* Templates / Monads / Sets Interfaces
* ResourceURNs inputs (State DIDs Feedback / Matching):
* ResourceURN: (ResourceURN, Context, Attribute, Value);
* Uniform identifier across occurrences. DID URN. Endpoint. IRIs values. ResourceURN statements: uniform functional metadata (contextual type / name, relations). IDs Encodings.
* ResourceURN::getOccurrences
* Occurrence::getResourceURN
* Occurrences populated from functional traversal parsing / mapping / matching of ResourceURN statements.
* Functional DataFlow ResourceURN messages streams / events / traversal:
* ResourceURNs Bindings (aggregations)
* (ResourceURN, Resource, Occurrence, Kind) :
* ResourceURN is for Resource in Occurrence Kind. Encodings.
* (ResourceURN, Kind, Occurrence, Resource) :
* ResourceURN is for Kind in Occurrence Resource. Encodings.
* (ResourceURN, Occurrence, Resource, Kind) :
* ResourceURN is for Occurrence in Resource for Kind. Encodings.
* ResourceMonad<ResourceClass : SubjectResource, etc.>
* KindMonad<KindClass, SubjectKind, etc.> Monad
* ResourceQuadMonad<ResourceQuadClass : ISubject, etc.> Monad
* OccurrenceQuadMonad<OccurrenceQuadClass : ISubject, etc.> Monad
* Interfaces (sets / roles):
* IQuad : (IContext, ISubject, IPredicate, IObject)
* IResource : IQuad
* IContext : IResource
* ISubject : IResource
* IPredicate : IResource
* IObject : IResource
* IKind : IContext
* ISubjectKind : IKind, IPredicate, IObject
* IPredicateKind : IKind, ISubject, IObject
* IObjectKind : IKind, ISubject, IPredicate
* IStatement : ISubjectKind, IPredicateKind, IObjectKind
* IStatementKind : IKind, ISubjectKind, IPredicateKind, IObjectKind
* IMapping : ISubjectKind, IPredicateKind, IObjectKind
* IMappingKind : IKind, ISubjectKind, IPredicateKind, IObjectKind
* ITransform : ISubjectKind, IPredicateKind, IObjectKind
* ITransformKind : IKind, ISubjectKind, IPredicateKind, IObjectKind
* IState : ISubjectKind, IPredicateKind, IObjectKind
* IStateKind : IKind, ISubjectKind, IPredicateKind, IObjectKind
* IQuad : CSPO Resource Interfaces classes quad. Builder fluent API.
* (IContext, ISubject, IPredicate, IObject);
* In an occurrence (Quad) CSPO Resources URNs are wrapped in their respective Resource roles. Occurrence Monad wraps specific Quad types (CSPOs, Kinds, Statements, etc.).
* Kinds interfaces differentiate in their CSPOs return value types.
* Model Object Hierarchy: ClassName :: (aggregatingClass, subject / instance, attribute / predicate, value / object);
* SubjectResource : ISubject
* Subject : (ISubjectKind, ISubject, IPredicateKind, IObjectKind) : ISubject
* SubjectKind : (IState, ISubjectKind, IPredicate, IObject) : ISubjectKind
* PredicateResource : IPredicate
* Predicate : (IPredicateKind, ISubjectKind, IPredicate, IObjectKind) : IPredicate
* PredicateKind : (IState, ISubject, IPredicateKind, IObject) : ISubjectKind
* ObjectResource : IObject
* Object : (ISubjectKind, ISubject, IPredicateKind, IObjectKind) : IObject
* ObjectKind : (IState, ISubject, IPredicate, IObjectKind) : ISubjectKind
* StatementResource : IStatement
* Statement: State, Kind, SPOs. (IStatementKind, IPredicate, ISubject, IObject) : IStatement
* StatementKind: (IState, PK of SK, OK: Relation, SK, OK) : IStatementKind
* MappingResource : IMapping
* Mapping: (IMappingKind, IStatementKind, ISubject, IPredicate) : IMapping
* MappingKind: (IState, SK of PK, OK: Schema, PK, OK) : IMappingKind
* TransformResource : ITransform
* Transform: (ITransformKind, IMappingKind, IPredicate, IObject) : ITransform
* TransformKind: (IState, OK of SK, PK: Behavior, SK, PK);
* StateResource : IState
* State : (Relationship, Relation, Entity, Value) : DCI Model I/O.
* StateKind : (PK of SK / OK: Dataflow domain / range)
* APIs. Traversal. State aggregate Kinds, Kinds aggregate Occurrences, Occurrences aggregate Resources (roles). Perform Mappings Transforms. Layers dataflows.
* Augmentations:
* Incremental / Feedback.
* Aggregations: ResourceURNs Source IRIs Sets / Layers streams / events (Resources, Occurrences, Kinds) parse / Occurrences population.
