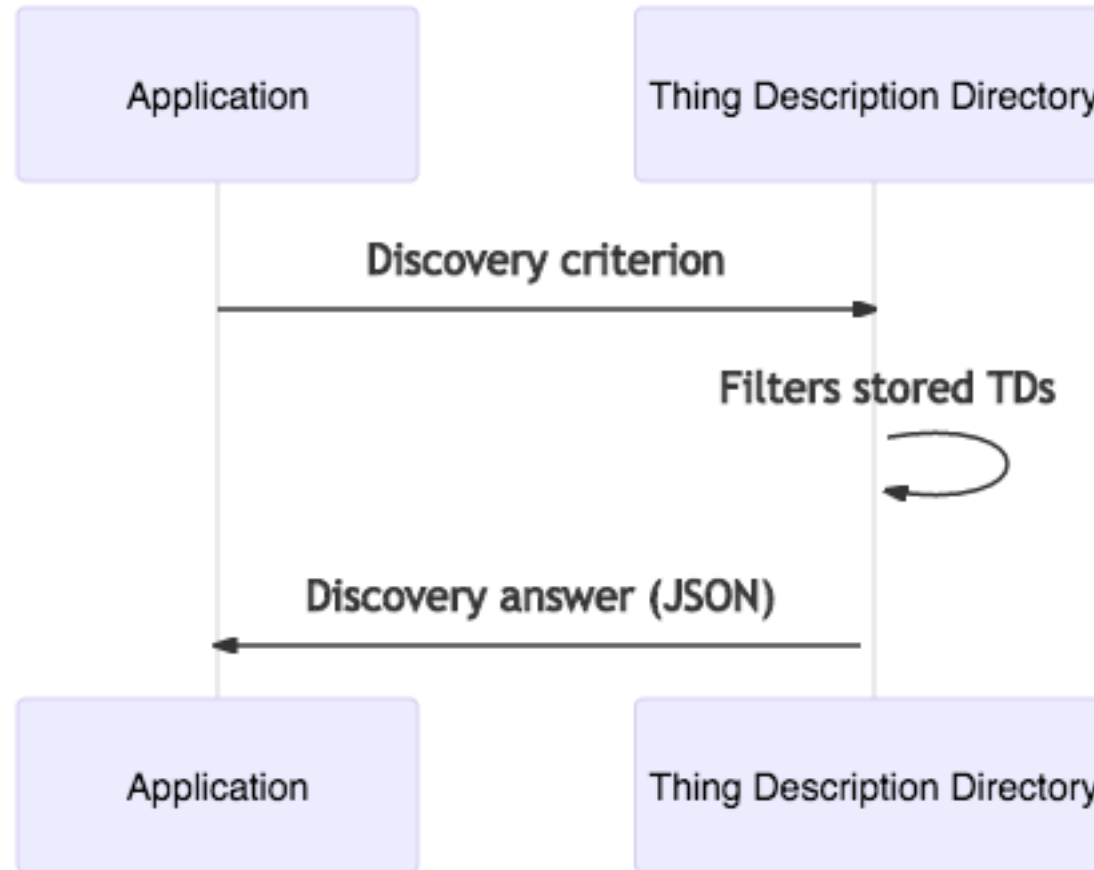


Thing Description Directory Semantic Discovery Search

Andrea Cimmino

Oct 20, 2020

Discovery in a nutshell



Thing Description Directory (TDD) and discovery

- TDD syntactic discovery criterion:
 - (Syntactic discovery) MUST support JSONPath as discovery criterion
 - (Syntactic discovery) SHOULD support XPath as discovery criterion
- TDD semantic discovery criterion:
 - (Semantic discovery) May support SPARQL as discovery criterion
- TDD discovery answer:
 - A set (array) of Thing Descriptions (TD) meeting the criterion are found
 - A set (array) of Thing Descriptions (TD) fragments fulfilling the criterion are found
 - Discovery results could be paginated (depending on the discovery criterion)

SPARQL semantic discovery API

- The discovery criterion is a SPARQL query
 - <https://www.w3.org/TR/rdf-sparql-query/>
 - W3C standard
 - All SPARQL queries should use content-negotiation to return JSON-LD or JSON
 - Supported: SELECT, ASK, CONSTRUCT, and DESCRIBE
 - Not-Supported: UPDATE
- Response code:
 - 200 (Ok) with application/json Content-Type header
- Error codes:
 - 400 (Bad Request): SPARQL query not provided or contains syntax errors.
 - 401 (Unauthorized): No authentication.
 - 403 (Forbidden): Insufficient rights to the resource.
 - 501 (Not Implemented): SPARQL API not supported.

