements/1.1/creator	http://orcid.org/0000-0003-2309-5068
http://purl.org/dc/elements/1.1/date	2020-04-01T17:28:32Z
rdf:type	owl:Class
rdfs:isDefinedBy	ekg-story-service:

23.3 Properties

23.3.1 Has Parameter

Label: Has Parameter

Identifier: `ekg-story-service:hasParameter`

Associates a statement with one of its parameters; it is the inverse of parameterFor.

IRI: `https://ekgf.org/ontology/story-service#hasParameter`

RDF predicate	Value
http://purl.org/dc/elements/1.1/creator	http://orcid.org/0000-0003-2309-5068
http://purl.org/dc/elements/1.1/date	2020-04-01T18:16:18Z
rdf:type	owl:InverseFunctionalProperty
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-story-service:Statement
rdfs:isDefinedBy	ekg-story-service:
rdfs:range	ekg-story-service:Parameter
owl:inverseOf	ekg-story-service:parameterFor

23.3.2 Is Service Parameter

Label: Is Service Parameter

Identifier: `ekg-story-service:isServiceParameter`

Records whether a parameter is classified as a service parameter by the story-service configuration.

IRI: `https://ekgf.org/ontology/story-service#isServiceParameter`

RDF predicate	Value
http://purl.org/dc/elements/1.1/creator	http://orcid.org/0000-0003-2309-5068
http://purl.org/dc/elements/1.1/date	2020-04-08T00:06:23Z
rdf:type	owl:DatatypeProperty

RDF predicate	Value
rdfs:domain	ekg-story-service:Parameter
rdfs:isDefinedBy	ekg-story-service:
rdfs:range	xsd:boolean

23.3.3 On

Label: On

Identifier: `ekg-story-service:on`

Identifies the story to which a parameter or result description relates.

IRI: `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:isDefinedBy	ekg-story-service:
rdfs:range	ekg-story:Story

23.3.4 Parameter Direction

Label: Parameter Direction

Identifier: `ekg-story-service:parameterDirection`

Records the direction assigned to a parameter as a string. The ontology does not declare a controlled set of direction values.

IRI: `https://ekgf.org/ontology/story-service#parameterDirection`

RDF predicate	Value
http://purl.org/dc/elements/1.1/creator	http://orcid.org/0000-0003-2309-5068
http://purl.org/dc/elements/1.1/date	2020-04-08T00:03:25Z
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-story-service:Parameter
rdfs:isDefinedBy	ekg-story-service:
rdfs:range	xsd:string

23.3.5 Parameter For

Label: Parameter For

Identifier: `ekg-story-service:parameterFor`

Identifies the statement for which a parameter is defined. The functional-property axiom limits a parameter to one statement.

IRI: `https://ekgf.org/ontology/story-service#parameterFor`

RDF predicate	Value
http://purl.org/dc/elements/1.1/creator	http://orcid.org/0000-0003-2309-5068
http://purl.org/dc/elements/1.1/date	2020-04-01T17:32:57Z
rdf:type	owl:FunctionalProperty
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-story-service:Parameter

RDF predicate	Value
rdfs:isDefinedBy	ekg-story-service:
rdfs:range	ekg-story-service:Statement

23.3.6 Parameter Key

Label: Parameter Key

Identifier: `ekg-story-service:parameterKey`

Records the string key by which a parameter is identified in a statement or service configuration.

IRI: `https://ekgf.org/ontology/story-service#parameterKey`

RDF predicate	Value
http://purl.org/dc/elements/1.1/creator	http://orcid.org/0000-0003-2309-5068
http://purl.org/dc/elements/1.1/date	2020-04-01T18:19:13Z
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-story-service:Parameter
rdfs:isDefinedBy	ekg-story-service:
rdfs:range	xsd:string

23.3.7 Parameter Op

Label: Parameter Op

Identifier: `ekg-story-service:parameterOp`

Records an operation value associated with a parameter. The ontology does not specify its literal datatype or the allowed operations.

