


Article

# Extending INSPIRE to the Internet of Things through SensorThings API

Alexander Kotsev<sup>1,†,‡,\*</sup> , Katherina Schleidt<sup>2,‡</sup>, Steve Liang<sup>3,‡</sup>, Hylke van der Schaaf<sup>4,‡</sup>,  
Tania Khalafbeigi<sup>3,‡</sup>, Sylvain Grellet<sup>5,‡</sup>, Michael Lutz<sup>1,‡</sup>, Simon Jirka<sup>6,‡</sup> and Mickaël Beaufiles<sup>5,‡</sup>

<sup>1</sup> European Commission, Joint Research Centre

<sup>2</sup> DataCove e.U., Austria

<sup>3</sup> University of Calgary, Canada

<sup>4</sup> Fraunhofer IOSB, Germany

<sup>5</sup> French Geological Survey (BRGM)

<sup>6</sup> 52°North GmbH, Germany

\* Correspondence: alexander.kotsev@ec.europa.eu; Tel.: +39 0332 78 9069

† Via E. Fermi, 2749 – TP 263 - 21027 Ispra (VA), Italy

‡ These authors contributed equally to this work.

§ The views expressed are purely those of the authors and may not in any circumstances be regarded as stating an official position of the European Commission.

Version May 2, 2018 submitted to

**Abstract:** Spatial Data Infrastructures (SDI) established during the past two decades ‘unlocked’ heterogeneous geospatial datasets. The European Union INSPIRE Directive laid down the foundation of a pan-European SDI where thousands of public sector data providers make their data available for cross-border and cross-domain reuse. At the same time, SDIs should inevitably adopt new technology and standards in order to remain fit for purpose and address in the best possible way the needs of different stakeholders (government, businesses and citizens). Some of the recurring technical requirements raised by SDI stakeholders include (i) the need for adoption of RESTful architectures, together with (ii) alternative (to GML) data encodings, such as JavaScript Object Notation (JSON) and binary exchange formats, and (iii) adoption of asynchronous publish-subscribe-based messaging protocols. The newly established OGC standard SensorThings API is particularly interesting to investigate for INSPIRE, as it addresses together all three topics. In this manuscript, we provide our synthesised perspective on the necessary steps for the OGC SensorThings API standard to be considered as a solution that meets the legal obligations stemming out of the INSPIRE Directive. We share our perspective on what should be done concerning (i) data encoding, and (ii) the use of SensorThings API as a download service.

**Keywords:** SensorThings API, INSPIRE, download services, spatio-temporal data interoperability, Internet of Things

## 1. Introduction

Back in 2007 a piece of legislation laid the foundation of the pan-European spatial data infrastructure (SDI), allowing the combined use of environmental data across borders and across domains. The Infrastructure for Spatial Information in Europe Directive [1] was established. It has ever since been catalysing the digital transformation of a vast number of public sector authorities. The adoption of the Directive brought multiple novelties to public sector authorities, including (i) the requirement to expose data and metadata in a service oriented architecture, (ii) strong reliance on international standards, and (iii) a collaborative consensus-based approach during the specification drafting. Those developments, combined, ‘unlocked’ thousands of spatial datasets. Currently, more than 100 000 datasets are made available in INSPIRE by roughly 7,000 institutions. In addition, geospatial data available within SDIs are an important contributor to the EU data economy, which is estimated at EUR 300 billion in 2016 (1.99% of the EU GDP). The estimated growth rate is also

30 remarkable, as the contribution of data to the European economy is expected to increase to EUR 739  
31 billion by 2020, representing thus 4.00% [2].

32 Nowadays, more than ten years after the adoption of the Directive, the technological scenery  
33 is rather different. Nonetheless, INSPIRE is still a driver for change with a profound impact on all  
34 actors involved. It is seen on a European level as a best practice that should be extended beyond the  
35 environmental domain [2]. The legislation and guidance documents in INSPIRE are designed to be  
36 (i) neutral from a particular software solution, and (ii) to be open to emerging technological trends.  
37 Multiple activities that contribute to the technological evolution of INSPIRE are already ongoing. The  
38 objectives of those activities are different, but they often propose solutions that would add value  
39 to data provider's infrastructures, while preserving the semantics and, wherever possible, ensure  
40 backwards compatibility with already established solutions. Multiple European actors are involved  
41 on different levels (local, regional, international) in activities dedicated to the technological evolution  
42 of SDIs and INSPIRE. Experts collaborate on improvements, such as [3] that would ensure that the  
43 infrastructure remains fit for purpose despite the rapidly changing technological scenery.

44 Within the context described above, some of the recurring technical requirements raised by  
45 stakeholders include (i) the need for adoption of RESTful architectures, together with (ii) alternative (to  
46 GML) data encodings, such as JavaScript Object Notation (JSON) and binary exchange formats, and (iii)  
47 adoption of asynchronous publish-subscribe-based messaging protocols. The newly established OGC  
48 standard SensorThings API [4] is particularly interesting to investigate for INSPIRE, as it addresses  
49 together all three topics. Developed to cover a multitude of Internet of Things (IoT) use cases, the  
50 standard is also designed to be 'developer friendly'. That is why, considering the SensorThings  
51 API within the context of INSPIRE would not only provide a possibility for the evolution of the  
52 infrastructure, but also contribute to its simplification and extension to new domains.

53 In this manuscript, we provide our synthesised perspective on the necessary steps for the OGC  
54 SensorThings API standard to be considered as a solution that meets the legal obligations stemming  
55 out of the INSPIRE Directive. We share our perspective on what should be done with regards to (i)  
56 data encoding, and (ii) the use of SensorThings API as a download service. Structurally, the paper  
57 is divided into four sections. Following a brief introduction (Section 1), in Section 2 we describe the  
58 background with an emphasis on the legislative and technological context in Europe, as well as the  
59 SensorThings API standard along with the existing implementations and an overview of use cases.  
60 Section 3 describes a mapping between the specifications of the standard and the legal requirements of  
61 INSPIRE. Both data encoding and web service operations are covered. Consequently, in Section 4 we  
62 discuss our main findings, pending challenges, and give an outlook of our future research direction.

## 63 2. Background

### 64 2.1. INSPIRE in a nutshell

65 INSPIRE came into force on 15 May 2007, with full implementation in every EU Member State  
66 required by 2021. It combines a legal and a technical framework for all EU Member States, to make  
67 relevant spatial data accessible for further reuse [5]. In particular, this means that data shall be  
68 discoverable and interoperable through the implementation of a common set of standards, data models  
69 and network (web-based) services. The thematic scope of the Directive covers 34 interdependent  
70 themes (Figure 1). The legal obligations are now transposed into the legislation of the 28 EU Member  
71 States, and the implementation is ongoing.

