OGC SensorThings API

The new standard for collecting and managing sensor data

Dr. Hylke van der Schaaf

Fraunhofer
IOSB

What is the OGC SensorThings API?

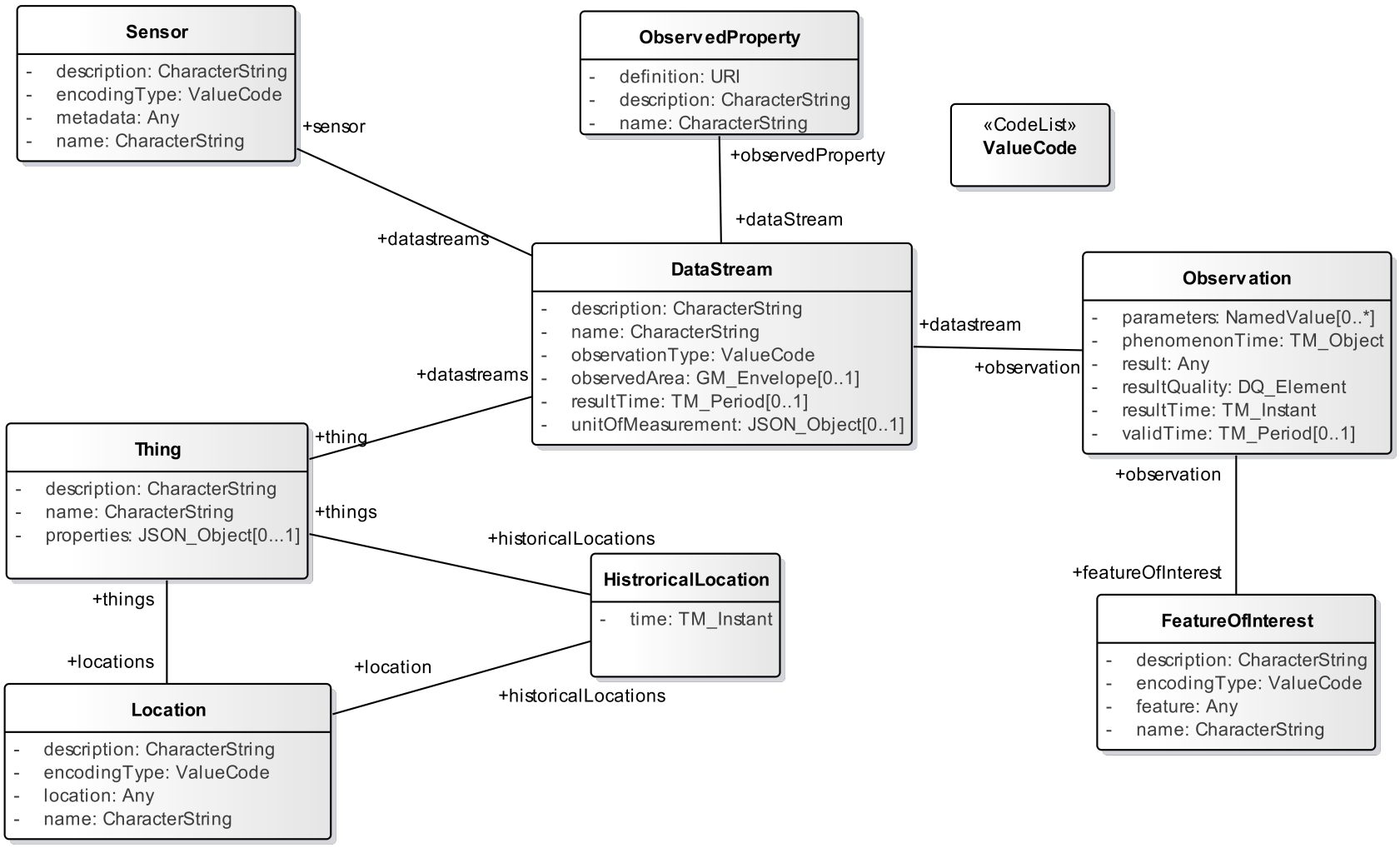
A standard for exchanging sensor data and metadata.

- Sensor-Web-Enablement-Light: SWE for IoT
- Like Sensor Observation Service (SOS), but:
  - RESTful
  - Using JSON
  - Adapting OASIS Odata URL patterns and query options
  - Supporting ISO MQTT messaging

How does the SensorThings API work?

