Towards an International Data Spaces Connector for the Internet of Things

Michael Nast, Benjamin Rother,
Frank Golatowski, Dirk Timmermann
Institute of Applied Microelectronics and Computer Engineering
University of Rostock
Rostock, Germany
{first name}.{surname}@uni-rostock.de

Jens Leveling, Christian Olms, Christian Nissen
Software and Information Engineering
Fraunhofer Institute for Material Flow and Logistics
Dortmund, Germany
{first name}.{surname}@iml.fraunhofer.de

***Abstract*—In the age of digitalization, data protection plays an important role. Data is created, modified and shared over the Internet between data owners and data users. A key issue in this context is the respect of data sovereignty. The International Data Spaces (IDS) Reference Architecture has been developed in order to preserve data sovereignty. In this regard, Internet of Things (IoT) devices, such as sensors, can play an important role for providing data. However, an IoT device is not capable of integrating directly into the IDS by default. We propose an approach to enable the IDS for vendor independent IoT devices, using an open interoperability standard for the IoT specified by the Open Geospatial Consortium (OGC).**

***Index Terms*—International Data Spaces (IDS), Internet of Things (IoT), Open Geospatial Consortium (OGC) SensorThings API, Message Queuing Telemetry Transport (MQTT), Microservices**

I. Introduction

In the past years there was a significant increase in the total number of devices, which are interconnected in the Internet of Things (IoT). This ongoing trend will lead to even more devices exchanging data over the internet in the future. According to a recent study, the number of devices connected to IP networks will increase from 3.9 in 2018 billion to 29.3 billion in 2023 [1]. The share of IoT devices will be around 14.7 billion devices, which is roughly 50 percent. The huge amount of produced data plays a central role. One challenge that has to be overcome here is interoperability between the devices. In most cases data comes from heterogeneous data sources and there is often no common understanding of the data. Besides the pure ability to exchange data between devices, this is primarily a matter of syntactic and semantic interoperability. This means the focus is on the format, e.g. JavaScript Object Notation (JSON), Extensible Markup Language (XML), Efficient XML Interchange (EXI), Comma Separated Values (CSV) (syntactic) and the correct meaning as well as appropriate usage (semantic) of the data [2]. Although this problem has not been solved comprehensively, a variety of approaches exist, e.g. the Semantic Sensor Network (SSN) Ontology [3], the Modular SSN Ontology [4] or the Media Types for Sensor Markup Language (SENML) [5]. In this work we focus on the Open Geospatial Consortium (OGC) SensorThings API [6], [7], which makes use of JSON encoding, defines its own data model and is based on well-known OGC standards.

Another aspect regarding data exchange in the IoT is data sovereignty. To share data between different parties, e.g. companies, governmental or scientific institutions, it is important for the data owners to retain control over their data. This is a problem that is to be solved by International Data Spaces (IDS). The combination of ensuring both device-level interoperability and maintaining data sovereignty is the main objective of this work. We propose an IDS-Connector for the IoT, which will be able to exchange data stored in OGC SensorThings API format.

This paper is structured as follows. In Section II we provide a description of the state of the art and related work for IDS and OGC SensorThings API. Then our proposed architecture and its components are described in Section III. Finally, in Section IV we give a brief summary and ideas for future work.

II. State of the Art & Related Work

A. Internet of Things Data

Four properties has been identified in [8] for describing IoT data characteristics, based on an extensive literature review of real use cases. These four properties are volume, diversity, traffic, and criticality. Data volume describes how complex and large the processed data are. Data variety denotes the diversity in the structure of the data used by an use case. Data traffic describes the traffic patterns that are required in an use case, influenced by the data volume, data variety and velocity of the data creation. Data criticality describes the requirement of an use case how fast a system must respond according to new data.