<p style="text-align: center;"><b>Annex I</b></p> <ol style="list-style-type: none"> <li>1. Coordinate reference systems</li> <li>2. Geographical grid systems</li> <li>3. Geographical names</li> <li>4. Administrative units</li> <li>5. Addresses</li> <li>6. Cadastral parcels</li> <li>7. Transport networks</li> <li>8. Hydrography</li> <li>9. Protected sites</li> </ol>	<p style="text-align: center;"><b>Annex III</b></p> <ol style="list-style-type: none"> <li>1. Statistical units</li> <li>2. Buildings</li> <li>3. Soil</li> <li>4. Land use</li> <li>5. Human health and safety</li> <li>6. Utility and governmental services</li> <li>7. Environmental monitoring facilities</li> <li>8. Production and industrial facilities</li> <li>9. Agricultural and aquaculture facilities</li> <li>10. Population distribution – demography</li> </ol>	<ol style="list-style-type: none"> <li>11. Area management/restriction/regulation zones &amp; reporting units</li> <li>12. Natural risk zones</li> <li>13. Atmospheric conditions</li> <li>14. Meteorological geographical features</li> <li>15. Oceanographic geographical features</li> <li>16. Sea regions</li> <li>17. Bio-geographical regions</li> <li>18. Habitats and biotopes</li> <li>19. Species distribution</li> <li>20. Energy Resources</li> <li>21. Mineral resources</li> </ol>
<p style="text-align: center;"><b>Annex II</b></p> <ol style="list-style-type: none"> <li>1. Elevation</li> <li>2. Land cover</li> <li>3. Ortho-imagery</li> <li>4. Geology</li> </ol>		
<p style="text-align: center;"><b>Implementation 2012/2017</b></p>	<p style="text-align: center;"><b>Implementation 2015/2020</b></p>	

**Figure 1.** Thematic scope of INSPIRE.

#### 72 2.1.1. Observational Data

73 The Directive is in its core about the provision of geospatial data. Nonetheless, several of  
 74 the data themes explicitly focus on observational or measurement data pertaining to different  
 75 environmental media at specific locations. Those include, but are not limited to, meteorology,  
 76 hydrology, oceanography, and geology. Technical guidance documents are prepared that cover the  
 77 encoding of observation data [6], as well as the implementation of services for downloading the data  
 78 [7]. In addition, INSPIRE includes the 'Environmental Monitoring Facilities' - EF [8]. It describes  
 79 the facility where measurements are taken, together with all relevant metadata on the measurement  
 80 process, as well as the measurement data available from the facility. The EF Theme (Figure 2) is relevant  
 81 for measurements pertaining to all environmental media, and is independent of whether a data theme  
 82 exists for this particular environmental media.

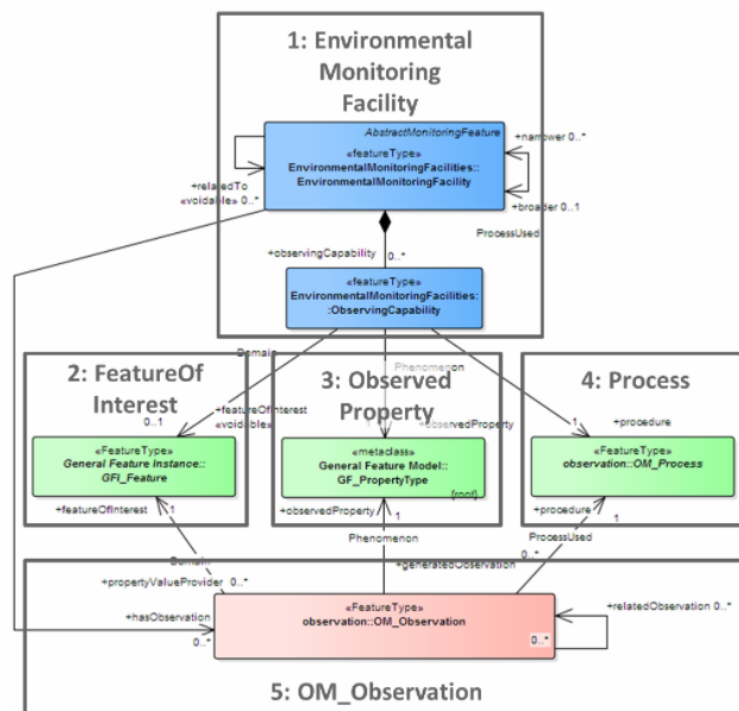


Figure 2. Environmental Monitoring Facilities.

83 • Environmental Monitoring Facilities

84 The INSPIRE EF model can be broken down into 3 interdependent parts (Figure 2) as follows:

- 85 – A description of the Environmental Monitoring Facility (blue classes)  
 86 – A description of the measurement methodology, referenced by both the Environmental  
 87 Monitoring Facility and the Observations (green classes)  
 88 – Observations providing the actual spatio-temporal data (red class)

89 • Observational Model

90 In addition to the data specification detailing what information related to an Environmental  
 91 Monitoring Facility is to be provided, the Observational Model provides specifications for  
 92 observational concepts that go beyond the EF Theme. The observational model is based on the  
 93 specifications of ISO 19156:2011 (Geographic information—observations and measurements) [9].

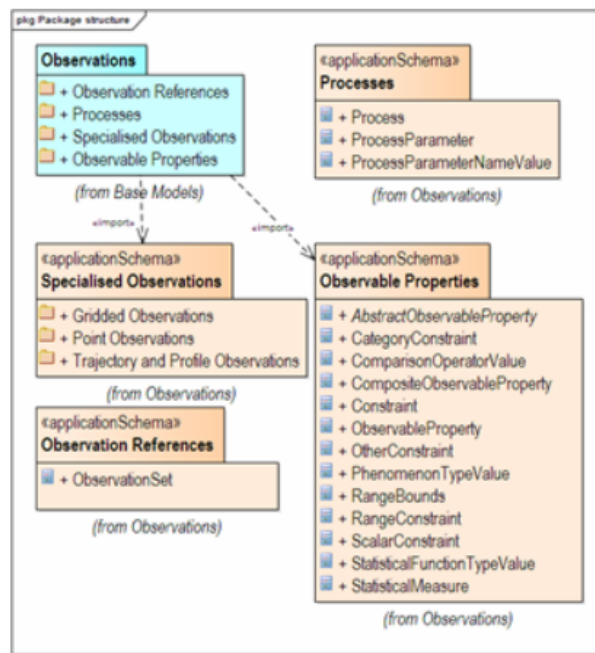


Figure 3. Observational model in INSPIRE.

Structurally, the Observational model is subdivided into four sub-packages (Figure 3):

- 94
- 95 – *Package 1. Specialized Observations.* Specializations of the base O&M Observation have
- 96 been defined through the application of constraints. Specialised observations provide an
- 97 approach that is tailored to the needs of a multitude of environmental use cases. The
- 98 following types of observations are defined in [6]:
  - 99 – *Profile observation.* Observation representing the measurement of a property along a
  - 100 vertical profile in space at a single time instant.
  - 101 – *Trajectory observation.* Observation representing the measurement of a property along
  - 102 a meandering curve in time and space.
  - 103 – *Point observation.* Observation that represents a measurement of a property at a single
  - 104 point in time and space.
  - 105 – *MultiPointObservation.* Observation that represents a set of measurements all made at
  - 106 exactly the same time but at different locations.
  - 107 – *Specimen observation.* Observation that represents a measurement of a property of a
  - 108 Specimen at a single point in time.
  - 109 – *GridObservation.* Observation representing a gridded field at a single time instant.
  - 110 – *GridSeriesObservation.* Observation representing an evolving gridded field at a
  - 111 succession of time instants.
- 112
- 113 – *Package 2. Observable Properties.* The INSPIRE Observable Properties model allows for
- 114 the specification of complex properties. The complex properties enable the creation of
- 115 composite properties, for the joint provision of strongly linked properties i.e. wind speed
- 116 and direction. Further options provided by the INSPIRE Observable Properties model
- 117 include the description of various statistical aggregates being provided, as well as the
- 118 provision of constraints on the types of result values that may be provided.
- 119 – *Package 3. Processes.* Within the OGC SWE, SensorML [10] is the preferred method for the
- 120 provision of information on the measurement process. However, this model is quite abstract
- 121 and not easy to understand and use. Thus, within INSPIRE, a simple feature type has been

122 defined for the provision of information on the measurement process. This feature type is  
 123 integrated within the EF model, and is presented in Figure 3.  
 124 – *Package 4. Observation References.* The Observation References section of the Observational  
 125 Model provides the mechanisms required for linking between features and OM\_Observation  
 126 object. Thus, an Environmental Monitoring Facility can provide association references to  
 127 observations stemming from this facility. In the same manner, an OM\_Observation can  
 128 provide a reference to the facility at which the observation was made.

