Graph based Meta Schema for collaborative Operations

draft-henderickx-meta-graph-schema-00

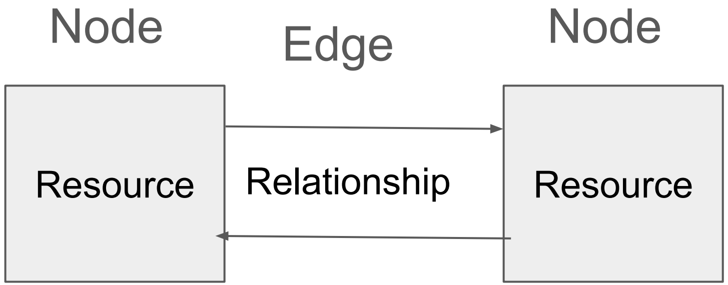
Wim Henderickx - Nokia

Goal

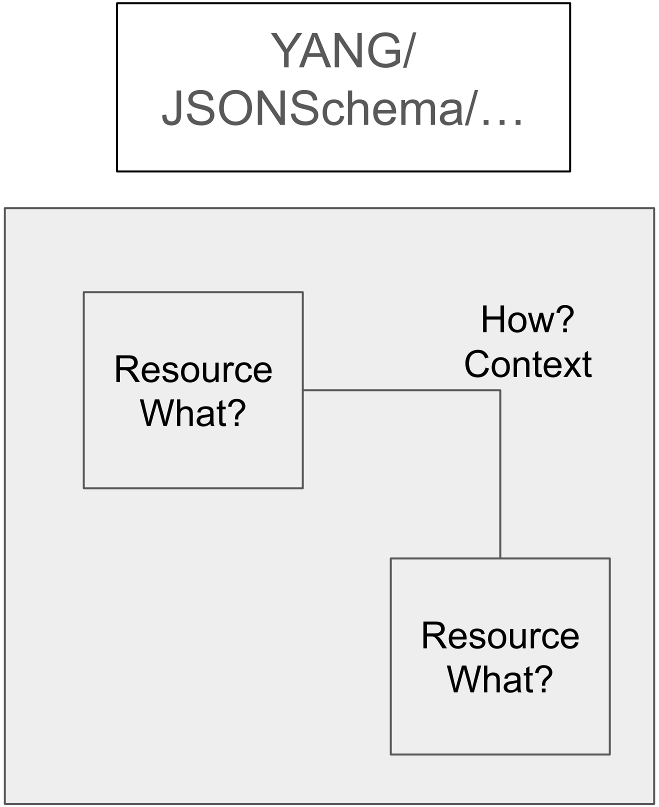
- Define a generalized mechanism for modeling resources through a graph
    - Does not replace existing modeling languages (YANG, JSONSchema, …)
    - It defines a meta-layer that integrates existing modeling languages and standardizes how resources and relations are described, linked, and managed.
- Defines a set of unified operations that assist the automation/business logic
    - Multi-actor
    - Versioning, DryRun
    - GraphQL.

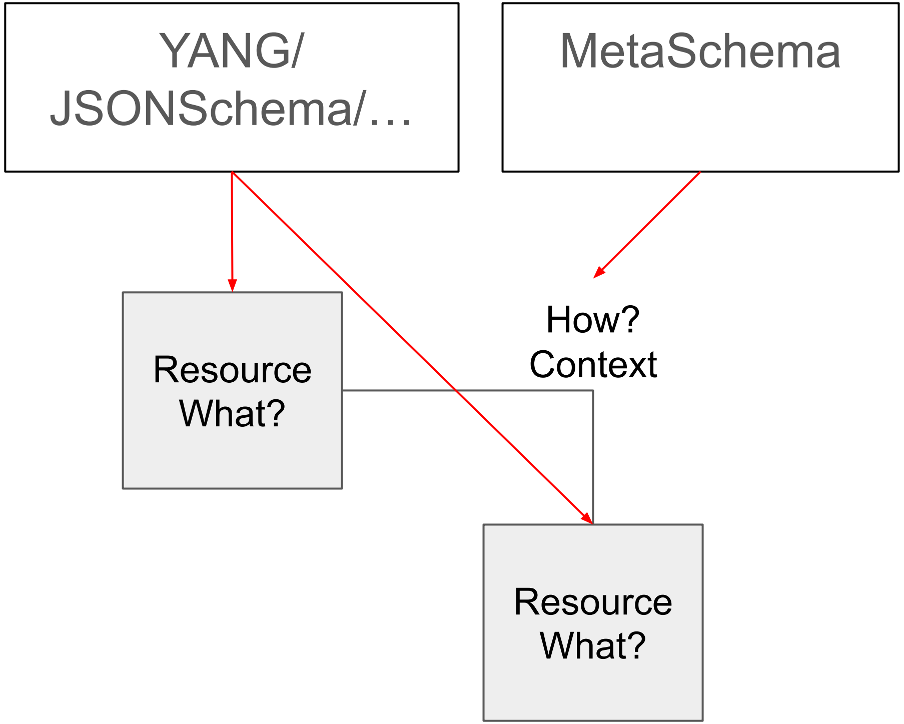
Terminology

- Resource (Node):
  - API object: YANG, openAPI, JsonSchema, etc
  - Unique identifier: Group, Version, Kind, Scope
  - Focus on the what
- Relationship (Edge):
  - connecting <u>resources</u> using the unique identifiers
  - Constraints + metadata
  - Provide context, focus on the how resources are consumed.