- Part 1: Data model
  - What does the data look like?
  - Things, Locations, Datastreams, Sensors, ObservedProperties, Observations, FeaturesOfInterest
- Part 2: URL patterns for queries
  - How do I get to the data?

Data model



Fraunhofer IOSB

Query URL patterns

- Base URL: http://server.de/SensorThingsService/v1.0
- GET
  - V1.0 → Get collection index
  - v1.0/Collection → Get all entities in a collection
  - v1.0/Collection(id) → Get one entity from a collection
- POST
  - v1.0/Collection → Create a new entity
- PATCH
  - v1.0/Collection(id) → Update an entity
- PUT
  - v1.0/Collection(id) → Replace an entity
- DELETE
  - v1.0/Collection(id) → Remove an entity

Query URL patterns

GET Index

- http://server.de/SensorThingsService/v1.0
- Response:

```
{
  "value" : [
    {
      "name" : "Datastreams",
      "url" : "http://server.de/SensorThingsService/v1.0/Datastreams"
    },
    {
      "name" : "FeaturesOfInterest",
      "url" : "http://server.de/SensorThingsService/v1.0/FeaturesOfInterest"
    },
    {
    …
    …
    },
    {
      "name" : "Things",
      "url" : "http://server.de/SensorThingsService/v1.0/Things"
    }
  ]
}
```

Query URL patterns

GET all Things

- http://server.de/SensorThingsService/v1.0/Things
- Response:


```
{
  "value" : [
    {
      "name" : "My camping lantern",
      "description" : "camping lantern",
      "properties" : {
        "property1" : "it's waterproof",
        "property2" : "it glows in the dark"
      },
      "Locations@iot.navigationLink" : "Things(1)/Locations",
      "HistoricalLocations@iot.navigationLink": "Things(1)/HistoricalLocations",
      "Datastreams@iot.navigationLink" : "Things(1)/Datastreams",
      "@iot.id" : 1,
      "@iot.selfLink" : "/SensorThingsService/v1.0/Things(1)"
    },
    {
      a second thing…
    }, { … }, { … }, { … }
  ]
}
```

Query URL patterns

GET a specific Thing

- http://server.de/SensorThingsService/v1.0/Things(1)
- Response:


```
{
  "name" : "My camping lantern",
  "description" : "camping lantern",
  "properties" : {
    "property1" : "it's waterproof",
    "property2" : "it glows in the dark"
  },
  "Locations@iot.navigationLink" : "Things(1)/Locations",
  "HistoricalLocations@iot.navigationLink" : "Things(1)/HistoricalLocations",
  "Datastreams@iot.navigationLink" : "Things(1)/Datastreams",
  "@iot.id" : 1,
  "@iot.selfLink" : "/SensorThingsService/v1.0/Things(1)"
}
```

Query URL patterns

GET all Datastreams of a specific Thing

- http://server.de/SensorThingsService/v1.0/Things(1)/Datastreams
- Response:

```
{
  "value" : [
    {…},
    {…},
    {…}
  ]
}
```

Query URL patterns: $top, $skip, $count

GET only 4 Observations and the total count of Observations

- /v1.0/Observations?
  $top=4&
  $count=true
- Response:

```
{
  "@iot.count" : 16,
  "@iot.nextLink" : "/SensorThingsService/v1.0/Observations?$top=4&$skip=4",
  "value" : [
    { … },
    { … },
    { … },
    { … }
  ]
}
```

Query URL patterns: $select

GET only description und id for all Things

- http://server.de/SensorThingsService/v1.0/Things?
  $select=@iot.id,description
- Response:

```
{
  "value" : [
    {
      "description" : "camping lantern",
      "@iot.id" : 1
    },
    {
      "description" : "camping stove",
      "@iot.id" : 2
    }
  ]
}
```

Query URL patterns: $orderby

GET all Observations sorted by phenomenonTime, newest first

- /v1.0/Observations?
  $orderby=phenomenonTime desc

Query URL patterns: $filter

GET only Observations with result (value) > 5

- /v1.0/Observations?
  $filter=result gt 5
- Response:

```
{
  "@iot.count" : 8,
  "@iot.nextLink" : "/v1.0/Observations?$filter=result gt 5&$top=4&$skip=4",
  "value" : [
    {
      "phenomenonTime" : "2016-06-22T13:21:31.144Z",
      "resultTime" : null,
      "result" : 10,
      "@iot.id" : 34,
      "@iot.selfLink" : "/SensorThingsService/v1.0/Observations(34)"
    }, {
      …
    }, {
      …
    }, {
      …
    }
  ]
}
```