IRI: `https://ekgf.org/ontology/story-service#parameterOp`

RDF predicate	Value
http://purl.org/dc/elements/1.1/creator	http://orcid.org/0000-0003-2309-5068
http://purl.org/dc/elements/1.1/date	2020-04-01T18:21:12Z
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-story-service:Parameter
rdfs:isDefinedBy	ekg-story-service:

23.3.8 Parameter Type

Label: Parameter Type

Identifier: `ekg-story-service:parameterType`

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.

IRI: `https://ekgf.org/ontology/story-service#parameterType`

RDF predicate	Value
http://purl.org/dc/elements/1.1/creator	http://orcid.org/0000-0003-2309-5068
http://purl.org/dc/elements/1.1/date	2020-04-01T18:20:08Z
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty

RDF predicate	Value
rdfs:domain	ekg-story-service:Parameter
rdfs:isDefinedBy	ekg-story-service:
rdfs:range	xsd:string

23.3.9 should be supplied by

Label: should be supplied by

Identifier: `ekg-story-service:providedBy`

The execution of any given story can be done by (supplied by) a given (logical) instance of the story Service.

IRI: `https://ekgf.org/ontology/story-service#providedBy`

RDF predicate	Value
http://purl.org/dc/elements/1.1/creator	http://orcid.org/0000-0003-2309-5068
http://purl.org/dc/elements/1.1/date	2020-04-08T00:10:29Z
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-story:Story
rdfs:isDefinedBy	ekg-story-service:
rdfs:range	ekg-story-service:StoryService

23.3.10 Result For

Label: Result For

Identifier: `ekg-story-service:resultFor`

Relates a SPARQL result description to the story that produces the result.

IRI: `https://ekgf.org/ontology/story-service#resultFor`

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-story-service:SPARQLResults
rdfs:isDefinedBy	ekg-story-service:
rdfs:range	ekg-story:Story

23.3.11 Result Key

Label: Result Key

Identifier: `ekg-story-service:resultKey`

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.

IRI: `https://ekgf.org/ontology/story-service#resultKey`

RDF predicate	Value
http://purl.org/dc/elements/1.1/creator	http://orcid.org/0000-0003-2309-5068
http://purl.org/dc/elements/1.1/date	2020-04-07T20:46:55Z
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-story-service:Parameter
rdfs:isDefinedBy	ekg-story-service:
rdfs:range	xsd:string

23.3.12 Result Type

Label: Result Type

Identifier: `ekg-story-service:resultType`

Records the result type identifier used by the story-service configuration as a string.

IRI: `https://ekgf.org/ontology/story-service#resultType`

RDF predicate	Value
http://purl.org/dc/elements/1.1/creator	http://orcid.org/0000-0003-2309-5068
http://purl.org/dc/elements/1.1/date	2020-04-07T20:51:59Z
rdf:type	owl:DatatypeProperty
rdfs:domain	ekg-story-service:Result
rdfs:isDefinedBy	ekg-story-service:
rdfs:range	xsd:string

23.3.13 Should Be Generated By

Label: Should Be Generated By

Identifier: `ekg-story-service:shouldBeGeneratedBy`

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.

IRI: `https://ekgf.org/ontology/story-service#shouldBeGeneratedBy`

RDF predicate	Value
http://purl.org/dc/elements/1.1/creator	http://orcid.org/0000-0003-2309-5068
http://purl.org/dc/elements/1.1/date	2020-04-01T17:25:50Z
rdf:type	owl:FunctionalProperty
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-story-service:Parameter
rdfs:isDefinedBy	ekg-story-service:
rdfs:range	ekg-story-service:StoryService

23.3.14 Sparql Query Name

Label: Sparql Query Name

Identifier: `ekg-story-service:sparqlQueryName`

Records the configured name of a story's SPARQL query.