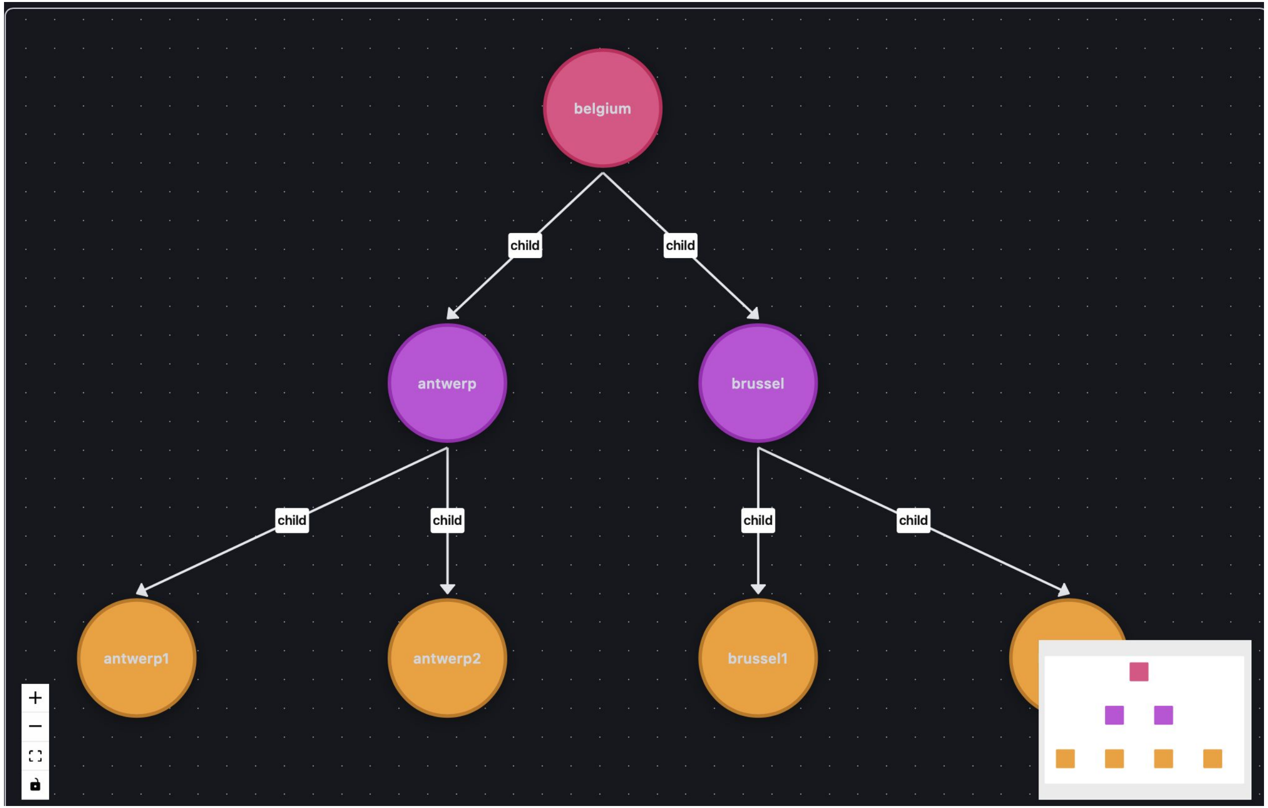


What and How


YANG/
JSONSchema/…
How?
Context
Resource
What?
Resource
What?


YANG/
JSONSchema/…
MetaSchema
How?
Context
Resource
What?
Resource
What?

Example graph



Why?

- Separation of concerns
  - Better reuse of resource models
  - Extendability
  - Simplified data management
- Generalized concept
  - Generalized lookups/query/mutate
    - GraphQL
  - Generic code in the server (offloading the client software)
    - Business logic
    - Validation
    - Binding logic
    - Name generation
    - Attribute computation

API Operations

- Meta Schema/Header provide support for multi actor operations, version control dry-run, etc
    - Conflict Management using server side apply
    - Coordinated deletion
    - Ownership (modelled as a edge)
    - Multi-tenancy
    - Bulk operation
    - Version Control
    - Dry Run
    - Streaming and eventing
    - Opaque metadata
    - graphQL

Thank You

- Questions, comments ?