### 129 2.1.2. Download services

130 Apart from data encoding, a set of network (web) services are defined in INSPIRE. They cover  
 131 the (i) discoverability of datasets through metadata, (ii) data visualisation, (iii) download, and (iv)  
 132 transformation of data (if not done offline) from their native format to the exchange format required  
 133 by INSPIRE.

134 Download services shall provide a means for downloading of whole datasets (pre-defined  
 135 access), or their subsets (direct access) through the use of a query. In addition, the specification  
 136 of INSPIRE Download Services [11] requires functionality for downloading pre-defined data sets that  
 137 are characterised by the following aspects:

- 138 • A pre-defined data set shall have a metadata record, and be discoverable through an INSPIRE  
 139 conformant discovery service.
- 140 • A resolvable link (URL) must be provided, allowing that the dataset be immediately downloaded  
 141 through a simple HTTP-protocol GET-request.

142 Guidance for data providers is made available that covers several possible technological options  
 143 (Table 1).

**Table 1.** Types of download services in INSPIRE

Name	Service type	Type of data
1. ATOM Syndication Format	Pre-defined	Whole datasets
2. WCS 2.0 (Web Coverage Service)	Pre-defined and direct access	Coverages (incl. multidimensional)
3. WFS 2.0 (Web Feature Service)	Pre-defined and direct access	Spatial features (vector)
4. SOS 2.0 (Sensor Observation System)	Pre-defined and direct access	Spatio-temporal observations

144 All the solutions described in Table 1 might be used to serve spatio-temporal data with certain  
 145 limitations. Most relevant to our work is the Sensor Observation Service, as it is to a large extent similar  
 146 to the SensorThings API. The limitations of the SOS approach relate to (i) the missing functionality of  
 147 the SOS to filter observations based on their result values, although a corresponding extension has  
 148 recently been proposed, (ii) lack of pagination, and (iii) the lack of an adopted REST/JSON binding of  
 149 the SOS standard.

### 150 2.2. The OGC SensorThings API

151 The OGC SensorThings API [4] is an open geospatially-enabled IoT standard that overcome  
 152 interoperability challenge in the IoT. It can be described as a ‘Sensor Web Enablement for the Internet  
 153 of Things’. SensorThings is designed to interconnect heterogeneous devices, data, and applications  
 154 through the web. It provides a REST-like application programming interface (API) to manipulate the  
 155 data. A publish-subscribe-based messaging protocol extension for real-time operations is available  
 156 through the use of the Message Queuing Telemetry Transport (MQTT) ISO standard [12]. SensorThings  
 157 API benefits from the use JSON as a light-weight means for data encoding. In addition, the standard  
 158 is based on OGC SWE standards and OData [13] with the focus on being light-weight, providing  
 159 comprehensive data model applicable for different IoT use cases, and ease of use. An increasing  
 160 number of use cases are implement the SensorThings API for handling spatio-temporal observation



161 data. They span across different application domains such as air pollution monitoring [14], smart city  
 162 services [15,16] and Internet of Things [17].

163 The specifications of the standard consist of two parts: ‘Sensing’ and ‘Tasking’. The ‘Sensing’ part,  
 164 which is the focus of this paper, is for sensing and gathering observations from sensing devices. The  
 165 ‘Tasking’ part [18] focuses on controlling IoT devices, which is out of the scope of this paper. That is  
 166 why, in this paper wherever we using Sensorthings API, we refers to the Sensing part only.

### 167 2.2.1. The Data Model

168 SensorThings incorporates the OGC Observations and Measurement standard [9] for data  
 169 encoding. The comprehensive data model (Figure 4) makes SensorThings API adaptable to wide  
 170 variety of IoT use cases.

171 Conceptually, the data model is subdivided into eight entities (Figure 4) that are further described  
 172 below. A ‘Thing’ is the central entity in the data model. It can be physical or virtual, and is equipped  
 173 with one or more ‘Sensor’ to collect Observations. Depending on the use case this can be the  
 174 object being observed, or the sensor platform, such as a satellite. Each Thing has a Location.  
 175 If the Thing is static this Location would never change. However, if the ‘Thing’ is moving, its  
 176 ‘Location’ is also changing frequently. In those cases the HistoricalLocation is the entity for  
 177 keeping track of previous Locations of the moving Thing. The data model is defined, so that a  
 178 Thing can be linked to more than one Location. However, all the Locations that are connected to  
 179 one Things should be different representations of the same physical location. This feature is useful  
 180 when multiple representations of a Thing’s Location should be modelled (e.g. lat-lon as well as a room  
 181 number). Each Thing can have one or more Datastream. The Datastream is an entity for grouping  
 182 Observations of one Sensor that are observing the same phenomenon, called ObservedProperty.  
 183 Each Observation has a FeatureOfInterest that it observes, as well as a Datastream.

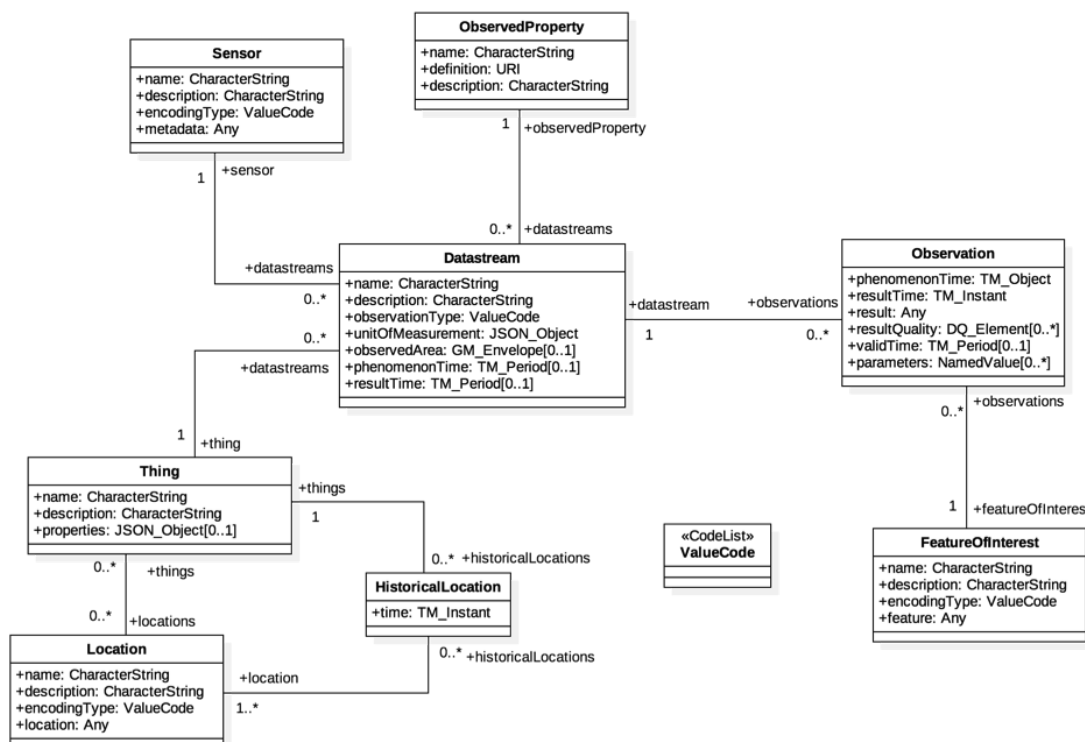


Figure 4. The SensorThings API data model. Source: [4].

