



Evaluation of Application Programming Interfaces for INSPIRE

Katharina Schleidt, Hylke van der Schaaf, Jürgen Moßgraber, Sylvain Grellet, Nuno Oliveira

2022



This publication is a Technical report by the Joint Research Centre (JRC), the European Commission's science and knowledge service. It aims to provide evidence-based scientific support to the European policymaking process. The scientific output expressed does not imply a policy position of the European Commission. Neither the European Commission nor any person acting on behalf of the Commission is responsible for the use that might be made of this publication. For information on the methodology and quality underlying the data used in this publication for which the source is neither Eurostat nor other Commission services, users should contact the referenced source. The designations employed and the presentation of material on the maps do not imply the expression of any opinion whatsoever on the part of the European Union concerning the legal status of any country, territory, city or area or of its authorities, or concerning the delimitation of its frontiers or boundaries.

Contact information

Name: Alexander Kotsev
Address: Via E. Fermi, 2749 – TP 263 - 21027 Ispra (VA), Italy
Email: alexander.kotsev@ec.europa.eu
Tel.: +39 0332 78 9069

EU Science Hub

<https://ec.europa.eu/jrc>

JRC128885

PDF ISBN 978-92-76-49644-1 doi:10.2760/00811

Luxembourg: Publications Office of the European Union, 2022

© European Union, 2022



The reuse policy of the European Commission is implemented by the Commission Decision 2011/833/EU of 12 December 2011 on the reuse of Commission documents (OJ L 330, 14.12.2011, p. 39). Except otherwise noted, the reuse of this document is authorised under the Creative Commons Attribution 4.0 International (CC BY 4.0) licence (<https://creativecommons.org/licenses/by/4.0/>). This means that reuse is allowed provided appropriate credit is given and any changes are indicated. For any use or reproduction of photos or other material that is not owned by the EU, permission must be sought directly from the copyright holders.

All content © European Union, 2022, except: *front page* by [Conny Schneider](#) on Unsplash.

How to cite this report: Schleidt, K., Van der Schaaf, H., Moßgraber, J., Grellet, S., Oliveira, N. *Evaluation of Application Programming Interfaces for INSPIRE*, Publications Office of the European Union, Luxembourg, 2022, ISBN 978-92-76-49644-1, doi:10.2760/00811 JRC128885.

Contents

- Executive summary 1
- 1 Introduction..... 3
- 2 Methodology for evaluation of standard based APIs..... 4
 - 2.1 Introduction..... 4
 - 2.1.1 Evaluation Process Overview 4
 - 2.1.2 Goals..... 6
 - 2.1.3 Evaluation Dimensions - Degrees of Freedom 6
 - 2.1.4 Evaluation Components 7
 - 2.1.5 Evaluation Process..... 8
 - 2.2 Stakeholder Analysis..... 8
 - 2.2.1 Stakeholder Overview 9
 - 2.2.2 Provision Spectrum..... 11
 - 2.2.3 Usage Spectrum 12
 - 2.2.4 Stakeholder Perspectives 13
 - 2.3 Evaluation Criteria - Five Level Open Data API evaluation model..... 13
 - 2.3.1 Level 1: All find..... 16
 - 2.3.2 Level 2: All use..... 17
 - 2.3.3 Level 3: All trust 19
 - 2.3.4 Level 4: All involved 21
 - 2.3.5 Level 5: All develop 22
 - 2.3.6 SQuaRE: ISO 25010 Model 24
 - 2.4 Evaluation Methods..... 24
 - 2.4.1 Evaluation Types..... 24
 - 2.4.2 API Development and Deployment 27
 - 2.4.3 API Use 28
 - 2.5 Evaluation Process..... 30
 - 2.5.1 Project Methodology Design..... 32
 - 2.5.2 Consultations..... 32
 - 2.5.3 Analysis..... 33
 - 2.5.4 Recommendations..... 36
- 3 Deployment strategies for standard-based APIs..... 37
 - 3.1 Introduction..... 37
 - 3.2 Data Providers..... 37
 - 3.2.1 Selection Criteria..... 37
 - 3.2.2 Data Nests 38
 - 3.3 Deployment Methodology 39
 - 3.3.1 GeoServer Deployment..... 42

