FROST-Server Cheat-Sheet

Inhalt

Über den FROST-Server

Der FROST-Server (FRaunhofer Opensource SensorThings-Server) ist eine Open-Source-Software, die entwickelt wurde, um Sensordaten über das Internet der Dinge (IoT) bereitzustellen. Es ermöglicht die Speicherung, Abfrage und Bereitstellung von Daten aus verschiedenen Sensoren und Geräten in einer standardisierten Weise. Der FROST-Server basiert auf dem SensorThings API-Standard, der vom Open Geospatial Consortium (OGC) definiert wurde. Mit diesem Server können Entwickler leicht auf Sensordaten zugreifen und diese in ihren Anwendungen nutzen. Der FROST-Server bietet Flexibilität und Robustheit, um große Mengen an Sensordaten zu verwalten und verschiedene IoT-Anwendungen zu unterstützen.

Die Technischen Betriebe Solingen betreiben einen FROST-Server, der Sensordaten aus dem Solinger Sensornetz bereitstellt. Die Sensordaten stammen aus dem RUDIS-Backend und von diversen LoRa-Sensoren

Zugangsdaten

HTTP

Für den Zugriff auf die REST-API des FROST-Servers können die folgenden Zugangsdaten verwendet werden:

- https://frost.solingen.de:8443/FROST-Server/v1.1
- Benutzer: smarthomeuser
- Passwort: Solingen2030!

MQTT

Der FROST-Server beinhaltet eine Implementierung eines MQTT-Brokers. Damit ermöglicht es dieser Änderungen von Entitäten in Echtzeit abzurufen. Der FROST-Server ist über MQTT an folgender Adresse erreichbar.

- Host: frost.solingen.de:8883
- Benutzer: smarthomeuser
- Passwort: Solingen2030!
  Da der FROST-Server ein selbst-signiertes Zertifikat nutzt, müssen entsprechende Maßnahmen auf client-Seite gemacht werden.