SPARQL queries

```
PREFIX wot: <https://www.w3.org/2019/wot/td#>  
PREFIX sch: <https://www.w3.org/2019/wot/json-schema#>  
PREFIX hyp: <https://www.w3.org/2019/wot/hypermedia#>
```



```
SELECT DISTINCT ?href WHERE {  
  ?device wot:hasPropertyAffordance ?property .  
  ?property sch:propertyName "status" .  
  ?property wot:hasForm ?form .  
  ?form hyp:hasTarget ?href .  
}
```

Find the href of devices that have the property status

- XPath: `*//properties/status//href`

SPARQL queries

```
PREFIX wot: <https://www.w3.org/2019/wot/td#>
PREFIX sch: <https://www.w3.org/2019/wot/json-schema#>
PREFIX hyp: <https://www.w3.org/2019/wot/hypermedia#>
```

```
SELECT DISTINCT ?href WHERE {
  ?device wot:hasPropertyAffordance ?property .
  ?property sch:propertyName "status" .
  ?property wot:hasForm ?form .
  ?form hyp:hasTarget ?href .
}
```



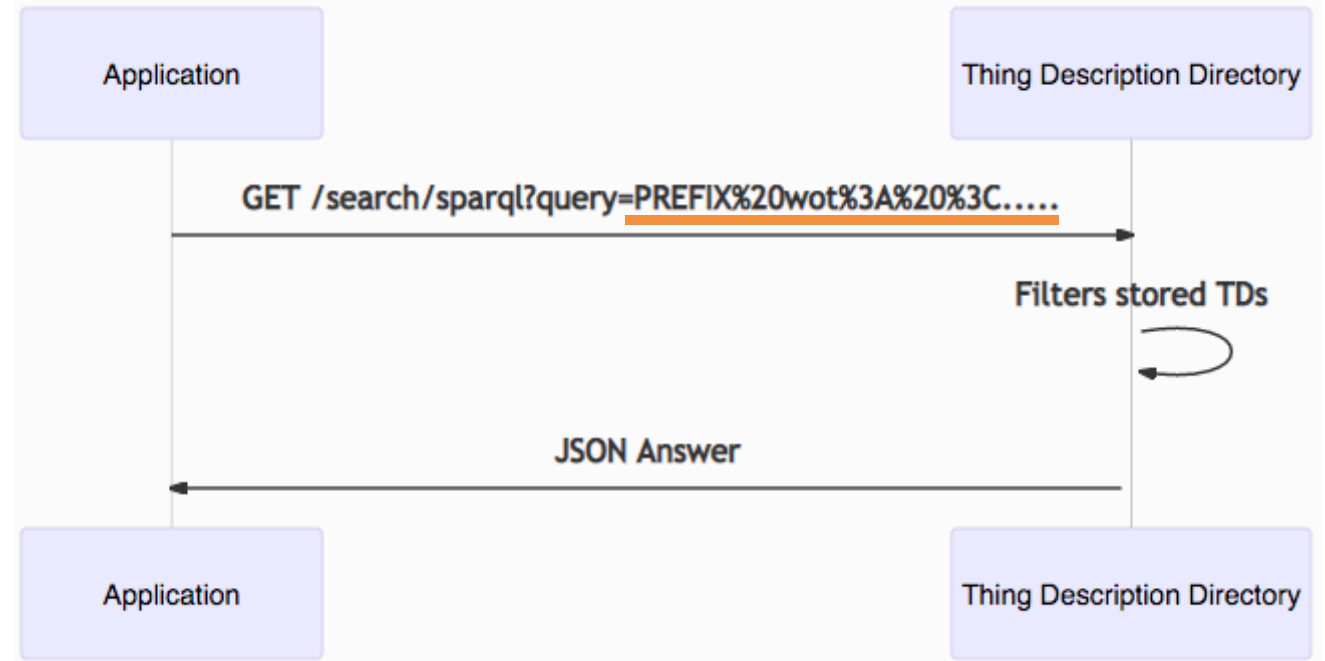
```
PREFIX%20wot%3A%20%3Chttps%3A%2F%2Fwww.w3.org%2F2019%2F
wot%2Ftd%23%3E%0APREFIX%20sch%3A%20%3Chttps%3A%2F%2Fww
w.w3.org%2F2019%2Fwot%2Fjson-
schema%23%3E%0APREFIX%20hyp%3A%20%3Chttps%3A%2F%2Fwww.
w3.org%2F2019%2Fwot%2Fhypermedia%23%3E%0A%0ASELECT%20DIS
TINCT%20%3Fhref%20WHERE%20%7B%0A%3Fdevice%20wot%3AhasPr
opertyAffordance%20%3Fproperty%20.%0A%3Fproperty%20sch%3Aprop
ertyName%20%22status%20%22%20.%0A%3Fproperty%20wot%3Ahas
asForm%20%3Fform%20.%0A%3Fform%20hyp%3AhasTarget%20%3Fhre
f%20.%0A%7D%0A%00
```