B. International Data Spaces

The International Data Spaces (IDS) Reference Architecture (former Industrial Data Space) for "link[ing] different cloud platforms through secure exchange and trusted sharing of data" as published by [9] aims to enable participants to exchange sensitive data. The architecture has two major goals, retaining control over how the data is used and to promote the innovative use of data for products, services and business models. In summary, the IDS offers participants the opportunity to

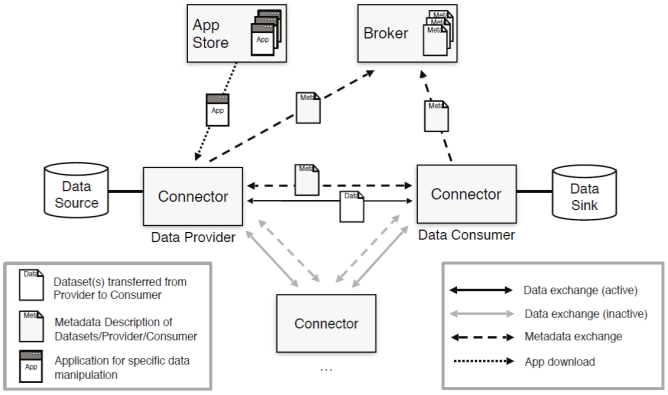

Fig. 1. IDS Reference Architecture [9]

exchange data for greater benefit while maintaining their data sovereignty. The core technical components and their interaction are shown in Fig. 1. A connector is a software component that links cloud environments and/or IoT devices to a data space. The architecture of the IDS consists of a distributed network of connectors, brokers (for searching connectors) and app stores (for providing connector functionality) without a centralized data storage. Each connector can define rules that the other connectors must fulfill to use its data.

The connectors are the gatekeepers between data and internal infrastructure and the data space. The broker only exchanges (machine readable) metadata of the data source and data usage policies. The actual data flows directly between connectors, not through a broker. Furthermore, the connectors load and execute data applications that encapsulate data processing or transformation services. Thus, a participant in a data space can technically control at his connector to what extent and with what accuracy his data is used by other participants.

C. OGC SensorThings API

The OGC SensorThings API is an open interoperability standard for the IoT, which is specified by the Open Geospatial Consortium (OGC). It is based and oriented on widely-used open standards such as OGC Sensor Web Enablement (SWE) standards, including Observation & Measurements (O&M) [10], Sensor Observation Service (SOS) [11] and Sensor Planning Service (SPS) [12], and OASIS Open Data Protocol (OData) [13]. To meet the requirements of resource-constrained IoT devices, it makes use of RESTful services and an efficient encoding based on JSON. Furthermore, it also enables event-based communication using Message Queuing Telemetry Transport (MQTT).

The standard is divided into two parts: the Sensing [6] and the Tasking [7] part. The Sensing part provides functions to collect data from heterogeneous data sources and store them in an uniform format. To query data OGC SensorThings makes use of RESTful Create, Read, Update, and Delete (CRUD) methods and OData. As depicted in Fig. 2, the core of the Sensing part is the *Thing* entity. A *Thing* can be understood as a thing in the sense of IoT. As defined in [7], a *Thing* is an object, either physical or virtual, being identifiable and integrable into a communication network. It is characterized by a *Location* and *HistoricalLocation*, which provide information about the current and previous locations of the *Thing*. A *Thing* can have one or multiple *Datastreams*. A *Datastream* is a set of *Observations* for the same *ObservedProperty* generated by the same *Sensor*. The standard defines an *Observation* as the act of measuring or determining the value of a property [6]. Here, the property is called *ObservedProperty*. The *FeatureOfInterest* defines on which target object the measurement was performed.

The Tasking part (see Fig. 2) defines the interaction with the IoT device and its physical environment, this includes, for example, controlling, setting, configuring and switching actions. A *Thing* can have one or multiple *TaskingCapabilities*. This entity type describes the capabilities of the device wrt. controlling it. Each *TaskingCapability* entity can be mapped to one *Actuator* and multiple *Tasks*. An actuator is the actual device that can be controlled. The *Task* describes the control task being performed by the the device. In the following we will focus on the Sensing part, as we are only interested in gathering data from sensors, which we can then store, process and exchange.

D. Related Work

When integrating sensor data, the heterogeneity of the devices must be taken into account. There are a large number of different sensor types, data models used, technologies for communication and the functionality of the devices as described in [14]. As shown in [14], an adapter and microservice concept can take this into account. For each device class, an adapter should be implemented that encapsulates the device specifics and transforms the data into a standardized model.