IRI: `https://ekgf.org/ontology/story-service#sparqlQueryName`

RDF predicate	Value
http://purl.org/dc/elements/1.1/creator	http://orcid.org/0000-0003-2309-5068
http://purl.org/dc/elements/1.1/date	2020-04-07T20:50:13Z
rdf:type	owl:DatatypeProperty
rdf:type	owl:FunctionalProperty
rdfs:domain	ekg-story:Story
rdfs:isDefinedBy	ekg-story-service:
rdfs:range	xsd:string

23.3.15 Sparql Statement File Name

Label: Sparql Statement File Name

Identifier: `ekg-story-service:sparqlStatementFileName`

Records the name of the SPARQL statement file associated with a story.

IRI: `https://ekgf.org/ontology/story-service#sparqlStatementFileName`

RDF predicate	Value
`http://purl.org/dc/elements/1.1/creator`	`http://orcid.org/0000-0003-2309-5068`
`http://purl.org/dc/elements/1.1/date`	2020-04-07T20:51:10Z
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-story:Story
`rdfs:isDefinedBy`	ekg-story-service:
`rdfs:range`	`xsd:string`

24 Use Case Ontology

24.1 Overview

Use Case Ontology

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

24.1.1 Use cases, stories, and concept vocabularies

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.

24.2 Classes

24.2.1 Use Case

Label: Use Case

Identifier: `ekg-use-case:UseCase`

A description of a business interaction or need, organized in a use-case tree and connected to its stories, outcomes, and concepts.

IRI: `https://ekgf.org/ontology/use-case#UseCase`

RDF predicate	Value
`http://purl.org/dc/elements/1.1/creator`	`http://orcid.org/0000-0003-2309-5068`
`http://purl.org/dc/elements/1.1/date`	2020-04-08T00:00:01Z
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-use-case:

24.2.2 Use Case Stereotype

Label: Use Case Stereotype

Identifier: `ekg-use-case:UseCaseStereoType`

The kind of node a use case is in the use case tree.

IRI: `https://ekgf.org/ontology/use-case#UseCaseStereoType`

RDF predicate	Value
`http://purl.org/dc/elements/1.1/creator`	`http://orcid.org/0000-0003-2309-5068`
`rdf:type`	`owl:Class`
`rdfs:isDefinedBy`	ekg-use-case:

24.3 Properties

24.3.1 has colour

Label: has colour

Identifier: `ekg-use-case:hasColor`

Associates a resource with a color resource used for presentation. It is an object property, distinct from a hexadecimal color literal annotation.

IRI: `https://ekgf.org/ontology/use-case#hasColor`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:isDefinedBy`	ekg-use-case:

24.3.2 has outcome

Label: has outcome

Identifier: `ekg-use-case:hasOutcome`

An outcome the use case is meant to achieve.

IRI: `https://ekgf.org/ontology/use-case#hasOutcome`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-use-case:UseCase
`rdfs:isDefinedBy`	ekg-use-case:
`rdfs:range`	ekg-outcome:Outcome

24.3.3 has stereotype

Label: has stereotype

Identifier: `ekg-use-case:hasStereoType`

Assigns a use-case stereotype to a use case. Its functional-property axiom limits a use case to one stereotype through this property.

IRI: `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:isDefinedBy`	ekg-use-case:
`rdfs:range`	ekg-use-case:UseCaseStereoType

24.3.4 hex color

Label: hex color

Identifier: `ekg-use-case:hexColor`
Display color as a CSS hex string, such as #3b82f6.
IRI: `https://ekgf.org/ontology/use-case#hexColor`

RDF predicate	Value
rdf:type	owl:DatatypeProperty
rdfs:isDefinedBy	ekg-use-case:
rdfs:range	xsd:string

24.3.5 Involves persona
Label: Involves persona
Identifier: `ekg-use-case:involvesPersona`
The given persona is involved in the current use case
IRI: `https://ekgf.org/ontology/use-case#involvesPersona`