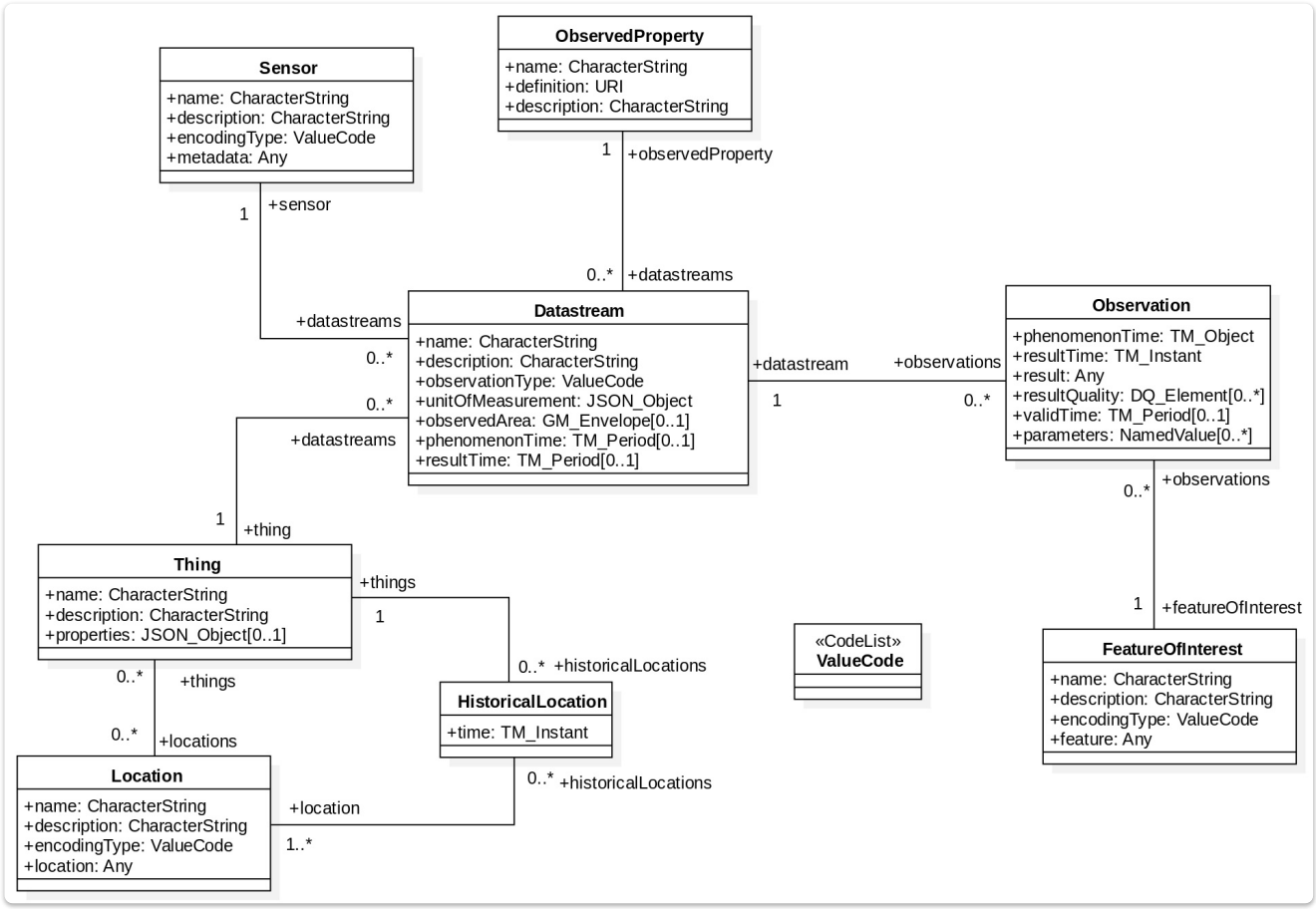
Parking Pilot - Parkraumservice

Die Parksensoren von Smart City System können über die folgenden Zugangsdaten abgerufen werden:

API Key: `eW0x7rkDNyNAlrWswP70wUplTHWFRl1QQa9wXmd-Z88=`
Doku API: https://swagger.parking-pilot.com/

Entitäten

Der FROST-Server basiert auf der Sensor-Things-API, welche ein festgelegtes Datenmodell hat (siehe Bild).



Locations

Die Locations beschreiben einen geografischen Ort, der durch Geokoordinaten (Längengrad und Breitengrad) definiert ist. Eine Location kann mehreren Things zugewiesen sein.

Thing

Ein Thing beschreibt ein Ding, dass eine Messung ausführt. Im Falle des Solinger-FROST-Servers beschreibt das Thing eine Messstelle, an welche mehrere Sensoren angebunden sind. Die Things im Solinger-FROST-Server haben zusätzlich das `properties`-Feld gesetzt, mit dem aktuellen Status der Messstelle. Diese sind

- `online` - Eine Messstelle ist Online und sendet Daten
- `offline` - Eine Messstelle ist offline und sendet keine Daten
- `decomissioned` - Eine Messstelle wurde außer Betrieb genommen. Es liegen keine aktuellen Wetterdaten vor. Es bestehen evtl. Zeitreihen für die Messstelle, die aber in der Vergangenheit liegen.

Sensor

In der Sensor-Tabelle können entweder konkrete Instanzen eines Sensors, oder eine Sensor-Definition abgelegt werden. Im Falle des Solinger-FROST-Server werden konkrete Instanzen eines Sensors abgebildete. Jeder Sensor, der an einer Messstelle angebunden ist, hat einen Eintrag in der Sensor-Tabelle.

ObservedProperty

Das ObservedProperty beschreibt eine Eigenschaft, die von einem Thing mit Sensor beobachtet wird. Das kann z.B. die Lufttemperatur, Luftdruck, die absolute oder relative Luftfeuchtigkeit sein.

Datastream

Ein Datastream bildet eine logische Verknüpfung zwischen dem Thing (und implizit der Location), dem Sensor und einer ObservedProperty. An einem Datastream ist die Tabelle der Observations angebunden. Diese bilden die Zeitreihen ab.

Observations

Die Observations-Tabelle beinhaltet alle Messswerte bzw. Zeitreihen. Das Abrufen von Observations macht nur in Verbindung eines Datastreams sinn.

Query Design

Der FROST-Server erlaubt es beim Abrufen von Daten, diese zu Filtern, bestimmte Properties zu selektieren und referenzierte Entitäten zu erweitern (Expand). Dies ist durch Angabe von Query-Parametern möglich. Im folgenden sind die verfügbaren Query-Parameter aufgelistet.

- $top: Gibt die maximale Anzahl der Entitäten an, die der FROST-Server zurückgeben soll.
- $skip: Überspringt N Entitäten in der Antwort. Kann für Paging verwendet werden.
- $count: Gibt die Anzahl der zurückgegebenen Entitäten aus. `Standard=true`
- $orderBy: Gibt die Entitäten sortiert nach einer Eigenschaft aus.
- $select: Gibt an welche Properties vom FROST zurückgegeben werden sollen. Reduziert die Ausgabe.
- $filter: Gibt nur Entitäten zurück, die bestimmten Kriterien entsprechen
- $expand: Erweitert verlinkte Entitäten und gibt diese als verschachtelte Objekte aus.

Filterung

> *aus https://fraunhoferiosb.github.io/FROST-Server/sensorthingsapi/requestingData/STA-Filtering.html*

Mit dem `$filter` Parameter kann man Entitäten bei der Abfrage nach bestimmten Kriterien filtern.
Da Filter Sonderzeichen enthalten können, müssen diese URL-encodiert werden.