Query URL patterns: $expand

GET only description, id and all Datastreams for the Thing with id=17

- /v1.0/Things(17)?
  $select=@iot.id,description&
  $expand=Datastreams
- Response:

```
{
  "description" : "camping lantern",
  "@iot.id" : 17,
  "Datastreams" : [
    { … },
    { … },
    { … }
  ]
}
```

Query URL patterns: $expand(…)

GET only description, id and Datastreams for Thing 17 and for the Datastreams only id and description:

- /v1.0/Things(17)?
  $select=@iot.id,description&
  $expand=Datastreams($select=@iot.id,description)
- Response:

```
{
  "description" : "camping lantern",
  "@iot.id" : 17,
  "Datastreams" : [
    {
      "description" : "Temperature measurement",
      "@iot.id" : 19
    },
    {
      "description" : "Humidity measurement",
      "@iot.id" : 21
    }
  ]
}
```

Query URL patterns: Functions

- Comparison:
  - gt: >
  - ge: >=
  - eq: =
  - le: <=
  - lt: <
  - ne: !=
- Logical:
  - and
  - or
  - not
- Mathematical:
  - add
  - sub
  - mul
  - div
  - mod
- String Functions:
  - substringof(p0, p1)
  - endswith(p0, p1)
  - startswith(p0, p1)
  - substring(p0, p1)
  - indexof(p0, p1)
  - length(p0)
  - tolower(p0)
  - toupper(p0)
  - trim(p0)
  - concat(p0, p1)
- Mathematical:
  - round(n1)
  - floor(n1)
  - ceiling(n1)

Query URL patterns: Functions

- Geospatial:
  - geo.intersects(g1, g2)
  - geo.length(l1)
  - geo.distance(g1, g2)
  - st_equals(g1, g2)
  - st_disjoint(g1, g2)
  - st_touches(g1, g2)
  - st_within(g1, g2)
  - st_overlaps(g1, g2)
  - st_crosses(g1, g2)
  - st_intersects(g1, g2)
  - st_contains(g1, g2)
  - st_relate(g1, g2)
- Date and Time:
  - now()
  - mindatetime()
  - maxdatetime()
  - date(t1)
  - time(t1)
  - year(t1)
  - month(t1)
  - day(t1)
  - hour(t1)
  - minute(t1)
  - second(t1)
  - fractionalseconds(t1)
  - totaloffsetminutes(t1)

Summary: Query URL patterns

- GET
  - v1.0
  - v1.0/Collection
  - v1.0/Collection(id)
- Options
  - $top
  - $skip
  - $count
  - $orderby
  - $select
  - $expand
  - $filter

Creating new objects

POST a new Thing

- http://server.de/SensorThingsService/v1.0/Things

```
{
  "name" : "MyLantern1",
  "description" : "camping lantern",
  "properties" : {
    "property1" : "it's waterproof",
    "property2" : "it glows in the dark"
  }
}
```

Creating new objects

POST a new Observation

- http://server.de/SensorThingsService/v1.0/Observations

```
{
  "result" : 123,
  "Datastream" : {
    "@iot.id" : 1
  }
}
```

POST a new Observation

- http://server.de/SensorThingsService/v1.0/Datastream(1)/Observations

```
{
  "result" : 123
}
```

- phenomenonTime and FeatureOfInterest are generated automatically if not provided.

Creating new objects

POST a new Thing with a new Location

- http://server.de/SensorThingsService/v1.0/Things

```
{
  "name" : "lantern 1",
  "description" : "camping lantern",
  "properties" : {
    "property1" : "it's waterproof",
    "property2" : "it glows in the dark"
  }
  "Locations" : [
    {
      "name" : "Backyard",
      "description" : "my backyard",
      "encodingType" : "application/vnd.geo+json",
      "location" : {
        "type":"Point",
        "coordinates":[-117.123,54.123]
      }
    }
  ]
}
```

Creates both a new Thing and a new Location and links the Thing to the Location

Changing objects

PATCH on an existing Thing

- http://server.de/SensorThingsService/v1.0/Things(1)

```
{
  "description" : "A new description"
}
```

Replaces only the specified fields

PUT on an existing Thing

- http://server.de/SensorThingsService/v1.0/Things(1)

```
{
  "name" : "The old name",
  "description" : "A new description"
}
```

Replaces all fields.
Fields that are not set are removed (properties in this case)!