### 184 2.2.2. RESTful interface

185 The SensorThings rest API is loosely based on the OData API [19]. Each of the entity types  
186 described in Section 2.2.1 has an entity collection through which these entities can be accessed. Fetching  
187 the base URL of the API with a HTTP Get request would return an index document listing the URLs of  
188 each of the available entity collections.

189 A HTTP Get request on a collection returns a list of all entities in the collection. Each entity  
190 can also be fetched individually by appending the entity ID, in parentheses, to the entity collection.  
191 Besides its specific properties, each entity has an identifier, encoded in JSON as "@iot.id", a self link,  
192 under "@iot.selfLink" and links to other entities as described in the data model. For example,  
193 a Thing can have relations to multiple Datastreams. Therefore, each Thing has a navigation link to  
194 a collection of Datastreams, listed under "Datastreams@iot.navigationLink" in its JSON, to  
195 /Things(<id>)/Datastreams. Likewise, each Datastream is linked to exactly one Thing. Therefore  
196 each Datastream has a navigation link to this Thing, listed under "Thing@iot.navigationLink",  
197 to /Datastreams(<id>)/Thing.

198 A request to a collection is subject to pagination, based on the request parameters and server  
199 settings. A client can request the number of entities returned to be limited using the "\$top" query  
200 parameter. The server will not return more than this number of entities, but it can return fewer if it is  
201 configured with a lower threshold. If there are more entities to be returned than allowed in a single  
202 request, the server adds a link to the result, named "@iot.nextLink", that returns the next batch  
203 of entities. The client itself can also request that a number of entities are skipped, using the query  
204 parameter "\$skip". For example, a client can request entities 11 to 15 by using \$top=5&\$skip=10.

205 Entities in a collection can be ordered using the "\$orderby" query parameter. The  
206 entities can be ordered by one or more of their properties, in ascending or descending  
207 order. If a client is not interested in all properties of the requested entities, it can limit  
208 the properties to be returned by using the "\$select" query parameter. For example, the  
209 query to /Observations?\$select=result,phenomenonTime only returns the result and  
210 phenomenonTime of all Observations.

211 When requesting entities from a collection, a filter can be applied to the entities with the  
212 "\$filter" query parameter. This filter can act on any of the properties of the entities in the collection,  
213 or any of the properties of related entities. It is, for example, possible to request all Observations that  
214 have a phenomenonTime in a certain time range, or all Observations that have a Datastream that has a  
215 Thing with a certain name. The filtering options are extensive and include geospatial, mathematical  
216 and string functions. In addition, multiple filters can be combined with a Boolean operator (and, or,  
217 and not) and parenthesis.

218 Finally, when requesting entities, it is possible to have related entities be directly included in  
219 the response, by using the "\$expand" query parameter. The expanded items can be subjected to all  
220 query parameters, including the "\$expand" query parameter. This makes it possible to perform a  
221 single request for a Thing, including its Datastreams and their ObservedProperties, and the latest  
222 Observation of each of these Datastreams.

### 223 2.2.3. Implementations

#### 224 • FROST-Server

225 FROST-Server (Fraunhofer OpenSource SensorThings Server) is an open-source implementation  
226 of the standard, developed by the German research institute Fraunhofer IOSB, to cover their need  
227 for a standards-based, easy to use sensor management and sensor-data storage platform, for use in  
228 various research programs. It is written in the Java programming language and can be deployed on  
229 servlet containers like Apache Tomcat or Wildfly. For data persistence it currently has backends for the  
230 PostgreSQL database management system, with either numeric, string or UUID entity identifiers.



231 Since it was developed to cover the wide range of use-cases that appear in research projects,  
232 the focus of development was on feature completeness and extendibility. The server implements the  
233 complete specification of the OGC SensorThings API Part 1: Sensing, including all extensions [4]. The  
234 open-source nature of the implementation means that users can tune and optimise the implementation  
235 for their specific use-case.

236 • 52°North

237 52°North provide another comprehensive open source implementation that allows to share  
238 observation data and corresponding metadata not only based on the OGC Sensor Observation Service  
239 (SOS) interface, but also through a range of complementary interfaces (e.g. a non-standardised REST  
240 API especially optimised for the development of lightweight client applications).

241 In 2017 the 52°North Sensor Web team has completed a first prototype for evaluating different  
242 ways how to enhance the 52°North SOS implementation with support of the SensorThings API.  
243 Because of strong similarities between the SOS and SensorThings API interfaces as well as data models  
244 (i.e. ISO/OGC Observations and Measurements is also a core foundation of the SensorThings API) this  
245 enhancement was rather easy to achieve. Currently, there is ongoing work at 52°North to advance the  
246 prototypical developments into a stable module that will be published together with the next major  
247 SOS release expected for 2018.

248 • SensorUp SensorThings

249 SensorUp, a Calgary-based startup, developed the first compliant SensorThings implementation  
250 and the implementation is considered as SensorThings reference implementation. It is a Java-based  
251 implementation and the current persistence system is PostgreSQL database. The implementation  
252 is tuned for scalability without loss of performance. In addition to server development, SensorUp  
253 provides multiple clients and client libraries to make SensorThings even easier to use for client  
254 developers. Moreover, wide variety of documentations, interactive SDKs, tutorials, and examples are  
255 provided by SensorUp for developers.

256 • GOST

257 GOST is an open source implementation of SensorThings based on GO programming language.  
258 It is developed by Geodan, a geo-ICT company in Amsterdam. The implementation passed the  
259 SensorThings test suite and is OGC certified. It also has MQTT extension implemented. It is considered  
260 as alpha software and is not yet considered appropriate for customer use. GOST provides docker  
261 images as their easy deployment as well as binaries for people don't prefer docker.

262 • Mozilla

263 Mozilla has a Node implementation of SensorThings. This implementation is open source and  
264 almost passed all the OGC test suite tests. The implementation uses PostgreSQL for persistence layer.  
265 The development is not active since February 2017.

266 • CGI Kinota Big Data

267 CGI develops a modular implementation of SensorThings called Kinota Big Data. Kinota is  
268 designed to support different persistence platforms from RDBMS to NoSQL database. The current  
269 implementation supports Apache Cassandra. Kinota implements only a subset of SensorThings  
270 requirements. It is also a Java-based implementation and has GNU license.

### 271 3. SensorThings API for INSPIRE

#### 272 3.1. Alignment of Data Specifications

273 In order to determine whether and how the SensorThings API can be utilised to fulfil the INSPIRE  
274 requirements pertaining to the data scope defined within the legislation, an alignment between these  
275 two data models must be specified. We approached this alignment on two hierarchical levels. Initially,  
276 the SensorThings API data model was analysed to determine which class or classes best correspond to  
277 the classes specified for the INSPIRE EF Theme [8], as well as for the specialized observations from  
278 the observational model of the INSPIRE Generic Conceptual Model - GCM [20]. Table 2 provides an  
279 overview of the proposed alignment for EF as well as for the base O&M Observation type. Secondly,  
280 once corresponding classes were identified, the individual attributes were aligned. In addition,  
281 necessary extensions were specified for attributes not available within the core SensorThings API data  
282 model.