Type	encoding	example
number	`direct in URL`	result gt 5
strings	`quoted with ', quotes doubled`	name eq 'Michael''s Thing'
times	`direct in URL`	phenomenonTime ge phenomenonTime ge 2022-12-23T13:21:31%2B01:00 (URL Encoded from 2022-12-23T13:21:31+01:00)
durations	`duration'<ISO 8601 code>'`	phenomenonTime gt now() sub duration'P1D'
geometries	`geography'<WKT geometry>'`	st_within(location, geography'POLYGON ((30 10, 10 20, 20 40, 40 40, 30 10))')

Filter können Operatoren und Funktionen enthalten.

Vergleichsoperatoren

Operator	Description	Example
eq	Equal	/ObservedProperties?$filter=name eq ‘CO2’
ne	Not equal	/ObservedProperties?$filter=name ne ‘CO2’
gt	Greater than	/Observations?$filter=result gt 5
ge	Greater than or equal	/Observations?$filter=result ge 5
lt	Less than	/Observations?$filter=result lt 5
le	Less than or equal	/Observations?$filter=result le 5

Logische Operatoren

Operator	Description	Example
and	Logical and	/Observations?$filter=result le 5 and FeatureOfInterest/id eq ‘1’
or	Logical or	/Observations?$filter=result gt 20 or result le 3.5

Operator	Description	Example
not	Logical negation	/Things?$filter=not startswith(description,'test')

Gruppierungsoperatoren

Operator	Description	Example
( )	Precedence grouping	/Observations?$filter=(result sub 5) gt 10

Expand

Mit dem `$expand` Parameter können Objekte, die nur als Link in der Antwort referenziert sind also vollständiges Objekt zurückgegeben werden.
Durch die Angabe von `$expand=Things` wird aus folgender Antwort


```
{
        "@iot.selfLink": "https://frost.solingen.de:8443/FROST-
Server/v1.1/Locations(72)",
        "@iot.id": 72,
        "description": "Location for sensor 4A47513169011643",
        "encodingType": "application/vnd.geo+json",
        "location": {
                "type": "Point",
                "coordinates": [
                        6.998458,
                        51.16284
                ]
        },
        "name": "Ohligs Markt",
        "HistoricalLocations@iot.navigationLink":
"https://frost.solingen.de:8443/FROST-
Server/v1.1/Locations(72)/HistoricalLocations",
        "Things@iot.navigationLink": "https://frost.solingen.de:8443/FROST-
Server/v1.1/Locations(72)/Things"
}
```


die folgende Antwort


```
{
        "@iot.count": 1,
        "value": [
                {
                        "@iot.selfLink":
"https://frost.solingen.de:8443/FROST-Server/v1.1/Locations(72)",
```

```
                        "@iot.id": 72,
                        "description": "Location for sensor
4A47513169011643",
                        "encodingType": "application/vnd.geo+json",
                        "location": {
                                "type": "Point",
                                "coordinates": [
                                        6.998458,
                                        51.16284
                                ]
                        },
                        "name": "Ohligs Markt",
                        "Things@iot.count": 6,
                        "Things": [
                                {
                                        "@iot.selfLink":
"https://frost.solingen.de:8443/FROST-Server/v1.1/Things(75)",
                                        "@iot.id": 75,
                                        "description": "LoRa Messstelle
Plantobelly",
                                        "name": "BFS Ohligser Markt 1C ",
                                        "properties": {
                                                "status": "offline"
                                        }
                                },
                                ...
                        ],
                        "HistoricalLocations@iot.navigationLink":
"https://frost.solingen.de:8443/FROST-
Server/v1.1/Locations(72)/HistoricalLocations",
                        "Things@iot.navigationLink":
"https://frost.solingen.de:8443/FROST-Server/v1.1/Locations(72)/Things"
                }
        ]
}
```

Abfrage-Beispiele

Mit den folgenden Beispielen kann man Daten aus dem FROST-Server abrufen.

Locations

Gibt alle Locations zurück

```
GET /FROST-Server/v1.1/Locations
```

Things

Gib alle Things zurück, die `online` sind.

```
GET /FROST-Server/v1.1/Things?$filter=substringof('online',
properties/status)
```

Sensors

Gibt alle Fahrbahnoberflächensensoren zurück

```
GET /FROST-Server/v1.1/Sensors?$filter=substringof('NIRS31', description)
```

Datastreams

Alle Datastreams, die `lufttemperatur` im Namen haben und deren Things `online` sind.

```
GET /FROST-Server/v1.1/Datastreams?$filter=substringof('lufttemperatur',
description) and
```