RDF predicate	Value
http://purl.org/dc/elements/1.1/creator	http://orcid.org/0000-0003-2309-5068
http://purl.org/dc/elements/1.1/date	2020-06-20T14:13:55Z
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-use-case:UseCase
rdfs:isDefinedBy	ekg-use-case:
rdfs:range	ekg-persona:Persona
owl:inverseOf	ekg-use-case:isInvolvedInUseCase

24.3.6 Is involved in
Label: Is involved in
Identifier: `ekg-use-case:isInvolvedInUseCase`
This persona is involved in the given use case
IRI: `https://ekgf.org/ontology/use-case#isInvolvedInUseCase`

RDF predicate	Value
http://purl.org/dc/elements/1.1/creator	http://orcid.org/0000-0003-2309-5068
http://purl.org/dc/elements/1.1/date	2020-04-08T00:01:05Z
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-persona:Persona
rdfs:isDefinedBy	ekg-use-case:
rdfs:range	ekg-use-case:UseCase

24.3.7 is part of
Label: is part of
Identifier: `ekg-use-case:isPartOf`
The parent of the use case in the use case tree. A use case without a parent is a root.
IRI: `https://ekgf.org/ontology/use-case#isPartOf`

RDF predicate	Value
http://purl.org/dc/elements/1.1/creator	http://orcid.org/0000-0003-2309-5068
http://purl.org/dc/elements/1.1/date	2020-06-20T12:36:22Z

RDF predicate	Value
rdf:type	owl:IrreflexiveProperty
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-use-case:UseCase
rdfs:isDefinedBy	ekg-use-case:
rdfs:range	ekg-use-case:UseCase

24.3.8 is used in

Label: is used in

Identifier: `ekg-use-case:isUsedIn`

Another use case in which this use case is used.

IRI: `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:isDefinedBy	ekg-use-case:
rdfs:range	ekg-use-case:UseCase

24.3.9 Is used in use case

Label: Is used in use case

Identifier: `ekg-use-case:isUsedInUseCase`

The current data element concept is used in the given UseCase

IRI: `https://ekgf.org/ontology/use-case#isUsedInUseCase`

RDF predicate	Value
http://purl.org/dc/elements/1.1/creator	http://orcid.org/0000-0003-2309-5068
http://purl.org/dc/elements/1.1/date	2020-06-20T12:40:56Z
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-concept:DataElementConcept
rdfs:isDefinedBy	ekg-use-case:
rdfs:range	ekg-use-case:UseCase
owl:inverseOf	ekg-use-case:usesConcept

24.3.10 owns concept-set

Label: owns concept-set

Identifier: `ekg-use-case:ownsConceptSet`

24.3.10.1 Vocabulary ownership and inheritance

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.
IRI: `https://ekgf.org/ontology/use-case#ownsConceptSet`

RDF predicate	Value
`rdf:type`	`owl:ObjectProperty`
`rdfs:domain`	ekg-use-case:UseCase
`rdfs:isDefinedBy`	ekg-use-case:
`rdfs:range`	ekg-concept:ConceptSet

24.3.11 screen layout index
Label: screen layout index
Identifier: `ekg-use-case:screenLayoutIndex`
Index that places the use case in the screen layout of the use case tree.
IRI: `https://ekgf.org/ontology/use-case#screenLayoutIndex`

RDF predicate	Value
`rdf:type`	`owl:DatatypeProperty`
`rdfs:domain`	ekg-use-case:UseCase
`rdfs:isDefinedBy`	ekg-use-case:
`rdfs:range`	`xsd:integer`

24.3.12 Used in
Label: Used in
Identifier: `ekg-use-case:usedIn`
The use case that a story or an outcome belongs to. It has no domain because both stories and outcomes carry it.
IRI: `https://ekgf.org/ontology/use-case#usedIn`