283 Once corresponding classes were identified, the individual attributes were aligned to the  
284 requirements of the EF data specification. While all requirements of the SensorThings API data  
285 model could be mapped to mandatory data from EF, EF requirements were identified that could not be  
286 mapped to the SensorThings API data model. However, the addition of a properties section, allowing  
287 for the inclusion of a block of JSON encoded data, has already been recommended for inclusion in  
288 the upcoming 1.1 version of the SensorThings API standard. This properties section has been utilised  
289 within the alignment tables (Appendix ??), allowing for full compliance with INSPIRE.

290 The specialized observations from the observational model of the GCM are defined by constraint.  
291 The ramifications of these constraints were analysed to determine a semantically equivalent usage of  
292 the SensorThings API data model. The following models for provision of the INSPIRE specialised  
293 observation types have been preliminarily sketched illustrating options for implementation. These  
294 should be further analysed before wide-spread use:

- 295 ● *Point observation*: Datastream with one Observation pertaining to a single FeatureOfInterest.
- 296 ● *Point Time Series Observation*: Datastream with multiple Observations over time pertaining to a  
297 single FeatureOfInterest.
- 298 ● *Multi Point Observation*: Datastream with multiple Observations pertaining to different  
299 FeaturesOfInterest. Some additional grouping of the FeaturesOfInterest may be required.
- 300 ● *Grid Observation*: Datastream with one Observation providing a complex result, FeatureOfInterest  
301 is a Grid. The Coverage Implementation Schema (CIS 1.1) should be explored for the provision  
302 of gridded coverages, as GeneralGridCoverage seems well suited for this purpose, and a JSON  
303 encoding is provided.
- 304 ● *Grid Series Observation*: Datastream with multiple Observations providing complex results,  
305 FeatureOfInterest is a Grid. CIS 1.1 [21] should be explored for the provision of gridded coverages,  
306 as GeneralGridCoverage seems well suited for this purpose, and a JSON encoding is provided.
- 307 ● *Profile Observation*: Depending on the type of Profile, various options exist for encoding via the  
308 SensorThings API. These range from the model sketched for Multi Point Observation, whereby all  
309 FeaturesOfInterest must share the same values for Lat/Long, over the use of a MultiDatastream  
310 providing a depth indicator (depth in m or pressure) as a second ObservedProperty to the  
311 utilization of grid types as specified within the constraints on the INSPIRE model.
- 312 ● *Trajectory Observation*: Datastream with multiple Observations with a varying FeatureOfInterest.  
313 Some additional grouping of the FeaturesOfInterest may be required in order to expose the  
314 trajectory to client applications.

315 In addition, the list of observation types available from the OGC Naming Authority [22] does not  
316 include all observation types required for INSPIRE. These should either be appended to this list, or  
317 alternatively be made available from an alternative source.

Table 2. Mapping between INSPIRE Feature Types and SensorThings API Entities

INSPIRE Feature Type	Definition	SensorThings API entity	Comments
1. Environmental Monitoring Facility	A georeferenced object directly collecting or processing data about objects whose properties (e.g. physical, chemical, biological or other aspects of environmental conditions) are repeatedly observed or measured. Can also host other environmental monitoring facilities.	'Thing' and 'Location'	The ObservingCapability Feature Type associated to the Environmental Monitoring Facility via composition corresponds to a 'Datastream'.
2. Feature Of Interest	Feature that is the subject of the observation, and carries the observed property being investigated.	'FeatureOfInterest'	
3. Observed Property	The Property Type for which the OM_Observation:result provides an estimate of its value.		
4. Process	Process used to generate the result.	'Sensor'	
5. OM_Observation	Links the domain and range of the observation being described, together with all relevant metadata required for the interpretation of this observation.	'Datastream' and 'Observation'	

### 318 3.2. Download service operations

319 As clarified in Section 2.1.2, INSPIRE distinguished between (i) pre-defined, and (ii) direct  
 320 access download services. The implementation of the former requires functionality for downloading  
 321 whole datasets. They shall be documented through metadata which is exposed online through a  
 322 discovery service, following the specifications of the OGC Catalogue Service for the Web - CSW [23]. A  
 323 resolvable link (URL) shall be provided within each metadata record through which the dataset can be  
 324 downloaded by sending a simple HTTP-protocol GET-request.

#### 325 3.2.1. Pre-defined dataset

326 The INSPIRE Network Service Regulation [11] defines an abstract service model through a set  
 327 of operations that each download service shall implement. No ability to query datasets or select  
 328 user-defined subsets of datasets is foreseen for the pre-defined datasets. The required operations,  
 329 together with the proposed equivalent in SensorThingsAPI are provided in Table 3.

**Table 3.** Operations for pre-defined dataset download services

Service operation	Definition	Mapping to SensorThings API
1. Get Download Service Metadata	Provides all necessary information about the service, the available Spatial Data Sets, and describes the service capabilities.	OpenAPI [24] document. This would require a mapping of the relevant (mostly static) contents of OGC GetCapabilities responses to OpenAPI.
2. Get Spatial Dataset	Retrieval a dataset as a whole, as defined by the data provider	The simplest option would be to map this operation to 'Datastreams', i.e. each datastream to deliver an INSPIRE dataset that is described separately through metadata. As a difference to the SOS offering concept [7], the datasets defined by the SensorThings API are restricted to one observed property. Thus, 'MultiDatastreams' shall be used if the dataset contains more than one 'ObservedProperty'. Alternatively, the powerful query mechanism of SensorThings API can be used to request information spread into multiple 'Datastreams', including 'Observations', 'Observed Properties', 'Things', 'Locations' and 'FoI', all filtered by time interval, area, or other criteria. Sample requests are provided in Appendix B.
3. Describe Spatial Dataset	Description of metadata about a dataset and all types of Spatial Objects contained in the Spatial Dataset.	This can be mapped to (i) 'Datastream' if the dataset consists of a single ObservedProperty, and (ii) 'MultiDatastream' if the dataset contains more than one ObservedProperty.
4. Link Download Service	Allows the declaration of the availability of a Download Service while maintaining the downloading capability at the Public Authority or a Third Party location.	

330 This mapping shows that the SensorThings API is generally capable of providing the functionality  
 331 needed for implementing the pre-defined dataset download functionality that is mandatory for  
 332 INSPIRE Download Services. However, there are several open issues to be addressed with regard to  
 333 the mapping proposed in Table 3:

- 334 • *Capabilities for SensorThings API services.* The standard provides a RESTful API for access to data.  
 335 This is a convenient approach which allows easy uptake and immediate use by mainstream  
 336 developers. The *GetCapabilities* operation included in the majority of OGC standards is not  
 337 available. A solution for INSPIRE might be to use the OpenAPI standard (former Swagger) [24]  
 338 to document the service, considering the required, mostly static, elements of the *Get Download*  
 339 *Service Metadata* and *Describe Spatial Dataset* operations.
- 340 • *Dataset in SensorThings API.* The concept of a 'dataset' does not exist in the SensorThings API  
 341 standard. If following our proposal to equalise a dataset to a 'Datastream', it should represent a  
 342 logically consistent grouping of individual observations.
- 343 • *Support for request parameters.* Request parameters are defined in [11] for the *Get Spatial Dataset*  
 344 operation. Users should be able to request their datasets or spatial objects in any of the (i) offered  
 345 coordinate reference systems (CRS), and (ii) natural languages, as well as (iii) through their  
 346 spatial data set identifier. The REST-like interface of SensorThings API does not support the first  
 347 two parameters.