| | | |
|-------|---|----|
| 3.3.2 | OGC API – Features Development - OGC API Simple..... | 46 |
| 3.3.3 | LD-Proxy | 47 |
| 3.3.4 | FROST Deployment..... | 50 |
| 3.4 | Data Provider Specific Deployment..... | 51 |
| 3.4.1 | Austrian Meteorological Agency – ZAMG (AT)..... | 51 |
| 3.4.2 | Austro Control – ACG (AT)..... | 52 |
| 3.4.3 | Umweltbundesamt - UBA (AT) & European Environment Agency - EEA (EU)..... | 52 |
| 3.4.4 | City of Hamburg – CH (DE)..... | 53 |
| 3.4.5 | French Geological Survey – BRGM (FR)..... | 53 |
| 3.4.6 | French Office for Biodiversity (OFB) – INSIDE - environmental information systems research centre (FR)..... | 54 |
| 3.4.7 | Environment Agency Baden-Württemberg – LUBW (DE)..... | 55 |
| 3.4.8 | Summary Overview..... | 55 |
| 3.5 | Conclusions..... | 56 |
| 3.6 | Next Steps..... | 56 |
| 4 | Technical documentation, source code and working prototype Information system and documentation..... | 57 |
| 4.1 | Introduction..... | 57 |
| 4.2 | Technical Documentation..... | 57 |
| 4.2.1 | OGC API – Features..... | 57 |
| 4.2.2 | SensorThings API..... | 57 |
| 4.3 | Working Prototype Documentation..... | 58 |
| 4.4 | Source Code..... | 58 |
| 4.4.1 | FROST-Server..... | 58 |
| 4.4.2 | SensorThings API Tools..... | 58 |
| 4.4.3 | GeoServer..... | 58 |
| 4.4.4 | OGC API Simple..... | 59 |
| 5 | API Evaluations..... | 60 |
| 5.1 | Evaluation Outcomes Development OGC API - Features..... | 60 |
| 5.1.1 | General Information on API Experience..... | 60 |
| 5.1.2 | Level 1: All find..... | 61 |
| 5.1.3 | Level 2: All use..... | 62 |
| 5.1.4 | Level 3: All trust..... | 62 |
| 5.1.5 | Level 4: All involved..... | 62 |
| 5.1.6 | Level 5: All develop..... | 63 |
| 5.2 | Evaluation Outcomes Deployment OGC API - Features..... | 63 |
| 5.2.1 | General Information on API Experience..... | 63 |
| 5.2.2 | Level 1: All find..... | 64 |
| 5.2.3 | Level 2: All use..... | 65 |
| 5.2.4 | Level 3: All trust..... | 65 |
| 5.2.5 | Level 4: All involved..... | 66 |