SPARQL semantic discovery API example

```
"searchSPARQL": {
  "description": "SPARQL semantic search",
  "uriVariables": {
    "query": {
      "title": "A valid SPARQL 1.1. query",
      "type": "string",
      "format": "iri-reference"
    }
  },
  "forms": [
    {
      "href": "/search/sparql?query={query}",
      "htv:methodName": "GET",
      "response": {
        "description": "Success response",
        "htv:statusCodeValue": 200
      },
      "scopes": "search"
    },
    {
      "href": "/search/sparql",
      "htv:methodName": "POST",
      "response": {
        "description": "Success response",
        "htv:statusCodeValue": 200
      },
      "scopes": "search"
    }
  ]
}
```

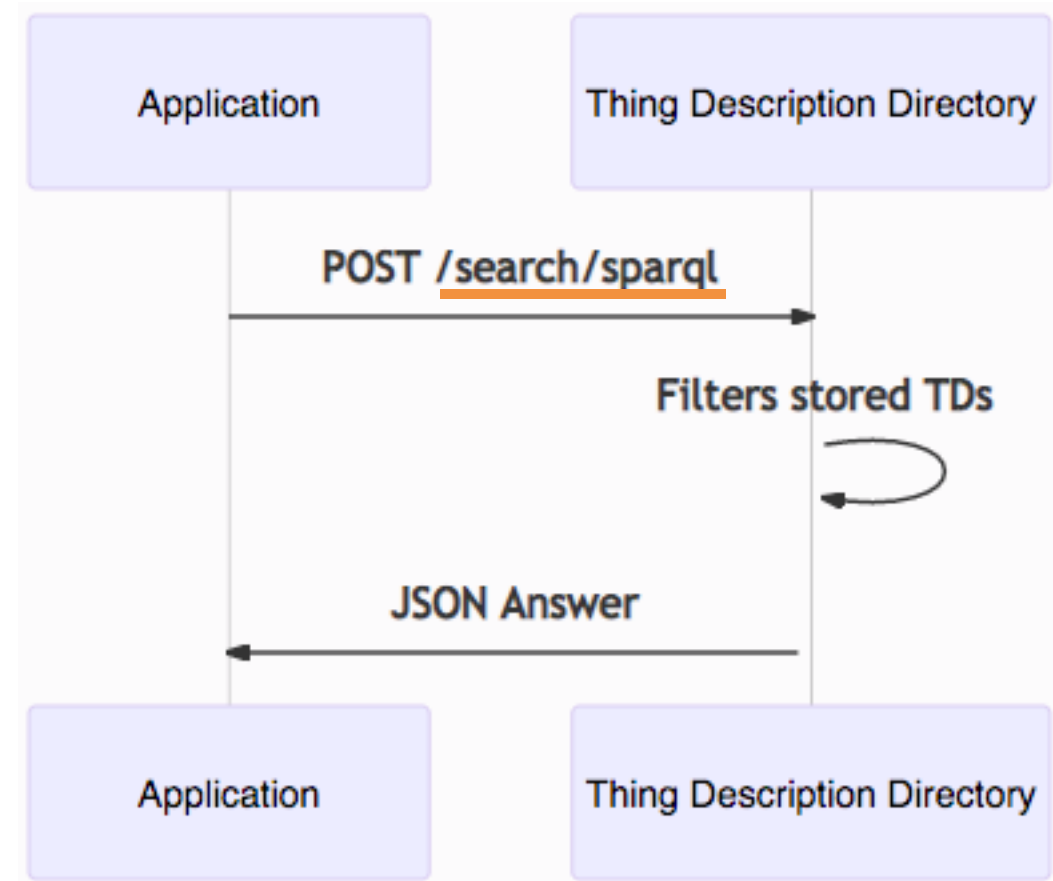
SPARQL semantic discovery API example

```
"searchSPARQL": {
  "description": "SPARQL semantic search",
  "uriVariables": {
    "query": {
      "title": "A valid SPARQL 1.1. query",
      "type": "string",
      "format": "iri-reference"
    }
  },
  "forms": [
    {
      "href": "/search/sparql?query={query}",
      "htv:methodName": "GET",
      "response": {
        "description": "Success response",
        "htv:statusCodeValue": 200
      },
      "scopes": "search"
    },
    {
      "href": "/search/sparql",
      "htv:methodName": "POST",
      "response": {
        "description": "Success response",
        "htv:statusCodeValue": 200
      },
      "scopes": "search"
    }
  ]
}
```