The IDS Reference Architecture [9] proposes a secure and interoperable data exchange in which the owner of the data retains control over how his data is used in order to keep data sovereignty. Due to its novelty, there are only a few works dealing with the IDS Reference Architecture so far. In [15] an approach for a trusted IDS-Connector is presented. It uses a microservices architecture and provides functionality to securely connect devices within the IDS based on a variety of mature mechanisms. An actual implementation of the trusted as well as a basic IoT-Connector implementation is given in [16]. Additionally, an information model is described. In [17] it is described how IDS can be enabled for the port sector. In particular, a first approach to the implementation of an IDS-Connector for IoT devices is given. The IDS has two basics roles, the data provider that is integrating data into a data space, and the data consumer that receives and uses data from a data space. Both roles can be combined.

III. Concept

A. Architecture

Figure 3 shows a high level overview of our architecture. On the left side, IoT devices are sending messages that are integrated via IoT-Connectors into the data space. In our

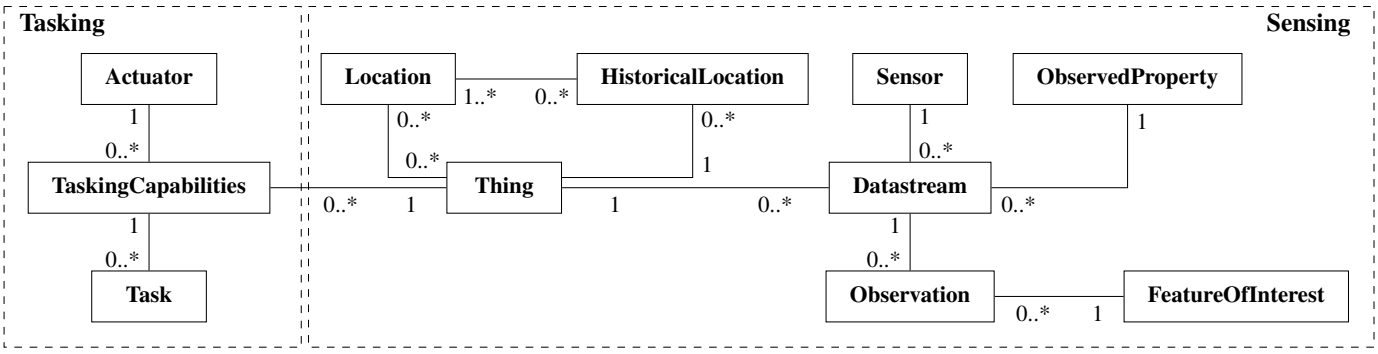

Fig. 2. Simplified version (attributes left out for brevity) of the OGC SensorThings Data Model [6], [7]

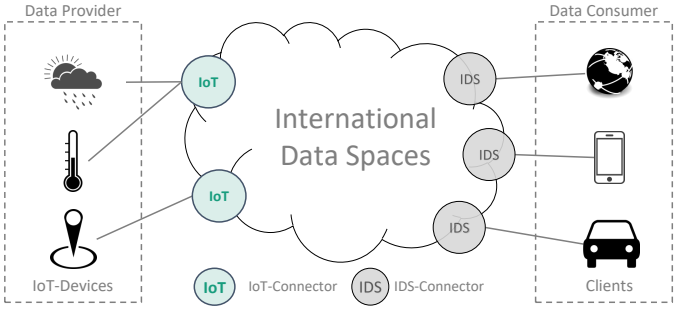

Fig. 3. Architecture model, consists of IoT devices as data provider and clients as data consumer

architecture, IoT devices are only providing data. In terms of IDS they are pure data provider. On the other side, several clients are retrieving IoT data from the data space. In our architecture, they are data consumers, each having its own IDS-Connector. The focus of our work is on the IoT devices and the IoT-Connector. By default, an IoT device is not able to integrate with the IDS directly. Additionally, IoT devices are sending data via proprietary protocols and in binary formats. The IoT-Connector contains necessary functionality to address these issues and integrate IoT data into the IDS in a valid format and standard. The next section III-B describes this in detail. With regard to IDS, the focus of our IoT-Connector is on basic communication functions. For the sake of simplicity, other concepts, such as interaction with brokers, app stores or apps, are left out for now.