* Alignments: Aggregation traversal: ResourceURN URN IDs Model population. Merge / Matching, order / relations / contexts. Encoding (methods).
* Activations: Relationship Models I/O (DCI Layers / expanded SPO Aggregations feedback). DIDs URN hashing / generation (HATEOAS Endpoints). Data Flow.
* Encoding:
* IDs: Resource, Occurrence, Kind.
* Functional DataFlow ResourceURN messages streams / events / traversal:
* (Resource, Occurrence, Kind)
* IDs: Count (axis / lists / SPOs aggregated), CIDs. Normalize / Order (sequence / hierarchy / graph position / relation IDs) / Augmentations.
* Quads / SPOs hierarchical list encoding.
* Graph / Tree List Parent / Child hierarchical encoding / hashing:
* (C (S (P (O).
* Dataflow (value expressions). Signatures (events subscriptions: domain / range). Encode Order.
* Order:
* StatementKind: PK(SK, OK). Relationship(Roles). Context, State, Mapping, Transform Kinds.
* SK(PK, OK)?
* OK(PK, SK)?
* Statement: abstract assertions (parsed / inferred). Mapping: abstract schema. Transform: abstract behavior. Relationships (dimensional / discrete): core model / ontology, Statement, Mapping, Transform synchronized (input / inferred Statement Events are fully parsed from CSPO Sets Layer).
* Order. States (Statements), Flows (Mappings), Events (Transforms). Kinds hierarchy tree / lattice (FCA). Action / Passion / State order. Kinds / Mappings domain / range Aggregation, Activation, Alignment. Comparisons. DCI / MVC / Relationships / Dimensional Aggregated upper onto matching gestures / flows.
* ResourceURNs DIDs:
* URN: DIDs. Endpoint APIs: Statements types / sets (Resource, Kind, Statement, Mapping, Transform OntResources hierarchy) content types / classes: Functional APIs. OntResource (DOM DTOs) quads representations references other DIDs, handle resolution, interactions, etc. via other DIDs endpoints and Resource Monad API.
* Method: did:ont:[ID]
* ID : OntClassName (Sets) ":" [HashedQuad];
* HashedQuad : [HashedURN] ":" [HashedURN] ":" [HashedURN] ":" [HashedURN];
* HashedURN : "[" HashedQuad "]" | HashedCSPOString;
* HashedCSPOString : Context ":" Subject ":" Predicate ":" Object;
* URN::ontResource (traversal parsed representation).
* OntResource::URN.
* Encoding: methods
* Hashing: four segments identifiers. Sets, binary octal digit order operable hashing (4 bit per segment). Aggregation: Statements graph layout. Occurrences. S-Expressions, MonParsec, CoSQL, map-reduce.
* URN: Encoded quad. Hashing: traversal, discovery, resolution. Merkle tree (DLT / Events). Encode typing / naming in context, about DID State Statements (hashing metadata):
* URN Typing / Naming (Resource Statement):
* URN : (Resource, Occurrence, Type, Name);
* Kinds: Aggregate Attributes.
* State: Aggregate Kinds Resources Attributes / Values.
* Hierarchy: Kinds Attributes set (super) subset (sub) Kinds relationship.
* Order. Aggregation: Kinds / States lattice / tree. Populate / encode ResourceURNs order in contexts.
* Functional Data Flow:
* Functor / Category: Resource Monad (of OntResource hierarchy). Dynamic typing DOM / DTOs Kinds members.
* Subject extends Quad<C, S, P, O> implements OntResource
* Resource<? extends Quad>
* Functional Contexts: Arrange Uniform Resource roles CSPO based interface roles Aggregation / relations dataflow traversal
* (Transform, Mapping, Statement, Kind, Occurrence, Resource);
* (Resource, Occurrence, Kind, Statement, Mapping, Transform);
* CSPOs, Kinds, States, OntResource: Uniform Resource functional domain category interface:
* getContext
* getSubject
* getPredicate
* getObject
* getOccurrences
* Resources::getKinds
* Kind::getResources
* Kind::getStates
* State::getKinds
* State::getContexts
* Context::getStates
* Context::getAttributes
* Attribute::getContexts
* Attribute::getValues
* Value::getAttributes
* getPrevious (in functional context / axis)
* getNext (in functional context / axis)
* getParent (in functional context / axis)
* getChildren (in functional context / axis)
* getEquals (in functional context / axis)
* OntResource:
* (OntResource, OntResource, OntResource, OntResource);
* Subjects : OntResource
* (SubjectKind, Subject, PredicateKind, ObjectKind);
* Predicates : OntResource
* (PredicateKind, SubjectKind, Predicate, ObjectKind);
* Objects : OntResource
* (ObjectKind, SubjectKind, PredicateKind, Object);
* SubjectKind (SK) : Subject. Predicate / Object Intersection:
* (State, SubjectKind, Predicate, Object);
* PredicateKind (PK) : Predicate. Subject / Object Intersection:
* (State, Subject, PredicateKind, Object);
* ObjectKind (OK) : Object. Predicate / Subject Intersection:
* (State, Subject, Predicate, ObjectKind);
* StatementKind:
* (State, PK of SK, OK: Relation, SK, OK);
* Statement: State, Kind, SPOs.