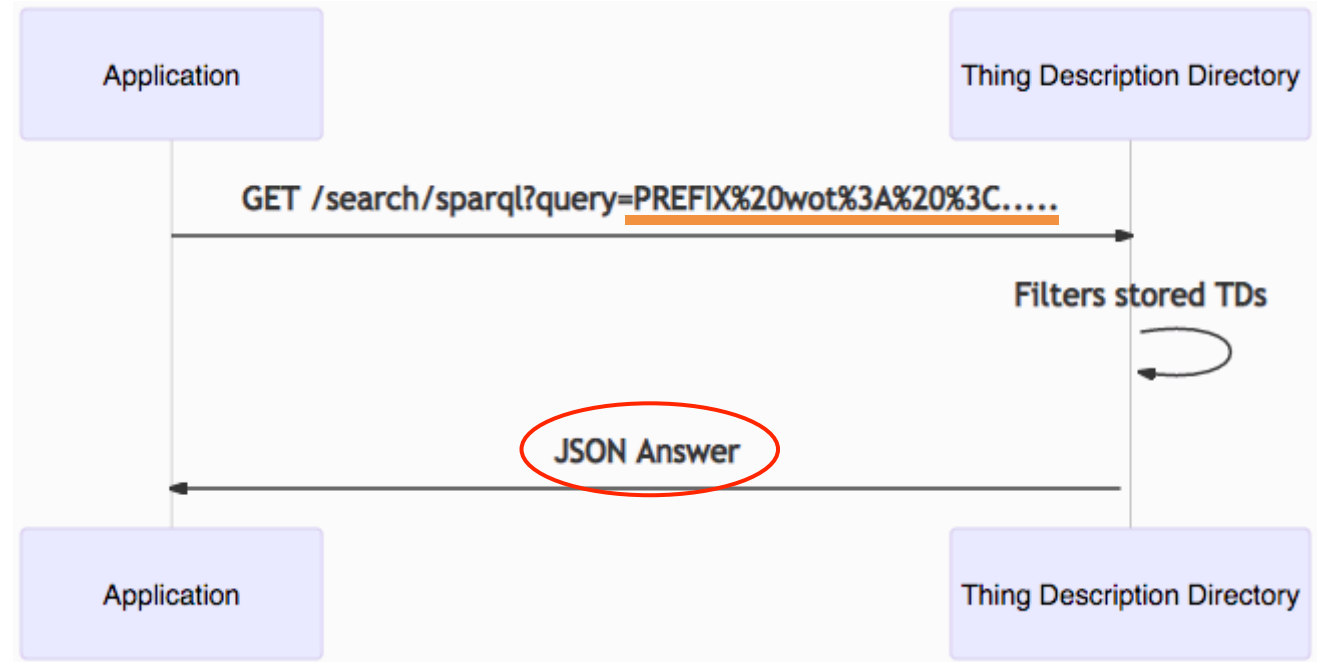
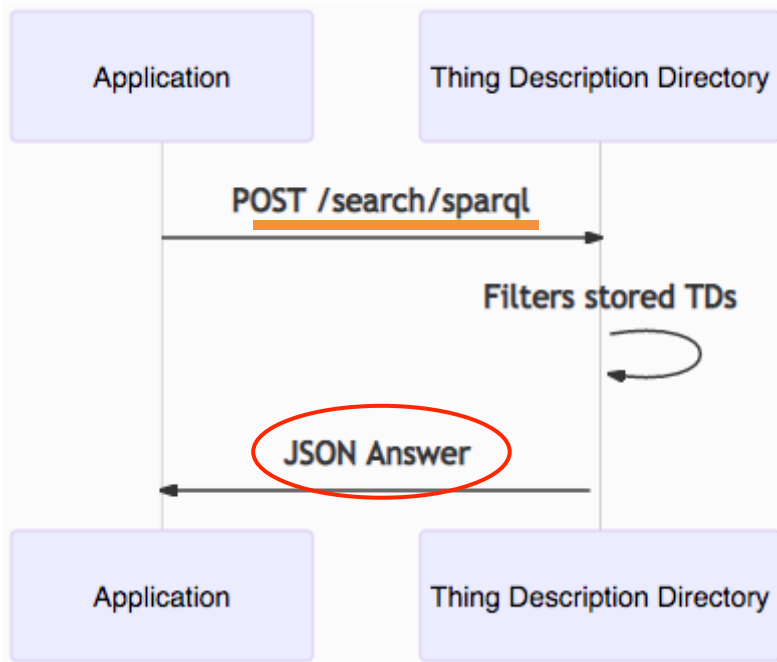