Datastreams über Observations

Gibt die Zeitreihe (Observations) für einen bestimmten Sensor für einen Tag zurück

```
GET /FROST-Server/v1.1/Datastreams?$filter=substringof('lufttemperatur',
name) and Thing/properties/status eq 'online' and Thing/Locations/name eq
'HBf Solingen'&$expand=Observations($orderby=phenomenonTime
desc;$filter=day(now()) sub day(phenomenonTime) le 1 and month(now()) eq
month(phenomenonTime)) HTTP/1.1
```

Abfragen der Sensoren

RUDIS-Sensoren (Wetter, Fahrbahnoberfläche)

Mit den folgenden Filtern kann man eine Liste aller RUDIS-Sensoren erhalten. Diese Sensoren unterscheiden sich in Wettersensoren (WS-10, WS-700, WS-800) und Fahrbahnsensoren (NIRS).

LUFFT WS-10

Intelligenter Wettersensor WS-10 (abgekündigt)

Location-Filter

```
substringof('Sensor-SG-', Things/name) and substringof('online',
Things/properties/status) and (substringof('WS-10',
Things/Datastreams/Sensor/description) or substringof('WS10',
Things/Datastreams/Sensor/description))
```

Things-Filter

```
substringof('Sensor-SG-', name) and substringof('online', properties/status)
and (substringof('WS-10', Datastreams/Sensor/description) or
substringof('WS10', Datastreams/Sensor/description))
```

Sensor-Filter

```
substringof('WS-10', description) or substringof('WS10', description)
```

ObserverdProperties-Filter

```
substringof('WS-10', Datastreams/Sensor/description) or substringof('WS10',
Datastreams/Sensor/description)
```

LUFFT WS-700/WS-800

Inteligenter Wettersensor WS-700 und WS-800 mit Blitzerkennung

Location-Filter

```
substringof('Sensor-SG-', Things/name) and substringof('online',
Things/properties/status) and (substringof('WS-700',
Things/Datastreams/Sensor/description) or substringof('WS700',
```

```
Things/Datastreams/Sensor/description) or substringof('WS-800',
Things/Datastreams/Sensor/description) or substringof('WS800',
Things/Datastreams/Sensor/description))
```

Things-Filter

```
substringof('Sensor-SG-', name) and substringof('online', properties/status)
and (substringof('WS-700', Datastreams/Sensor/description) or
substringof('WS700', Datastreams/Sensor/description) or substringof('WS-
800', Datastreams/Sensor/description) or substringof('WS800',
Datastreams/Sensor/description))
```

Sensor-Filter

```
substringof('WS-800', description) or substringof('WS800', description)
```

ObservedProperties-Filter

```
substringof('WS-800', Datastreams/Sensor/description) or substringof('WS800',
Datastreams/Sensor/description)
```

Fahrbahnoberflächensensoren LUFFT NIRS31

Non-Invasive Road Sensor 31 von LUFFT

Location-Filter

```
substringof('Sensor-SG-', Things/name) and substringof('online',
Things/properties/status) and substringof('NIRS31',
Things/Datastreams/Sensor/description)
```

Things-Filter

```
substringof('Sensor-SG-', name) and substringof('online', properties/status)
and substringof('NIRS31', Datastreams/Sensor/description)
```

Sensor-Filter

```
substringof('NIRS31', description)
```

ObservedProperties-Filter

```
substringof('NIRS31', Datastreams/Sensor/description)
```

LoRa Sensorik

Eine Auswahl an LoRa-Sensorik, die über das Niota-Backend in den FROST-Server eingespielt werden.

Grundwasserpegel

Lobaro Hybrid Gateway mit Pegeldrucksonde von Keller

Location-Filter

```
substringof('GWKA', Things/name)
```

Things-Filter

```
substringof('GWKA', name)
```

Sensor-Filter

```
substringof('Grundwasser', description)
```

ObservedProperties-Filter

```
substringof('Grundwasser', Datastreams/Sensor/description)
```

Bodenfeuchte Sensoren Bepflanzung

Bodenfeuchtesensorik von Plantobelly

Location-Filter

```
substringof('Plantobelly', Things/description)
```

Things-Filter

```
substringof('Plantobelly', description)
```

Sensor-Filter

```
substringof('Plantobelly', description)
```

ObservedProperties-Filter

```
substringof('Plantobelly', Datastreams/Sensor/description)
```

Bodenfeuchte Sensorik Bäume

Bodenfeuchtesensorik PR2 Profile Probe von Delta-T Devices

Location-Filter

```
substringof('PR2', Things/description)
```

Things-Filter

```
substringof('PR2', Things/description)
```

Sensor-Filter

```
substringof('PR2', description)
```

ObservedProperties-Filter

```
substringof('PR2', Datastreams/Sensor/description)
```

Links

- FROST-Server offizielle Dokumentation
- OGC SensorThings API Part 1: Sensing Version 1.1
- Masterportal Solingen