B. IoT-Connector

The IoT-Connector, based on [14], is designed to enable sharing of data from IoT devices. The IoT-Connector is currently a pure data provider as our focus is on the sensing part. The connector uses microservices as core architecture pattern to enable scalability for handling asynchronous messages from up to 100.000 devices (this amount is normal for use cases like container tracking) at once. In mobile scenarios, low power devices are used, which are sending short messages using binary formats. The connector encapsulates the communication with the devices in the respective protocols and transforms the messages into open standards. In this context we have chosen the lightweight JSON and open protocols. We are using MQTT to enable event-driven communication by means of publish/subscribe. HTTP/RESTful communication is used for function calls and to retrieve data from a database. Other protocols and standards, like XML and Advanced Message Queuing Protocol (AMQP), are subject of further investigation for our ongoing development. Furthermore, the target data format of our connector implements a semantic model for sensor data as provided by the IDS Information model. This model is based on OGC and SSN [5], [3] and [4]. Using a semantic data model for sensor data is highly recommend within a data space to support a common understanding of the inherent information, like origin, use case, possible value ranges, and other information related to the data by any data consumer.

C. Data Provider

Our IDS Data Provider (see Fig. 4) follows an event-driven approach. We use MQTT for communication with our sensor devices, as well as for IDS communication. Thus, the MQTT Broker is the key component of our IDS Provider. On device level we can either connect to a specific device directly, or via a gateway. The latter allows us to integrate devices that communicate on the basis of other protocols, like the Constrained Application Protocol (CoAP) or MQTT for Sensor Networks (MQTT-SN). OGC SensorThings API serves as interoperability layer. In our implementation we use the Fraunhofer Open Source SensorThings (FROST) API Server [18]. This includes a database, a web interface, a Representational state transfer (REST) and an MQTT interface, allowing us to transmit, store and query our sensor data, based on a well-defined data format. Additionally, we can receive event notifications using MQTT. Although the focus is on event-based communication, sensor data can also be accessed via REST API.

The interface between our data provider and the IDS is the IDS-Adapter. It is responsible for handling connections to the IDS-Adapter of the data consumer and to receive event notifications or query data from the OGC Server. For now, there are two different options. The IDS-Adapter can either use MQTT to get event notifications, or REST to access historical data (e.g. stored in a database). To handle multiple

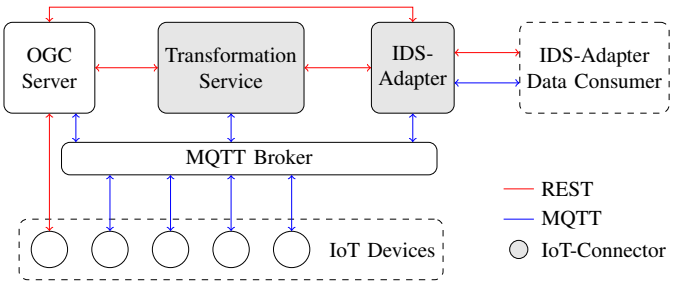

Fig. 4. IDS Data Provider

connections simultaneously, we instantiate one IDS-Adapter per IDS-Consumer. The second component, which is important for the communication with the IDS, is the Transformation Service. It transforms our sensor data from OGC SensorThings API compliant format into our IoT semantic data model, as mentioned in Section III-B. However, it is also possible to exchange data directly via the OGC SensorThings API format. Thus, the Transformation Service is an optional component.

D. Data Consumer

A Data Consumer connects to an IoT-Connector for retrieving IoT data using its own IDS-Connector. It can subscribe on the connector to get event notifications via MQTT or call an HTTP/RESTful interface to get data provided by the IDS-Adapter component of our IoT-Connector. The next step is depending on the use case. For example, the data can be analyzed using data mining or machine learning techniques to detect process anomalies and serve as a basis for decision-making in concrete business processes.

IV. Summary & Outlook

By default, IoT devices cannot be integrated in the IDS. In this paper we have proposed an approach to enable the IDS Reference Architecture for sharing sensor data provided by IoT devices. The IoT-Connector has been designed for this purpose. It is based on the IDS-Connector and acts as an interface between IoT devices and the IDS. The OGC SensorThings API has been proposed as an interoperability layer between the IoT-Connector and IoT devices in order to be able to use vendor independent IoT devices in this context. Our IoT-Connector can either communicate directly using the OGC SensorThings format or using our proposed semantic data model. Here the transformation is performed by a data transformation service.

Future work will focus on the implementation of the IDS with its IoT-Connector for concrete industrial applications. In order not to be limited to OGC, the integration of further data models and transformation services will be enabled in the future. We also plan to implement a consumer to perform further data processing, like data mining and machine learning, on the exchanged data.