SPARQL semantic discovery API example

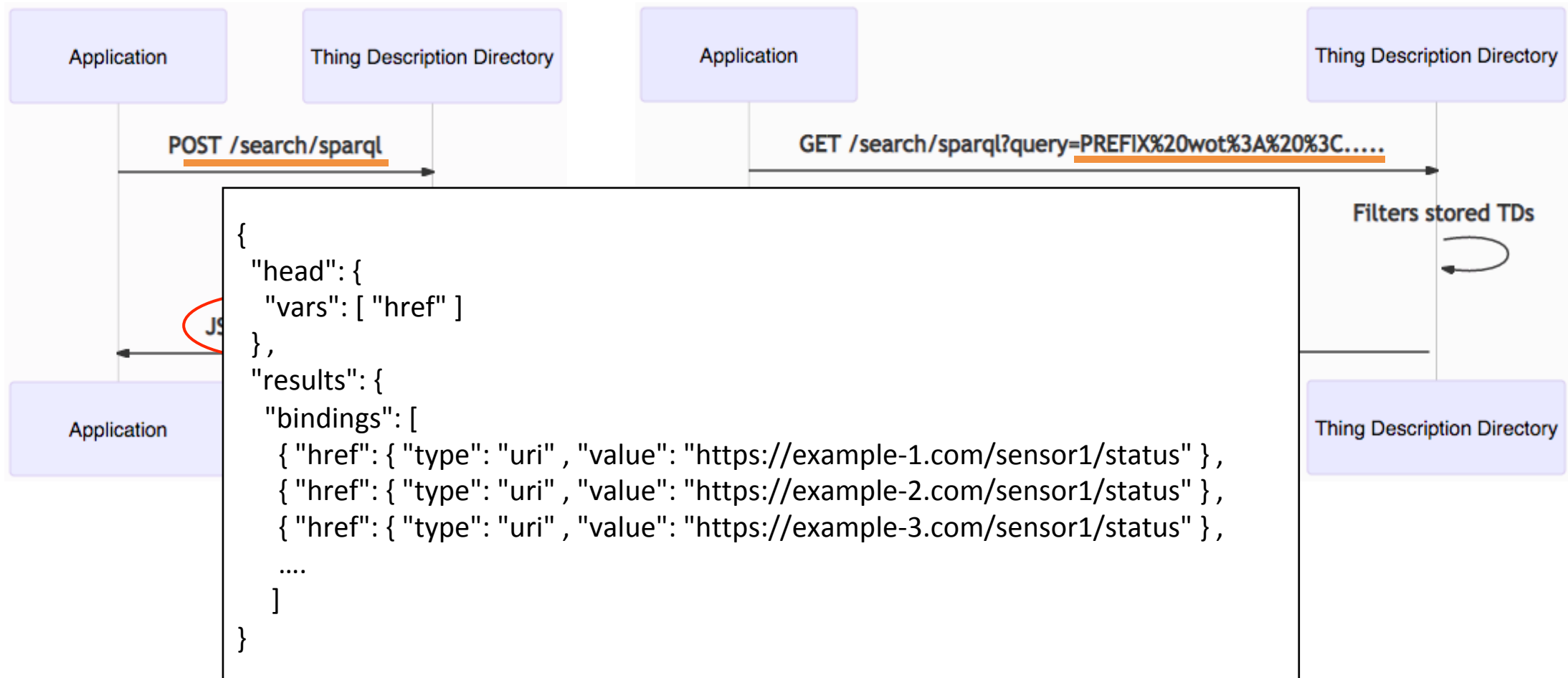
```
"searchSPARQL": {
  "description": "SPARQL semantic search",
  "uriVariables": {
    "query": {
      "title": "A valid SPARQL 1.1. query",
      "type": "string",
      "format": "iri-reference"
    }
  },
  "forms": [
    {
      "href": "/search/sparql?query={query}",
      "htv:methodName": "GET",
      "response": {
        "description": "Success response",
        "htv:statusCodeValue": 200
      },
      "scopes": "search"
    },
    {
      "href": "/search/sparql",
      "htv:methodName": "POST",
      "response": {
        "description": "Success response",
        "htv:statusCodeValue": 200
      },
      "scopes": "search"
    }
  ]
}
```