### 348 3.2.2. Direct access

349 In addition to the mandatory operations described in Section 3.2.1, non-mandatory 'direct access'  
 350 operations may be implemented to ensure that users can acquire the desired data in a flexible manner  
 351 (i.e. sub-setting the datasets by properties such as time periods and spatial extends). From our  
 352 perspective they would be easy to implement through SensorThings API. Their mappings are provided  
 353 in Table 4. Even though that direct access operations are not mandatory, from our perspective they are  
 354 the ones that are, and would increasingly be, desired by users, as they really add value to the data  
 355 provider infrastructure.

Table 4. Operations for direct access download services

Service operation	Definition	Mapping to SensorThings API
1. Get Spatial Object	Retrieval of Spatial Objects based upon a query.	Request that resolves to an Observation entity in a collection together with its id.
2. Describe Spatial Object Type	Description of the specified Spatial Objects types.	Metadata provided as part of the Datastreams or Observations downloaded from a SensorThings API endpoint.
3. Link Download Service	Allows the declaration of the availability of a Download Service while maintaining the downloading capability at the Public Authority or a Third Party location.	

356 Similarly to the *Get Spatial Dataset* operation, the *Get Spatial Object* operation shall support requests  
 357 for coordinate reference system and language that are not supported by the standard.

## 358 4. Discussion and conclusions

359 Within this manuscript we (i) outlined the challenges associated with the provision of  
 360 spatio-temporal data in spatial data infrastructures, (ii) described the contemporary technological  
 361 scenery and emerging standardisation initiatives, and most importantly (iii) proposed a solution that  
 362 would allow the new SensorThings API to be considered as an INSPIRE solution. In summary, we  
 363 consider that **no major blocking factors exist for proposing SensorThings as an INSPIRE solution.**

364 As outlined in Section 2.1, INSPIRE provides a transparent and straightforward means for  
 365 encoding the location of environmental monitoring facilities and observation data. Nonetheless, there  
 366 are several issues to be addressed. They include:



- 367 • *Definition of a dataset.* Datasets in INSPIRE should include observations for a predefined period of  
368 time and for logically consistent geographical entities usually observed by the same procedure.  
369 Similarly to SOS, the concept of a dataset is unknown to the SensorThings API. This we see as a  
370 broader issue that should be addressed within INSPIRE.
- 371 • *SensorThings API Extension Points.* The data model of the SensorThings API has been concisely  
372 tailored to the requirements of sensors within the IoT while the INSPIRE data specifications  
373 require the provision of additional contextual information as required within an SDI. While some  
374 attributes required by INSPIRE can be directly aligned with attributes from the SensorThings  
375 API, additional attributes are required. For this purpose, we utilized the currently proposed  
376 extension (Appendix ??) to the SensorThings API 1.1, whereby a properties attribute of type  
377 JSON\_Object, as presently provided by the class Thing, is appended to the classes Datastream  
378 and Sensor. All additional requirements stemming from INSPIRE can be supported by providing  
379 this within the JSON properties structure; for FeatureOfInterest the feature attribute serves the  
380 same purpose.
- 381 • *Metadata for services.* The SensorThings API provides a RESTful API for simultaneous access to  
382 both data and metadata, thus the API does not require a separate operation to request metadata.  
383 However, the INSPIRE Directive mandates such an operation. Considering the fact that the  
384 vast majority of required metadata elements are static, it would be fairly easy to document  
385 STA services through the OpenAPI specification [24]. Technical guidance should be released to  
386 describe such an approach. Within this context, it is important to highlight that SensorThings  
387 API provides an opportunity to rethink the INSPIRE Metadata vs. Data paradigm, which  
388 inevitably leads to an artificial divide between metadata and data, thus leading to unavoidable  
389 redundancies in information.
- 390 • *Request for CRS and Language.* As outlined in Section 3.2 requesting datasets or individual  
391 observations in a particular natural language or coordinate reference system is not supported by  
392 SensorThings API. Satisfying this requirement would require a workaround that goes beyond  
393 the current version of the standard. When addressing the issue of CRS in particular, two aspects  
394 should be considered. Firstly, the provision of functionality for requesting CRS would add  
395 complexity to the service interface which might be conflicting with the overarching objective of  
396 the standard to be as developer friendly as possible. Secondly, the SensorThings API suggests  
397 GeoJSON for data encoding, and the conformance tests only test with it. This requirement poses  
398 a limitation, as all coordinates shall be provided in a geographic coordinate reference system,  
399 using the World Geodetic System 1984 datum [25].

400 In our future research we would focus on the following:

- 401 • *Comparison with other download services.* So far we analysed the SensorThings API from a legislative  
402 and technical perspective without going deeper into the analysis of how our proposed solution  
403 would fit with the rest of the implementations in the pan-European spatial data infrastructure.  
404 This deserves to be further investigated from at least two angles, namely (i) comparison with  
405 other possible solutions such as SOS and WFS, and (ii) approaches combining two or more  
406 solutions, e.g. where static data is handled through a WFS, and SensorThings API handles the  
407 spatio-temporal data.
- 408 • *Specialised observation types.* INSPIRE defines a set of specialised observation types (Section  
409 3). For some, such as the *Point observation* and *Point Time Series Observation*, implementation  
410 options are quite straightforward. Others, foremost those pertaining to grids such as the *Grid  
411 Observation* and *Grid Series observation*, but also *Profile observation* and *Trajectory observation*  
412 would require additional attention. For the provision of observations pertaining to grids (*Grid  
413 Observation* and *Grid Series observation*) available grid encoding standards must be analysed for  
414 suitability. CIS 1.1 [21] seems promising in this regard. For the encoding of *Multi Point Observation*,  
415 *Profile Observation* and *Trajectory Observation* methods for grouping a set of FeaturesOfInterest  
416 should be explored, as such a mechanism would provide easier handling for client applications.



- 417 For the provision *Profile Observation*, alternatives utilising *MultiDatastreams* or Grids should also  
 418 be taken into consideration.
- 419 • *Asynchronous transactions for INSPIRE download services.* The SensorThings API offers  
 420 out-of-the-box functionality for publish-subscribe-based messaging through the use of MQTT  
 421 [12]. At the same time INSPIRE is, legislatively, bound to the request-response paradigm for data  
 422 exchange. Given the rapid development of the IoT with its constraint devices where the latency  
 423 of networks is challenging, the use of publish-subscribe services should be further investigated.
  - 424 • *Sensor Tasking.* The possibility to assign tasks to environmental sensors in a standardised manner  
 425 goes beyond the legal requirements of INSPIRE. However, considering the rapid growth of the  
 426 number of connected IoT devices, this would be a very interesting, particularly for the planning  
 427 and implementation of measurement campaigns. Within this context, the "SensorThings API,  
 428 Part 2 - Tasking Core" Candidate Standard [18] is to be investigated.
  - 429 • *Other standards.* From our perspective the approach, demonstrated in this paper, for analysing  
 430 possible technological solutions for INSPIRE is applicable for other standards as well. Most  
 431 prominent from that perspective would be to investigate the emerging Web Feature Service 3.0  
 432 (WFS 3.0) [26] as a potential INSPIRE solution.