| | | |
|-------|--|----|
| 5.2.6 | Level 5: All develop | 66 |
| 5.3 | Evaluation Outcomes Development SensorThings API (STA) | 66 |
| 5.3.1 | General Information on API Experience | 66 |
| 5.3.2 | Level 1: All find | 67 |
| 5.3.3 | Level 2: All use | 67 |
| 5.3.4 | Level 3: All trust | 68 |
| 5.3.5 | Level 4: All involved | 68 |
| 5.3.6 | Level 5: All develop | 68 |
| 5.4 | Evaluation Outcomes Deployment STA | 69 |
| 5.4.1 | General Information on API Experience | 69 |
| 5.4.2 | Level 1: All find | 69 |
| 5.4.3 | Level 2: All use | 69 |
| 5.4.4 | Level 3: All trust | 70 |
| 5.4.5 | Level 4: All involved | 70 |
| 5.4.6 | Level 5: All develop | 70 |
| 5.5 | Evaluation Outcomes Use STA | 71 |
| 5.5.1 | General Information on API Experience | 71 |
| 5.5.2 | Level 1: All find | 71 |
| 5.5.3 | Level 2: All use | 72 |
| 5.5.4 | Level 3: All trust | 72 |
| 5.5.5 | Level 4: All involved | 72 |
| 5.5.6 | Level 5: All develop | 73 |
| 6 | Conclusions | 74 |
| 6.1 | Evaluation Overview | 74 |
| 6.2 | General Issues | 75 |
| 6.2.1 | Cross-Border Alignment | 75 |
| 6.2.2 | Linking OAPIF and STA | 76 |
| 6.2.3 | Timelines and Transparency | 78 |
| 6.3 | Recommendations for Further Work | 78 |
| 6.3.1 | Error Handling | 78 |
| 6.3.2 | Data License Information | 78 |
| 6.3.3 | Validation and JSON Schema | 78 |
| 6.3.4 | STA Query Builder | 79 |
| 6.3.5 | Cross Linking between Representations | 79 |
| 6.3.6 | Client Support | 79 |
| 6.3.7 | Examples! | 80 |
| 6.4 | Recommendations for Member States | 80 |
| 6.4.1 | Considerations towards OAPIF | 81 |
| 6.4.2 | Considerations towards STA | 82 |

| | |
|---|-----|
| List of abbreviations and definitions | 83 |
| Acronyms | 83 |
| Definitions..... | 83 |
| Standards | 84 |
| Supporting Documents | 84 |
| List of figures | 85 |
| List of tables..... | 86 |
| Annexes | 87 |
| Annex A - Extended evaluation criteria..... | 87 |
| Annex B - SQuaRE: ISO 25010 Model..... | 97 |
| Annex C Mapping INSPIRE AQD to SensorThings API | 100 |

Acknowledgements

The “API4INSPIRE“ study was funded in the frame of the European Location Interoperability Solutions for e-Government action (ELISE)¹, part of the ISA² Programme. The ELISE Action supports Better Regulation and Digital Single Market Strategy goals, including specific actions of the e-Government Action Plan² and the European Interoperability Framework³, which are reinforced by the Tallinn Declaration⁴ vision and the Communications on Building the Data Economy⁵ and on Artificial Intelligence for Europe⁶.

In particular, ELISE studies explore the role of location information in digital government and the technologies involved in delivering innovative public services. Is worth mentioning, the “Digital Government Benchmark – API study”⁷ providing an early-stage analysis of Web APIs as enablers for the digital transformation of government. A multiple-case study comparative analysis has been applied to selected cases, with a particular focus on geospatial API.

Authors

Katharina Schleidt

Hylke van der Schaaf

Jürgen Moßgraber

Sylvain Grellet

Nuno Oliveira

1 https://ec.europa.eu/isa2/actions/elise_en

2 <https://ec.europa.eu/digital-single-market/en/european-egovernment-action-plan-2016-2020>

3 https://ec.europa.eu/isa2/eif_en

4 http://ec.europa.eu/newsroom/document.cfm?doc_id=47559

5 https://eur-lex.europa.eu/content/news/building_EU_data_economy.html

6 <https://ec.europa.eu/digital-single-market/en/news/communication-artificial-intelligence-europe>

7 <https://joinup.ec.europa.eu/collection/elise-european-location-interoperability-solutions-e-government/document/report-digital-government-benchmark-api-study>

Abstract

The EC has a long history of promoting open access to public data across Europe, breaking down electronic barriers at national borders through the creation of common data and service models, as well as through the provision of accompanying legislation facilitating such endeavours. The INSPIRE Directive has been a core building block in this work, which has been further elaborated within the “European Union Location Framework (EULF)” and “A Reusable INSPIRE Reference Platform (ARe3NA)”.