SPARQL semantic discovery API example



SPARQL semantic discovery API example



SPARQL queries examples: SELECT

- Useful for filtering TDs
 - filtering and exploratory queries
- As result return a JSON

```
PREFIX wot: <https://www.w3.org/2019/wot/td#>
PREFIX sch: <https://www.w3.org/2019/wot/json-schema#>
PREFIX hyp: <https://www.w3.org/2019/wot/hypermedia#>

ASK {
  ?device wot:hasPropertyAffordance ?property .
  ?property sch:propertyName "status" .
  ?property wot:hasForm ?form .
  ?form hyp:hasTarget ?href .
}
```

SPARQL queries examples: ASK

- Useful to know if a TD with some restrictions exists
- As result returns a boolean value inside a JSON

```
PREFIX wot: <https://www.w3.org/2019/wot/td#>
PREFIX sch: <https://www.w3.org/2019/wot/json-schema#>
PREFIX hyp: <https://www.w3.org/2019/wot/hypermedia#>

DESCRIBE DISTINCT ?href WHERE {
    ?device wot:hasPropertyAffordance ?property .
    ?property sch:propertyName "status" .
    ?property wot:hasForm ?form .
    ?form hyp:hasTarget ?href .
}
```

SPARQL queries examples: DESCRIBE

- Useful to retrieve TDs fulfilling some restrictions
- As result returns RDF → JSON-LD

```
PREFIX wot: <https://www.w3.org/2019/wot/td#>
```

```
DESCRIBE ?thing {  
  ?thing a wot:Thing .  
}
```

```
PREFIX wot: <https://www.w3.org/2019/wot/td#>  
PREFIX dc: <http://purl.org/dc/terms/>
```

```
DESCRIBE ?thing {  
  ?thing a wot:Thing .  
  ?thing dc:title "Smart-Coffee-Machine"@en .  
}
```