Deleting objects

DELETE on an existing object

- http://server.de/SensorThingsService/v1.0/Things(1)
- Deletes the Thing and all objects depending on the thing
  - Datastreams
    - Observations

Summary: Create / Update / Delete

- POST on a Collection
- PATCH on an Object
- PUT on an Object
- DELETE on an Object

Extensions

- Batch Processing
  - Multiple actions in 1 request
- MultiDataSteam
  - DataStream with multiple ObservedProperties
  - Observations with multiple result values
- MQTT
  - Receive a message when an object is created or updated
  - Create objects using MQTT messages

Future

- Conformance Test Suite
  - At the moment only 3 of 8 parts implemented
- Actuators
  - SensorThings API Part II: Tasking Profile

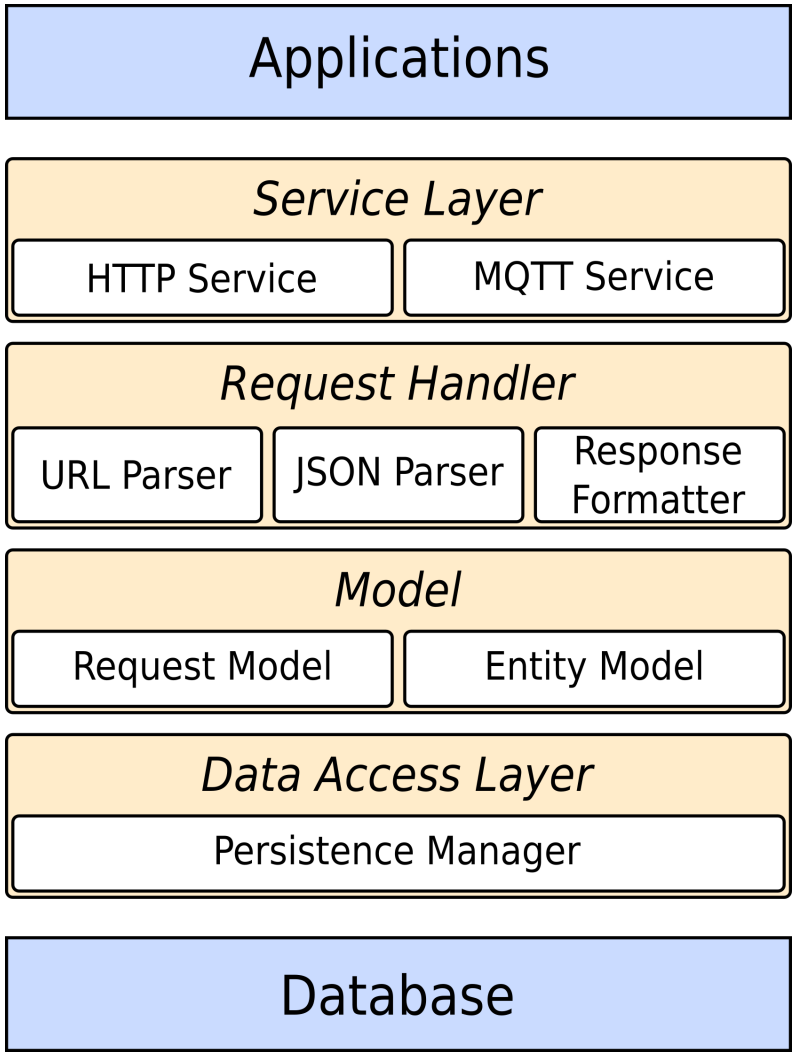
The Fraunhofer IOSB Sensor Things Server

The first and most compliant Open Source Implementation of the

- OGC SensorThingsAPI
  - https://github.com/FraunhoferIOSB/SensorThingsServer (sources)
  - http://akme-a3.iosb.fraunhofer.de/SensorThingsService (demo)
- Features:
  - GET
    - All standard paths
    - Navigating to sub-properties of complex properties
  - POST, PUT, PATCH
    - String and Numeric observation results. Json type is exactly retained
    - In-line objects, nested as deep as you want
    - GeoJSON geospatial objects for Location and FeatureOfInterest.
  - DELETE
    - Data integrity is maintained.

The Fraunhofer IOSB Sensor Things Server

- Java + Maven + Tomcat
- PostgreSQL + PostGIS



Finally

- Official website:
  - https://github.com/opengeospatial/sensorthings
- Fraunhofer IOSB implementation:
  - https://github.com/FraunhoferIOSB/SensorThingsServer