The API4INSPIRE project investigated new developments in geospatial standards and technologies, foremost the new OGC API – Features and SensorThings API standards, together with the outcomes of the INSPIRE MIG Action 2017.2 on alternative encodings for INSPIRE data, evaluating their suitability for use in the European spatial data landscape described above. For this purpose, an evaluation strategy was developed to determine how these new and emerging standards can best be utilized to leverage existing investments by EU Member States in INSPIRE implementation, as well as supporting new developments in e-Government and the Digital Single Market. The evaluation focused on various facets of usability ranging from the configuration and deployment aspects of service deployment to ease of uptake of the API, and included a wide range of stakeholders within the evaluation process. The outcomes of this investigation were analysed, relevant guidance materials were created based on insights gained, and were widely disseminated to interested stakeholders. In addition, various Open Source software solutions as well as extensions were developed where gaps were identified.

Executive summary

The European Commission has a long history of promoting open access to public data across Europe. The INSPIRE Directive is a core building block in this work, creating common data and service models together with governance modalities, relying strongly on standards for the (meta)data models and network services specified for data provision. At the time of writing, many of the Open Geospatial Consortium (OGC) standards utilised for the specification of INSPIRE were being updated, with new versions developed based on more modern API paradigms. These new standards promise to be easier to use, from (i) the perspective of the software engineer writing data provision implementations, (ii) of the data provider using these implementations to provide data, and (iii) of the user accessing the data. The goal of the API4INSPIRE study synthesised in this report was to evaluate two prominent new standards - OGC API – Features and OGC SensorThings API, and determine if these live up to their promise when it comes to INSPIRE data provision. As both APIs are developed by OGC, their development process is well monitored by many interested parties and the overall quality of the standards is assured. Both APIs were deemed valuable additions to the INSPIRE ecosystem.

For this purpose, an evaluation strategy was developed suited to determine how these new and emerging standards can best be utilised to leverage existing investments by EU Member States in the INSPIRE implementation. The Evaluation Criteria used were derived from the Five Level Open Data API evaluation model by Jarkko Moilanen, which itself follows the logic in Tim Berners-Lee's 5-star deployment scheme for Open Data. They are defined in increasing levels of maturity as follows:

- **Level 1: All find** - This level corresponds to very basic non-standardized data provision, data is available together with simple examples.
- **Level 2: All use** - This level corresponds to basic standardized data provision.
- **Level 3: All trust** - This level corresponds to mature standardized data provision.
- **Level 4: All involved** - This level pertains more to the maturity of the wider ecosystem built up around the standardized specifications.
- **Level 5: All develop** - This level describes a well developed and mature ecosystem built up around the standardized specifications.

For all of these levels, a higher level of maturity translates to less effort required when dealing with the various aspects of development, deployment and use of an API. Diverse evaluation criteria were defined for each of these levels, and scored pertaining to each of the three different stakeholder perspectives:

- **API Development:** Concerning the role of a software engineer, responsible for developing an implementation of the API Standard on the server side.
- **API Deployment:** Concerning the role of a data provider, responsible for publishing through the API by utilizing and configuring existing implementations.
- **API Use:** Concerning the role of API consumers.

Next, API endpoints were deployed together with data providers from a wide range of thematic areas, that in turn were made available to developers for experimentation with these new data provision modalities. Six data providers from Germany, France and Austria have contributed data, staff and infrastructure. Together, these data providers manage data from 14 INSPIRE themes. Together with the data providers, various related datasets were provided via OGC API – Features and SensorThings API endpoints. This data has been made openly available for exploration and application development. The data sources were grouped into three 'data nests':

- **Airy Austria**
 - Air Transport Network plus Meteorological Conditions.
 - Air Quality via SensorThings.
 - Compare and merge OGC API – Features and SensorThings API.
- **Urban Data Platform Hamburg**
 - Huge volumes sensor and geo data on the bottom, applications on the top, and APIs in between.

- One possible use-case combines the locations of the electro rollers with the road network data.
- **Franco-Germanic Flows**
 - Cross-border water data providing alternative perspectives on the Rhine.
 - Various other complementary French data sources.