Acknowledgment

This work has been achieved in the European ITEA project Intelligent, IoT-based Port Artefacts Communication, Administration and Maintenance (I2PANEMA) and has been funded by the German Federal Ministry of Education and Research (BMBF) under reference number 01IS18058H.

References

[1] Cisco, "Cisco Annual Internet Report (2018–2023), White Paper," https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html, 2020, Accessed on: 2020-04-05.

[2] M. Noura, M. Atiquzzaman, and M. Gaedke, "Interoperability in internet of things infrastructure: classification, challenges, and future work," in *International Conference on Internet of Things as a Service*. Springer, 2017, pp. 11–18.

[3] "Semantic sensor network ontology," Oct 2017. [Online]. Available: https://www.w3.org/TR/vocab-ssn/

[4] A. Haller, K. Janowicz, S. Cox, M. Lefranois, K. Taylor, D. Phuoc, J. Lieberman, R. Garca Castro, R. Atkinson, and C. Stadler, "The modular ssn ontology: A joint w3c and ogc standard specifying the semantics of sensors, observations, sampling, and actuation," *Semantic Web*, vol. 10, 08 2018.

[5] C. Jennings, "Media types for sensor markup language (senml)," Oct 2012. [Online]. Available: https://tools.ietf.org/id/draft-jennings-senml-10.txt

[6] S. Liang, C.-Y. Huang, and T. Khalafbeigi, "OGC SensorThings API Part 1: Sensing Version 1.0," http://docs.opengeospatial.org/is/15-078r6/15-078r6.html, 2016, Accessed on: 2020-02-27.

[7] S. Liang and T. Khalafbeigi, "OGC SensorThings API Part 2: Tasking Core Version 1.0," http://docs.opengeospatial.org/is/17-079r1/17-079r1.html, 2019, Accessed on: 2020-02-27.

[8] T. P. Raptis, A. Passarella, and M. Conti, "Data management in industry 4.0: State of the art and open challenges," *IEEE Access*, vol. 7, pp. 97 052–97 093, 2019.

[9] B. Otto, S. Lohmann, S. Auer, J. Cirullies, A. Eitel, T. Ernst, C. Haas, M. Huber, J. Jürjens, C. Lange, C. Mader, N. Menz, R. Nagel, H. Pettenpohl, J. Pullmann, C. Quix, J. Schon, D. Schulz, J. Schütte, M. Spiekermann, and S. Wenzel, "Reference architecture model for the international data spaces," International Data Spaces Association, Tech. Rep., 2019. [Online]. Available: https://www.fraunhofer.de/content/dam/zv/en/fields-of-research/industrial-data-space/IDS-Reference-Architecture-Model.pdf

[10] S. Cox, "OGC Abstract Specification Geographic information Observations and measurements Version 2.0," Open Geospatial Consortium (OGC), Tech. Rep., 2013.

[11] A. Bröring, C. Stasch, and J. Echterhoff, "OGC Sensor Observation Service Interface Standard Version 2.0," Open Geospatial Consortium (OGC), Tech. Rep., 2012.

[12] I. Simonis and J. Echterhoff, "OGC Sensor Planning Service Implementation Standard Version 2.0," Open Geospatial Consortium (OGC), Tech. Rep., 2011.

[13] M. Pizzo, R. Handl, and M. Zurmuehl, "OData Version 4.0. Part 1: Protocol Plus Errata 03," Organization for the Advancement of Structured Information Standards (OASIS), Tech. Rep., 2016.

[14] J. Leveling, L. Weickhmann, C. Nissen, and C. Kirsch, "Event-driven architecture for sensor data integration for logistics services," in *2018 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, Dec 2018, pp. 536–540.

[15] J. Schütte, G. Brost, and S. Wessel, "Der trusted connector im industrial data space," *arXiv preprint arXiv:1804.09442*, 2018.

[16] I. D. S. Initiative, "Open Source Repositories of the Industrial Data Space initiative," https://github.com/industrial-data-space, Accessed on: 2020-03-05.

[17] D. Sarabia-Jcome, I. Lacalle, C. E. Palau, and M. Esteve, "Enabling industrial data space architecture for seaport scenario," in *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, April 2019, pp. 101–106.

[18] M. Jacoby and H. van der Schaaf, "Fraunhofer Opensource SensorThings (FROST) Server," https://github.com/FraunhoferIOSB/FROST-Server, Accessed on: 2020-02-27.