* (StatementKind, Statement, Subject, Object);
* MappingKind:
* (State, SK of PK, OK: Schema, PK, OK);
* Mapping: State, Kind, SPOs.
* (MappingKind, Mapping, Predicate, Object);
* TransformKind:
* (State, OK of SK, PK: Behavior, SK, PK);
* Transform: State, Kind, SPOs.
* (TransformKind, Transform, Subject, Predicate);
* State: Kinds occurences Contexts. Plain Statements I/O.
* State : Kinds, SPO:
* (Context, Resource, Attribute, Value);
* Aggregates Kind / Resources Graph on their Kinds Attributes / Predicates. Aggregate / Order Statements in hierarchical Resource Kinds axis. Order / Aggregation Function (Kinds): Resource gt Attribute gt Value. Merge SPO Aggregations.
* Context: CSPO Context. Relationship Relations.
* Resource: SPO Kinds Resources.
* Attribute: Resource Predicate. Subject: P, Predicate: S, Object: S.
* Value: Resource Object. Subject: O, Predicate: O, Object: P.
* DCI Context Model:
* DCI Relationship Contexts: Data / Schema / Behavior Model. DCI / MVC / Relationships Upper onto matching: gestures / flows.
* Metaclasses: PredicateKind SubjectKinds / ObjectKinds.
* Relationship: (Relationship, SubjectKind, PredicateKind, ObjectKind);
* PredicateKind of SK / OK. Employment(Employer, Employee); Employment (Employee, Position);
* Relation : (Relationship, Statements / Context, Role, Occurrence);
* Role : (Relation, Resource, Occurrence, Metaclass : Kinds);
* Occurrence : (Role, Relation, Context / Relation Statements, Resource);
* Relationship Aggregated Statements:
* Aggregated Statements traversal: expanded SPO form.
* Context: (Relationship : Predicate Kind, Relation : Statements, Role : Kind, Player : Resource);
* Predicate Kind of Reified S SK, O OK. (Relationship: Employment, Roles: Employee SK, Employer OK). Employment PK aggregated by Subjects and Objects Kinds. Relation Statements: Aggregated SK, PK, OK by Contexts Statement Kinds.
* (Working, workingRelationStmt, employer, IBM);
* (Working, workingRelationStmt, employee, John);
* Dimensional Relationships:
* (Dimension : Relationship, Measure : Relation, Unit : Kind, Value : Resource);
* (Time, oneHourStmt, minutes, 60);
* Distance Dimension: PK of Time SK / Meters OK. Define Dimension in terms of Relationship Kinds.
* Dimension Measure Statements: Domain PK Statements. Kind interface for Functional Transforms.
* Unit: PK Measure SK / OK Statement Kinds (SK / OK Members).
* Value: Dimension Measure Statement Kind Resource.
* Dimensional Order / Comparison: OrderKinds. Templates (populate).
* Order: Comparison. A / B < 0: A < B. A, B indexes: hierarchy tree positions.
* Proof of Concept: Achieve REST Facade (synchronized) of Relationships given inputs from a system backend:
* Inputs: Aggregate SPO into CSPO: Aggregates Contexts Type / Table / Class Kinds. Aggregate PK Cols, Cols : Occurrence, Val : Resources.
* Inputs (Rel / Graph): (Type / Table / Class, PK : Resource, Col : Occurrence, Val : Resource).
* Inputs (Rel / Graph) FKs: Val : Resource equivalent PKs.
* Features: tailor implementations for
* ESB. BPM. CMS. NLP. BI. CRM.
* Naming: Semantic Hashing.
* Naming: URNs.
* Encoding: ML Embeddings. Data: classification, Schema: clustering, Behavior: regression.
* Encoding / Matching:
* Functional Context:
* Metaclass
* Class
* Instance
* Context
* Role
* Occurrence
* Hierarchy: Roles / Primitives.
* Upper Ontology: Need, Product, Good, Purpose. Goal.
* Upper Ontologies: From Primitives to Forms / UI Gestures.
* Units of Measurement (continuos) APIs /  Ontology.
* Discrete (events) APIs / Ontology. Relationships.
* Cube Statements:
* (Fact, Axis, Measure, Value);