In addition to these initial data providers, several ad-hoc data sources were added by the study as opportunities arose. Some of these sources resulted in a novel use of the SensorThings API:

- European NUTS regions in multiple resolutions, allowing clients to choose the resolution that best fits the application.
- Yearly Population statistics for European NUTS regions.
- Yearly Population statistics for the European 1km grid.
- COVID19 case data.
- European air quality data, sourced from the European Environmental Agency.

Feedback on the experiences gained with these new APIs was collected via interviews and surveys in a structure tightly aligned with the evaluation criteria specified for each of the maturity levels, providing insights into all aspects of API development, deployment and usage. Based on this feedback, an evaluation result on a scale from one (good) to three (bad) was determined for each criterion.

The results for the OGC SensorThings API were very positive, with most scores being good (72%) with only relatively few intermediate (25%) and only 1 bad (1%). The results for OGC API -- Features are only slightly less positive, with 27 good (42%), 32 intermediate (49%) and 6 bad (9%).

Available for more than five years, the OGC SensorThings API Standard is mature and proven standard, utilised in diverse contexts since its finalisation 2016. There are multiple implementations, both open-source and closed-source, on both server- and client-side. Conversely, OGC API – Features is a young standard that does not yet provide all functionality available via WFS, but looks very promising and for many use cases easier to implement than a WFS instance. Implementation of the basic server functionality is relatively easy, as demonstrated by the OGC-API Simple implementation that was developed for this project. However, as much of the more advanced functionality planned for future extensions to the standard was not specified at the time of this evaluation, commenting on the implementation complexity of future versions of OGC API-Features is not possible. This is especially true for query/search functionality that goes beyond simple bounding-box or time queries.

1 Introduction

The EC has a long history of promoting open access to public data across Europe, breaking down electronic barriers at national borders through the creation of common data and service models, as well as through the provision of accompanying legislation facilitating such endeavours. The INSPIRE Directive has been a core building block in this work, which has been further elaborated within the “European Union Location Framework (EULF)” and “A Reusable INSPIRE Reference Platform (ARe3NA)”. This foundation is now being leveraged by the European Location Interoperability Solutions for e-Government (ELISE) action, further broadening the scope in terms of both data scope and service provision technologies. This will go a long way towards widespread deployment of machine actionable services and enablement of data analytics across data sources.

API4INSPIRE investigated new developments in geospatial standards and technologies, foremost the new OGC API – Features and SensorThings API standards, together with the outcomes of the INSPIRE MIG Action 2017.2 on alternative encodings for INSPIRE data, evaluating their suitability for use in the European spatial data landscape described above. For this purpose, an evaluation strategy was developed to determine how these new and emerging standards can best be utilized to leverage existing investments by EU Member States in INSPIRE implementation, as well as supporting new developments in e-Government and the Digital Single Market. The evaluation focused on various facets of usability ranging from the configuration and deployment aspects of service deployment to ease of uptake of the API, and included a wide range of stakeholders within the evaluation process. The outcomes of this investigation were analysed, relevant guidance materials were created based on insights gained, and were widely disseminated to interested stakeholders.

While usability evaluation has long been an established practice pertaining to Human Computer Interaction (HCI), this has often been neglected in regard to APIs, alternatively subsumed within a more general code review process. Research has shown that a well-structured API usability evaluation can ameliorate issues in the implementation phase, and greatly increase uptake of the API.

The evaluation was initially foreseen as an amalgam of methodologies ranging from preliminary heuristic analysis and peer reviews over hackathons to questionnaires and interviews with both hackathon participants as well as data provider staff involved in the deployment and uptake of the new OGC services. Due to restrictions imposed by the Covid-19 pandemic, various aspects of the evaluation process had to be modified, with all interaction shifting to the virtual arena. Conversely, this crisis also triggered creativity, leading to novel applications of the APIs under investigation that will be carried on by other projects. All outputs of the evaluation process were thoroughly analysed, insights gained and lessons learned documented, recommendations formulated.

Based on initial interaction with data providers and users, requirements for additional support materials were determined early on in the evaluation process and provided via the API4INSPIRE web pages; these supporting documents include tutorials technical guidance, software tools, tutorials, webinar presentations, and based on our experience as exceedingly important, simple well documented examples, giving developers a good basis for rapid uptake of these new technologies. This will enable both data providers in the simpler provision of their data, as well as empowering potential developers and users to make the most of the data and services offered.