RDF predicate	Value
`http://purl.org/dc/elements/1.1/creator`	`http://orcid.org/0000-0003-2309-5068`
`http://purl.org/dc/elements/1.1/date`	2020-04-08T00:02:11Z
`rdf:type`	`owl:ObjectProperty`
`rdfs:isDefinedBy`	ekg-use-case:
`rdfs:range`	ekg-use-case:UseCase

24.3.13 Uses concept
Label: Uses concept
Identifier: `ekg-use-case:usesConcept`
Relates a use case to a data element concept involved in its interactions.
IRI: `https://ekgf.org/ontology/use-case#usesConcept`

RDF predicate	Value
`http://purl.org/dc/elements/1.1/creator`	`http://orcid.org/0000-0003-2309-5068`
`http://purl.org/dc/elements/1.1/date`	2020-06-20T12:40:26Z

RDF predicate	Value
rdf:type	owl:ObjectProperty
rdfs:domain	ekg-use-case:UseCase
rdfs:isDefinedBy	ekg-use-case:
rdfs:range	ekg-concept:DataElementConcept

24.4 Named Individuals

24.4.1 Application

Label: Application

Identifier: `ekg-use-case:UseCaseStereoTypeApplication`

Instance of: ekg-use-case:UseCaseStereoType

An application-level use case.

IRI: `https://ekgf.org/ontology/use-case#UseCaseStereoTypeApplication`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-use-case:UseCaseStereoType
rdfs:isDefinedBy	ekg-use-case:
ekg-api:slug	app
ekg-use-case:hexColor	#2563eb

24.4.2 Business Capability

Label: Business Capability

Identifier: `ekg-use-case:UseCaseStereoTypeBusinessCapability`

Instance of: ekg-use-case:UseCaseStereoType

A business capability in the enterprise capability map.

IRI: `https://ekgf.org/ontology/use-case#UseCaseStereoTypeBusinessCapability`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-use-case:UseCaseStereoType
rdfs:isDefinedBy	ekg-use-case:
ekg-api:slug	business-capability
ekg-use-case:hexColor	#365885

24.4.3 Business Capability Area

Label: Business Capability Area

Identifier: `ekg-use-case:UseCaseStereoTypeBusinessCapabilityArea`

Instance of: ekg-use-case:UseCaseStereoType

A grouping of related business capabilities.

IRI: `https://ekgf.org/ontology/use-case#UseCaseStereoTypeBusinessCapabilityArea`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-use-case:UseCaseStereoType
rdfs:isDefinedBy	ekg-use-case:

RDF predicate	Value
ekg-api:slug	business-capability-area
ekg-use-case:hexColor	#5b6d8a

24.4.4 Data Domain

Label: Data Domain

Identifier: `ekg-use-case:UseCaseStereoTypeDataDomain`

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.

IRI: `https://ekgf.org/ontology/use-case#UseCaseStereoTypeDataDomain`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-use-case:UseCaseStereoType
rdfs:isDefinedBy	ekg-use-case:
ekg-api:slug	data-domain
ekg-use-case:hexColor	#0f766e

24.4.5 Data Landscape

Label: Data Landscape

Identifier: `ekg-use-case:UseCaseStereoTypeDataLandscape`

Instance of: ekg-use-case:UseCaseStereoType

The controlled use-case stereotype identifying a data-landscape use case.

IRI: `https://ekgf.org/ontology/use-case#UseCaseStereoTypeDataLandscape`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-use-case:UseCaseStereoType
rdfs:isDefinedBy	ekg-use-case:
ekg-api:slug	data-landscape

24.4.6 Data Subdomain

Label: Data Subdomain

Identifier: `ekg-use-case:UseCaseStereoTypeDataSubdomain`

Instance of: ekg-use-case:UseCaseStereoType

A subdomain that belongs to a parent data domain.