SPARQL queries examples: CONSTRUCT

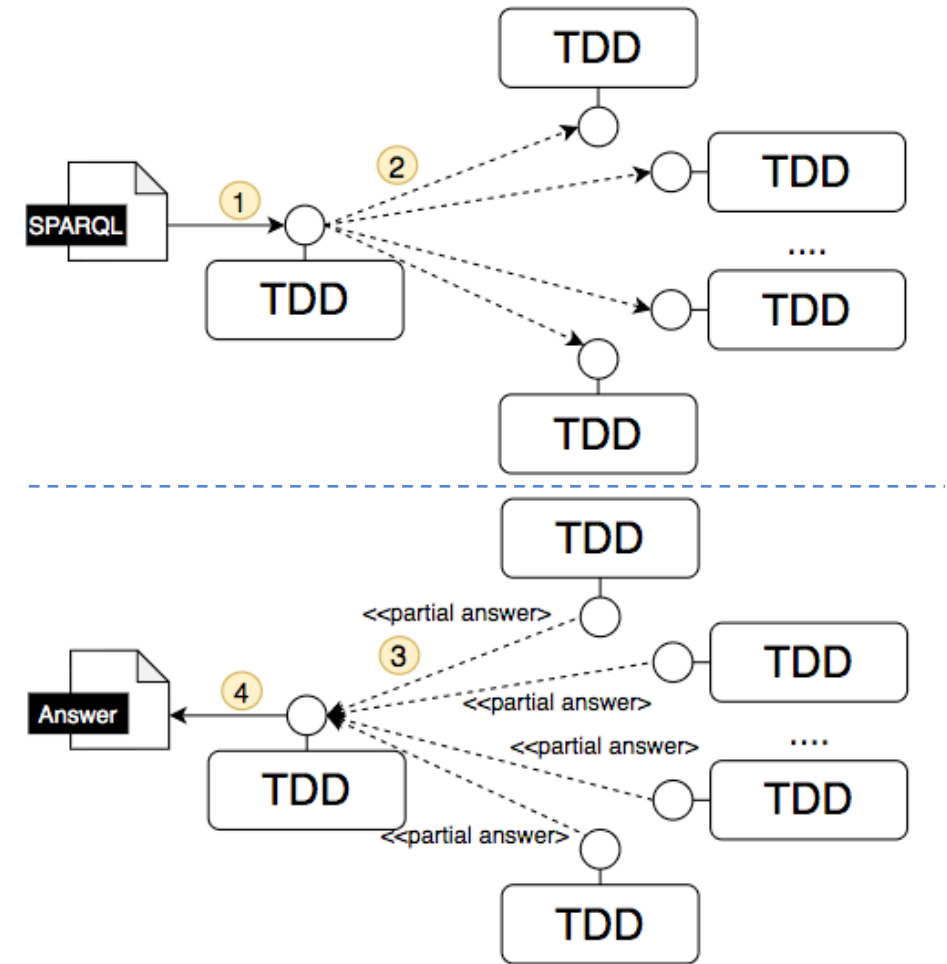
- Not very useful for discovery
 - Generates RDF from a set of restrictions and allows to introduce new triples
 - Does not modifies the original stored TD
- As result returns RDF → JSON-LD

```
PREFIX wot: <https://www.w3.org/2019/wot/td#>
PREFIX dc: <http://purl.org/dc/terms/>

CONSTRUCT {
    ?thing dc:title "new property".
    ?thing ?p ?o .
} WHERE {
    ?thing a wot:Thing .
    ?thing ?p ?o .
}
```

Advanced SPARQL queries

- Query Federation
 - SPARQL allows query federation
 - TDD forwards the query to other TDDs
 - Former TDD Returns a unified query answer
- Pros
 - Allows extending discovery criterion to other TDDs easily
 - Extends the discovery scope
- Cons
 - TDDs addresses must be known beforehand
 - Previous service discovery
 - Brute force forwarding



Advanced SPARQL queries example

- Assuming the following TDDs:
 - <http://tdd-example-1.org>,
 - <http://tdd-example-2.org>,
 - <http://tdd-example-3.org>,
 - ...
 - <http://tdd-example-100.org>

```
PREFIX wot: <https://www.w3.org/2019/wot/td#>
PREFIX sch: <https://www.w3.org/2019/wot/json-schema#>
PREFIX hyp: <https://www.w3.org/2019/wot/hypermedia#>

SELECT DISTINCT ?href WHERE {
  SERVICE ?tdd {
    ?device wot:hasPropertyAffordance ?property .
    ?property sch:propertyName "status" .
    ?property wot:hasForm ?form .
    ?form hyp:hasTarget ?href .
  } VALUES ?tdd {
    <http://tdd-example-1.org>,
    ....,
    <http://tdd-example-100.org>
  }
}
```

Pros and Cons of SPARQL

- Pros

- Expressive, takes advantages of ontology semantics and linked data
 - E.g., reasoning over data
 - E.g., linked Thing Descriptions → distributed TDs
- SPARQL is a query language
 - Has functions, e.g., for aggregation or data cleaning
 - Allows complex queries
 - Query federation
- SPARQL is a W3C standard

- Cons

- Simple queries are more verbose than those expressed in JSONPath or Xpath
- Consumes more resources than XPath and JSONPath?
- SPARQL can be complex and not easy to learn

Conclusion

- Semantic discovery allows to filter registered TDs using:
 - (MAY) SPARQL queries
- Semantic discovery (SPARQL) is very flexible:
 - Allows complex queries with functions
 - Exploits the semantics of ontologies, reasoning
 - Implementing SPARQL 1.1 enables query federation
- Privacy & Security
 - Discovery is binded to security and privacy policies defined in a TDD