2 Methodology for evaluation of standard based APIs

2.1 Introduction

This section reports on the methodology used to evaluate the introduction of standard based APIs into a wider administrative data provision framework as already established within the European Community. The methodology is generic and reusable for various stakeholder groups including both data providers and data users. The methodology includes the criteria deemed as relevant for evaluation of the transition towards the newly proposed OGC APIs, namely the OGC-API Features, and for dynamic data, the SensorThings API for both data providers and users, as well as methods for gaining insights pertaining to these criteria.

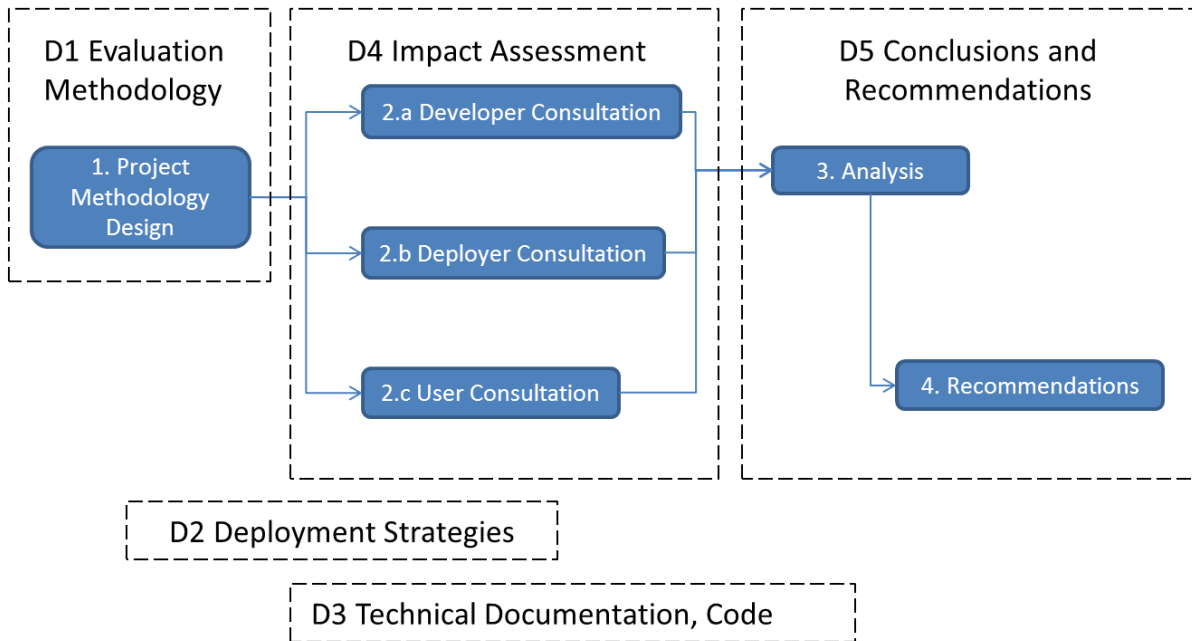
The evaluation methodology described has been designed to weigh costs and benefits against each other, highlighting both strengths and weaknesses of the APIs being evaluated. Pertaining to benefits, this includes flexibility, developer friendliness, and ease of discoverability, access and use. From a technical point of view, alignment with the current architecture of the Web and the Spatial Data on the Web Best Practices should be assured. Cost considerations, such as infrastructural changes, need for additional expertise, updated tooling, training, reengineering of existing practices and security aspects shall be included. While quantifiable metrics would be preferable, the effort entailed in gaining truly representative values would require a level of complexity that outweighs the assumed benefits of such quantification; thus, we have chosen to focus on qualitative metrics that can be easily abstracted to different operational environments.

