Advanced Topics

SHACL Cheat Sheet

Instructor: John Placek

Date: 2022-10-05

Node Shapes

```
sp:NodeExampleShape
    a sh:NodeShape ;
    sh:targetNode dbr:Citizen_Kane ;
    sh:property [
        sh:path dbo:director ;
        sh:class dbo:Person ;
    ] .
```

Shape is an IRI or Blank Node

Instance of sh:NodeShape

Target declaration

Constraint Component

Property Path

Constraint Component

Property Shapes

```
sp:PropertyExampleShape
    a sh:PropertyShape ;
    sh:targetClass dbo:Person ;
    sh:path rdfs:label ;
    sh:minCount 1 .
```

⟹ Shape is an IRI or Blank Node

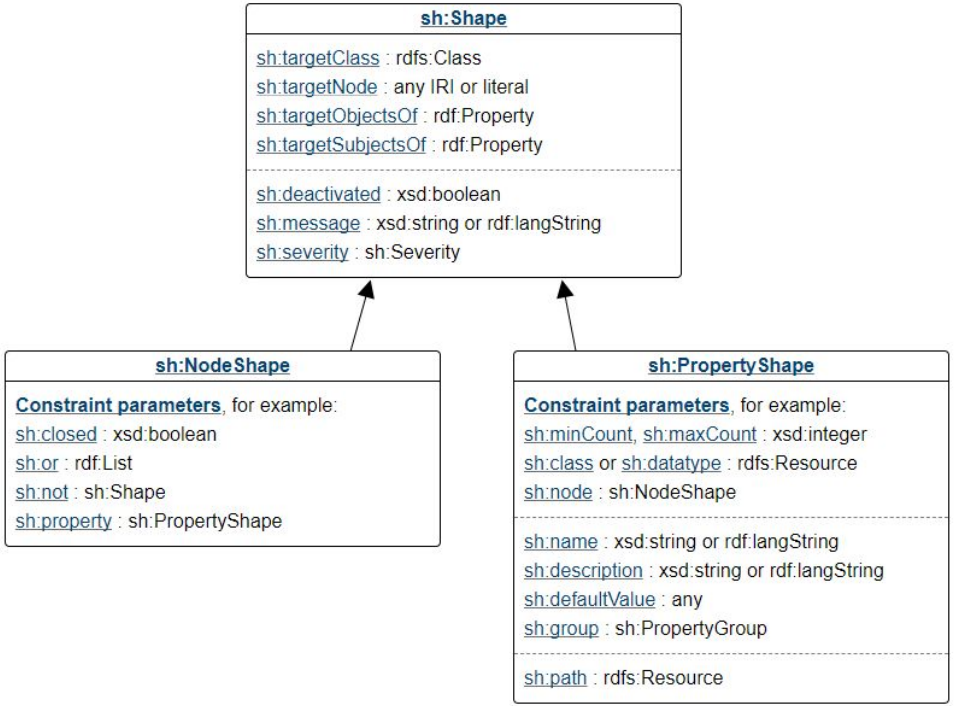
⟹ Instance of sh:PropertyShape

⟹ Target declaration

⟹ Property Path

⟹ Constraint Component

Key Classes



Value Type Constraint Components

- sh:class
  - Each value node is a SHACL instance of a given type
- sh:datatype
  - Specifies the datatype each value node should have
- sh:nodeKind
  - Specifies the RDF node kind each value node should have
    - E.g. `sh:BlankNode`, `sh:IRI`, `sh:Literal`

Cardinality Constraint Components

Cardinality Constraint Components define restrictions on the number of values for the given focus node

- sh:minCount
  - specifies the minimum number of value nodes
  - if the minimum cardinality value is 0 then this constraint is always satisfied and so may be omitted
- sh:maxCount
  - specifies the maximum number of value nodes

Value Range Constraint Components

- Specify value range conditions to be satisfied by value nodes that are comparable via operators such as <, <=, > and >=
  - sh:minExclusive
  - sh:minInclusive
  - sh:maxExclusive
  - sh:maxInclusive
- Example: Minimum and maximum age of a person

String-based Constraint Components

- sh:minLength
  - can be applied to any literals and IRIs, but not to blank nodes
- sh:maxLength
  - can be applied to any literals and IRIs, but not to blank nodes
- sh:pattern
  - specifies a regular expression that each value node should match
- sh:languageIn
  - the allowed language tags for each value node are limited by a given list of language tags
- sh:uniqueLang
  - can be set to true to specify that no pair of value nodes may use the same language tag

Property Pair Constraint Components

These constraints can only be used by property shapes:

- sh:equals
  - The sets of values of both properties at a given focus node must be equal
- sh:disjoint
  - The sets of values of both properties at a given focus node must be different
- sh:lessThan
  - The values at a given focus node must be smaller than the values of another property of the same focus node
- sh:lessThanorEquals
  - The values at a given focus node must be smaller than or equal to the values of another property of the same focus node

Logical Constraint Components

These constraints implement the common logical operators *and*, *or*, and *not*, as well as a variation of *exclusive or*.

- sh:not
  - Each value node cannot conform to a given shape
  - Comparable to negation and the logical "not" operator
- sh:and
  - Each value node conforms to all provided shapes
  - Comparable to conjunction and the logical "and" operator
- sh:or
  - Each value node conforms to at least one of the provided shapes
  - Comparable to disjunction and the logical "or" operator
- sh:xone
  - Each value node conforms to exactly one of the provided shapes

Shape-based Constraint Components

- sh:node
  - Each value node conforms to the given node shape
- sh:property
  - Each value node has a given property shape
- sh:qualifiedValueShape, sh:qualifiedMinCount, sh:qualifiedMaxCount
  - A specified number of value nodes conforms to the given shape
  - Each `sh:qualifiedValueShape` can have: one value for `sh:qualifiedMinCount`, one value for `sh:qualifiedMaxCount` or, one value for each, at the same subject

Other Constraint Components

- sh:closed
  - can be used to specify the condition that each value node has values only for those properties that have been explicitly enumerated
  - Set sh:closed to true to close the shape
- sh:ignoredProperties
  - a SHACL list of properties that are also permitted in addition to those explicitly enumerated
- sh:hasValue
  - specifies the condition that at least one value node is equal to the given RDF term
- sh:in
  - specifies the condition that each value node is a member of a provided SHACL list