433 **Supplementary Materials:** Mapping between INSPIRE and SensorThings API attributes, described in Appendix  
 434 A, is available online at <https://github.com/DataCoveEU/SensorThings/tree/master/INSPIRE>

435 **Acknowledgments:** The work on data model alignment between STA and INSPIRE was partially performed  
 436 under the BRGM contract "SensorThings API Deployment within the French Groundwater Information Network".  
 437 The contribution of Simon Jirka (52°North GmbH) was supported by the project COLABIS (German Federal  
 438 Ministry of Education and Research, programme Geotechnologien, under grant agreement No 03G0852A) as well  
 439 as the European projects BRIDGES (European Union's Horizon 2020 research and innovation programme under  
 440 grant agreement No 635359) and ODIP II (European Union's Horizon 2020 research and innovation programme  
 441 under grant agreement No 654310).

442 **Author Contributions:** The description of STA was prepared by SL, TK and HvdS. AK and ML provided an  
 443 overview of INSPIRE. The alignment of data requirements between STA and INSPIRE was done by KS, SG and  
 444 MB. SJ, HvdS and AK analysed the alignment of STA with INSPIRE download service specifications. AK and KS  
 445 built the overall storyline, contributed to all sections of the paper, and coordinated the input by the rest of the  
 446 co-authors.

447 **Conflicts of Interest:** The authors declare no conflict of interest.

## 448 Abbreviations

449 The following abbreviations are used in this manuscript:

450 API	Application Programming Interface
CIS	Coverage Implementation Schema
CRS	Coordinate Reference System
CSW	Catalogue Service for the Web
EF	Environmental Monitoring Facilities
GNU	GNU General Public License
IoT	Internet of Things
ISO	International Organization for Standardization
MQTT	Message Queuing Telemetry Transport
451 OData	Open Data Protocol
OpenAPI	OpenAPI Specification
RDBMS	Relational Database Management System
SDI	Spatial Data Infrastructure
SDK	Software Development Kit
STA	SensorThings API
SWE	OGC Sensor Web Enablement Suite
WFS	Web Feature Service
XML	Extensible Markup Language

**Table A1.** Alignment between the INSPIRE 'Environmental Monitoring Facility' class and STA

Environmental Monitoring Facility	Class	Attribute	CR on Part 1 for properties extension	ST for properties extension
gml:id	THINGS	ID		
gml:identifier	THINGS	ID		
inspire Id				
localId	THINGS	ID		
namespace	THINGS	PROPERTIES/namespace	X	
name	THINGS	NAME		
media Monitored	THINGS	PROPERTIES/mediaMonitored	X	
geometry	LOCATIONS			
gml:id	LOCATIONS	ID		
srsDimension	LOCATIONS	GEOM		
srsName	LOCATIONS	GEOM		
gml:Point	LOCATIONS	GEOM		
observing Capability	DATASTREAMS			
gml:id	DATASTREAMS	ID		
observing Time	DATASTREAMS			
gml:id	DATASTREAMS	ID		
beginPosition	DATASTREAMS	PHENOMENON_TIME_START		
endPosition	DATASTREAMS	PHENOMENON_TIME_END		
process Type	DATASTREAMS	PROPERTIES/processType	X	
result Nature	DATASTREAMS	PROPERTIES/resultNature	X	
online Resource	DATASTREAMS			
featureOfInterest	OBSERVATIONS	FEATURE_ID		
observedProperty	DATASTREAMS	OBS_PROPERTY_ID		
procedure	DATASTREAMS	SENSOR_ID		
measurement Regime	THINGS	PROPERTIES/measurementRegime	X	
mobile	THINGS	PROPERTIES/mobile	X	
operational Activity	THINGS	PROPERTIES	X	
Period				
gml:id	THINGS	ID		
beginPosition	THINGS	PROPERTIES/beginTime	X	
endPosition	THINGS	PROPERTIES/endTime	X	
comment	THINGS	DESCRIPTION		

## 452 Appendix A Alignment of INSPIRE EF Requirements to SensorThings API

### 453 Appendix A.1 EF SamplingPoint

### 454 Appendix A.2 Process

### 455 Appendix A.3 Feature of Interest

### 456 Appendix A.4 Observation

## 457 Appendix B Sample requests to SensorThings API services

### 458 Appendix B.1 Request of an Air Temperature and Humidity dataset

459 The query below returns, for a given 'Thing' its metadata, the 'Location', and 'Datastreams' that  
 460 measure Air Temperature and Humidity. For each 'Datastream' the 'Sensor' and 'ObservedProperty'  
 461 metadata, all observations for a given period, and for each 'Observation' the id of the

Table A2. Alignment between the INSPIRE 'Process' class and STA

Type	Class	Attribute	CR on ST Part 1 for properties extension	
OM_Process	gml:id	ID		
	gml:identifier	ID		
	inspireId			
	localId	ID		
	namespace	PROPERTIES/namespace	X	
	name	NAME		
	type	PROPERTIES/type		
	documentation	DESCRIPTION		
	responsible Party	PROPERTIES/responsibleParty	X	
RelatedParty	individualName	PROPERTIES/responsibleParty/individualName	X	
	organisationName	PROPERTIES/responsibleParty/organisationName	X	
	positionName	PROPERTIES/responsibleParty/positionName	X	
	contact//address//language	PROPERTIES/responsibleParty/language	X	
	contact//address//adminUnit//SpellingOfName			
	text	PROPERTIES/responsibleParty/adminUnit	X	
	script	PROPERTIES/responsibleParty/script	X	
	address//locatorDesignator	PROPERTIES/responsibleParty/locatorDesignator	X	
	address//postCode	PROPERTIES/responsibleParty/postCode	X	
	electronicMailAddress	PROPERTIES/responsibleParty/electronicMailAddress	X	
	telephoneVoice	PROPERTIES/responsibleParty/telephoneVoice	X	
	website	PROPERTIES/responsibleParty/website	X	
	role	PROPERTIES/responsibleParty/role	X	

**Table A3.** Alignment between the INSPIRE 'Feature of Interest' class and STA

Type	Class	Attribute	CR on Part 1 for properties extension
gml:id	FEATURES	ID	
gml:identifier	FEATURES	ID	
sf:type	FEATURES	PROPERTIES/type	X
sf:sampledFeature	FEATURES	PROPERTIES/sampledFeature	X
sams:shape	FEATURES	FEATURE	
gml:id	FEATURES	GEOM	
srsName	FEATURES	GEOM	
gml:geometry	FEATURES	GEOM	

**Table A4.** Alignment between the INSPIRE 'Observation' class and STA

Type	Class	Attribute
gml:id	DATASTREAMS	ID
phenomenonTime		
gml:id	DATASTREAMS	ID
beginPosition	DATASTREAMS	PHENOMENON_TIME_START
endPosition	DATASTREAMS	PHENOMENON_TIME_END
resultTime		
gml:id	DATASTREAMS	ID
beginPosition	DATASTREAMS	PHENOMENON_TIME_END
procedure	DATASTREAMS	SENSOR_ID
observedProperty	DATASTREAMS	OBS_PROPERTY_ID
featureOfInterest	OBSERVATIONS	FEATURE_ID
result	OBSERVATIONS	RESULT_NUMBER

462 'FeatureOfInterest'. The example covers a static station. That is why the FoI is always the same,  
463 coinciding with the 'Location' of the 'Thing', so this data could be left out.

```

https://sensorthingsserver.eu/SensorThingsService/v1.0/Things(10)
?$select=name,id
&$expand=Locations,Datastreams(
  $filter=ObservedProperty/id eq 2 or ObservedProperty/id eq 5
  ;$select=name,id,unitOfMeasurement
  ;$expand=
    Sensor($select=id,name,description)
    ,ObservedProperty($select=id,name,definition,description)
    ,Observations(
      $filter=phenomenonTime ge 2018-02-01T00:00:00Z and
      phenomenonTime lt 2018-03-01T00:00:00Z
      ;$orderby=phenomenonTime asc
      ;$select=phenomenonTime,result
      ;$expand=
        FeatureOfInterest(
          $select=id
        )
      )
    )
  )
)

```

**Figure A1.** Sample request to a SensorThings API server for retrieving an air temperature and humidity dataset

#### 464 Appendix B.2 Request of a Water level dataset

465 This query below returns all 'Datastreams' in a given area that measure a "Water Level" including  
466 the 'Datastreams' marked as 'forecast', including their 'Thing' and the Thing's 'Location', 'Sensor',  
467 'ObservedProperty' and 'Observations'.