IRI: `https://ekgf.org/ontology/use-case#UseCaseStereoTypeDataSubdomain`

RDF predicate	Value
rdf:type	owl:NamedIndividual
rdf:type	ekg-use-case:UseCaseStereoType
rdfs:isDefinedBy	ekg-use-case:
ekg-api:slug	data-subdomain
ekg-use-case:hexColor	#2ea39a

24.4.7 Data Transformation

Label: Data Transformation

Identifier: `ekg-use-case:UseCaseStereoTypeDataTransformation`

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.

IRI: `https://ekgf.org/ontology/use-case#UseCaseStereoTypeDataTransformation`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-use-case:UseCaseStereoType
`rdfs:isDefinedBy`	ekg-use-case:
`ekg-api:slug`	data-transformation
`ekg-use-case:hexColor`	#b7791f

24.4.8 Root

Label: Root

Identifier: `ekg-use-case:UseCaseStereoTypeRoot`

Instance of: ekg-use-case:UseCaseStereoType

A structural root node in the use case tree. A root use case has no parent. Ideally there is one, but nothing stops you from having several.

IRI: `https://ekgf.org/ontology/use-case#UseCaseStereoTypeRoot`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-use-case:UseCaseStereoType
`rdfs:isDefinedBy`	ekg-use-case:
`ekg-api:slug`	root
`ekg-use-case:hexColor`	#6b7280

24.4.9 Use Case

Label: Use Case

Identifier: `ekg-use-case:UseCaseStereoTypeUseCase`

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.

IRI: `https://ekgf.org/ontology/use-case#UseCaseStereoTypeUseCase`

RDF predicate	Value
`rdf:type`	`owl:NamedIndividual`
`rdf:type`	ekg-use-case:UseCaseStereoType
`rdfs:isDefinedBy`	ekg-use-case:
`ekg-api:slug`	use-case
`ekg-use-case:hexColor`	#5b5bd6

25 User Experience Ontology

25.1 Overview

User Experience Ontology

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

This ontology covers various minor details, "hints", that can be used by User Interface programmers to optimize the user experience. Details like the default sorting of lists of objects, the default color of things etc.

25.2 Properties

25.2.1 Hexadecimal RGB color

Label: Hexadecimal RGB color

Identifier: `ekg-ux:hexColor`

An RGB color code in hex

IRI: `https://ekgf.org/ontology/user-experience#hexColor`

RDF predicate	Value
`rdf:type`	`owl:AnnotationProperty`
`rdfs:isDefinedBy`	ekg-ux:

25.2.2 Sort key

Label: Sort key

Identifier: `ekg-ux:sortKey`

Attach a sortable string to any given subject IRI that can be used as a sort-key in UI code or report generators to sort a given set of objects (individuals). For instance two objects <john> and <doe> could have the following sortKeys: <john> ux:sortKey "0001john" . <doe> ux:sortKey "0001doe" . Resulting in them being sorted in the order <doe>, <john>. This is a primitive but often practical way to get things sorted in the right order.

IRI: `https://ekgf.org/ontology/user-experience#sortKey`

RDF predicate	Value
`rdf:type`	`owl:AnnotationProperty`
`rdfs:isDefinedBy`	ekg-ux:

EKGF Editor's Draft

A Documentation coverage

24 ontologies; 757 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	3	0
Platform API Ontology	2	0
Artifact Dependency Ontology	45	0
Artifact Governance Ontology	46	0
Artifact Publication Ontology	34	0
Authorization Ontology	4	0
Capability Ontology	5	0
Communication Ontology	141	0
Concept Ontology	58	0
DataOps Ontology	13	0
Dataset Ontology	9	0
File System Ontology	101	0
Lifecycle Ontology	23	0
Maturity Model Ontology	26	0
Outcome Ontology	25	0
Persona Ontology	5	0
Raw Data Ontology	41	0
Specification by Example Ontology	25	0
SHACL Shape Metadata Ontology	16	0
Story Ontology	41	0
SPARQL Story Implementation Ontology	45	0
Story Service Ontology	23	0
Use Case Ontology	24	0
User Experience Ontology	2	0