#### 468 References

469

- 470 1. EC. Directive 2007/2/EC of the European Parliament and of the Council of 14 March 2007 establishing  
471 an Infrastructure for Spatial Information in the European Community (INSPIRE). *Published in the official*  
472 *Journal on the 25th April 2007*.
- 473 2. EU. Building a European data economy. Communication from the Commission 2017. Accessed: 2018-02-22.
- 474 3. "INSPIRE – What if ..." workshop proceedings.
- 475 4. Liang, S.; Huang, C.Y.; Khalafbeigi, T.; others. OGC SensorThings API-Part 1: Sensing. OGC®  
476 *Implementation Standard*. <http://docs.openeospatial.org/is/15-078r6/15-078r6.html> 2016.
- 477 5. Cetl, V.; Tomas, R.; Kotsev, A.; de Lima, V.N.; Smith, R.S.; Jobst, M. Establishing common ground  
478 through INSPIRE: the legally-driven European Spatial Data Infra-structure. Springer Lecture Notes in  
479 Geoinformation and Cartography (in press), 2018.
- 480 6. Guidelines for the use of Observations and Measurements and Sensor Web Enablement-related standards  
481 in INSPIRE Annex II and III data specification development.
- 482 7. OGC. Sensor Observation Service interface standard. *Open Geospatial Consortium Interface Standard* 2012,  
483 pp. 12–006.
- 484 8. INSPIRE Data Specification on Environmental Monitoring Facilities – Technical Guidelines. [https://](https://inspire.ec.europa.eu/id/document/tg/ef)  
485 [inspire.ec.europa.eu/id/document/tg/ef](https://inspire.ec.europa.eu/id/document/tg/ef). Accessed: 2018-04-10.

```

https://sensorthingsserver.eu/SensorThingsService/v1.0/Datastreams?
$filter=ObservedProperty/definition eq 'http://dbpedia.org/page/Water_level'
and properties/type eq 'measurement'
and st_within(Thing/Locations/location , geography 'POLYGON ((11.4 45.7,11.4 45.4,11.4
&$select=id ,name , properties , unitOfMeasurement
&$expand=
Thing (
$select=id ,name , description
;$expand=Locations
)
,Sensor ($select=id ,name , description )
,ObservedProperty ( $select=id ,name , definition , description )
,Observations (
$filter=phenomenonTime ge 2018-02-01T00:00:00Z and phenomenonTime lt 2018-03-01T00:00:00Z
;$orderby=phenomenonTime asc
;$select=phenomenonTime , result
)

```

**Figure A2.** Sample request to a SensorThings API server for retrieving a water level dataset

- 486 9. ISO/TC211. ISO 19156: 2011-Geographic information-Observations and measurements-International  
487 Standard. Geneva, Switzerland. *International Organization for Standardization*.
- 488 10. Botts, M.; Robin, A. OpenGIS sensor model language (SensorML) implementation specification. *OpenGIS  
489 Implementation Specification OGC* **2007**, 7.
- 490 11. EC. COMMISSION REGULATION (EU) No 1088/2010 of 23 November 2010 amending Regulation (EC)  
491 No 976/2009 as regards download services and transformation services. *Published in the official Journal on  
492 December 8, 2010* **2010**.
- 493 12. ISO/IEC 20922:2016, Information technology – Message Queuing Telemetry Transport (MQTT) v3.1.1.  
494 <https://www.iso.org/standard/69466.html>. Accessed: 2018-02-10.
- 495 13. Botts, M.; Percivall, G.; Reed, C.; Davidson, J. OGC® sensor web enablement: Overview and high level  
496 architecture. In *GeoSensor networks*; Springer, 2008; pp. 175–190.
- 497 14. Grothe, M.; Broecke, J.V.; Carton, L.; Volten, H.; Kieboom, R. Smart emission: Building a spatial data  
498 infrastructure for an environmental citizen sensor network **2016**.
- 499 15. Soto, J.A.; Werner-Kytölä, O.; Jahn, M.; Pullmann, J.; Bonino, D.; Pastrone, C.; Spirito, M. Towards a  
500 federation of smart city services. *International Conference on Recent Advances in Computer Systems  
501 (RACS 2015)*, 2016, pp. 163–168.
- 502 16. Trilles, S.; Luján, A.; Belmonte, Ó.; Montoliu, R.; Torres-Sospedra, J.; Huerta, J. SEnviro: a sensorized  
503 platform proposal using open hardware and open standards. *Sensors* **2015**, *15*, 5555–5582.
- 504 17. Huang, C.Y.; Wu, C.H. A Web Service Protocol Realizing Interoperable Internet of Things Tasking  
505 Capability. *Sensors* **2016**, *16*, 1395.
- 506 18. SensorThings API Part 2 - Tasking. <http://www.opengeospatial.org/pressroom/pressreleases/2739>.  
507 Accessed: 2018-04-24.
- 508 19. ISO/IEC 20802-1:2016. Open Data Protocol (OData). <https://www.iso.org/standard/69208.html>.  
509 Accessed: 2018-04-24.
- 510 20. INSPIRE Generic Conceptual Model. [https://inspire.ec.europa.eu/documents/Data\\_Specifications/D2.  
511 5\\_v3.4rc3.pdf](https://inspire.ec.europa.eu/documents/Data_Specifications/D2.5_v3.4rc3.pdf). Accessed: 2018-04-24.
- 512 21. OGC Coverage Implementation Schema (CIS) v1.1. [http://docs.opengeospatial.org/is/09-146r6/09-146r6.  
513 html](http://docs.opengeospatial.org/is/09-146r6/09-146r6.html). Accessed: 2018-04-24.
- 514 22. OGC Naming Authority. <http://www.opengis.net/def/observationType/OGC-OM/2.0/>. Accessed:  
515 2018-04-24.



- 516 23. Nogueras-Iso, J.; Zarazaga-Soria, F.J.; Béjar, R.; Álvarez, P.; Muro-Medrano, P.R. OGC Catalog Services: a  
517 key element for the development of Spatial Data Infrastructures. *Computers & Geosciences* **2005**, *31*, 199–209.
- 518 24. OpenAPI Specification. <https://github.com/OAI/OpenAPI-Specification>. Accessed: 2018-03-10.
- 519 25. Butler, H.; Daly, M.; Doyle, A.; Gillies, S.; Hagen, S.; Schaub, T. The geojson format. Technical report, 2016.
- 520 26. Web Feature Service 3.0 and Filter Encoding standards. [https://github.com/opengeospatial/WFS\\_FES](https://github.com/opengeospatial/WFS_FES).  
521 Accessed: 2018-03-10.