The outcomes of this evaluation will provide guidance for the diverse stakeholders involved in this complex data ecosystem that incorporates such initiatives as INSPIRE. This includes the different roles involved in data provision and use, ranging from administrative stakeholders tasked with securing the necessary resources for deployment of APIs over domain experts concerned that their data is handled and provided correctly to developers designing new applications based on the available APIs. The ecosystem further encompasses not only the governance aspects entailed in maintaining and updating such an infrastructure, but also enabling community initiatives and platforms, as well as the standards bodies defining and updating the underlying standards.

2.1.1 Evaluation Process Overview

In this section, we provide an overview of the evaluation methodology. At the same time, we show how this methodology is embedded within the wider context of the evaluation process. In Figure 1: Evaluation Process Overview we illustrate how the individual steps of the evaluation process are aligned with the project deliverables.

Figure 1: Evaluation Process Overview



In an initial step, we have designed the project methodology. Relevant stakeholder groups have been analysed, evaluation criteria defined and evaluation methods specified and described. Based on this foundation, the evaluation process has been formulated. This is the topic of the current section “Methodology for evaluation of standard-based APIs”.

In parallel with the design of the project methodology, work on deployment strategies has commenced with the various data providers linked with this project. Based on the available data resources potential use cases are being defined, guiding our selection of data sources to be exposed, whereby we aim to assure collections of related data assuring spatiotemporal consistency. This work will be documented and provided as “D2 Deployment Strategies”.

Based on the deployment strategy, we shall begin implementing and deploying both the OGC API – Features and SensorThings API together with our data providers. During this process, feedback on issues encountered will be documented, in order to better understand what support data providers are currently missing; this feedback will provide complementary information to that collected through questionnaires and interviews during the formal consultation process. All technical documentation together with source code and operational prototypes will be made available as “D3 Technical Documentation, Code”.

The impact of the introduction of the new APIs will be investigated through stakeholder consultations. The methodology described in this document will be utilized in order gain insights into the experiences of the developers, deployers and users involved. This process will be staggered, with developer and deployer consultations taking place earlier on in the process as the APIs are being made available. Once the APIs are openly available, users will be invited to experiment with these resources in the context of various events and further feedback gained. This work will be documented in “D4 Impact Assessment”.

After the consultations of the impact assessment have been finalized, the outcomes of the assessment will be analysed together with inputs gained during the development and deployment process to provide insights into the strengths and weaknesses of the APIs under evaluation. As part of this analysis, effort required for API deployment will be compared to that required for the provision of more traditional services; a similar comparison will be performed pertaining to data usage aspects. Recommendations will be provided both towards the stakeholder groups identified on how to make best use of their resources for API provision as well as to the responsible standardization bodies on desired extensions to the API standards. All materials identified as relevant for the development, deployment and use of the OGC APIs will be collected and structured on the

project GitHub⁸; the content can later be transferred to an alternative platform if required. These resources will be extended in order to provide a solid foundation for webinar participants to rapidly acquire the skills required to both provide and utilize APIs. All outputs will be provided as “D5 Conclusions and recommendations for MS authorities”.

2.1.2 Goals

In order to gain a better understanding of the evaluation target, we must first clearly specify the goals of the evaluation process. The high level evaluation goal pertains to gaining a better understanding of the impacts and benefits of both

1. establishing the emerging APIs as valid INSPIRE download services as well as
2. where deemed appropriate based on the underlying data models and foreseen use cases, introducing simplifications within the INSPIRE data models as foreseen within the INSPIRE MIF activity 2017.2 on alternative encodings within the INSPIRE and beyond domain.

Based on these two high level targets, we began the process of refining these to their component parts. This analysis provides an overview of the “degrees of freedom” available within this complex data ecosystem, which in turn provides a robust framework for weighing costs and benefits pertaining to each of these dimensions, and providing well-founded recommendations.

An evaluation is always a comparison - the important question is against what explicitly are we evaluating? Based on the wider context, it became clear that the introduction of the OGC API – Features and SensorThings API together with potential data model simplification must be compared to the existing SOAP-Like OGC Web Services, Data Models and Ecosystems. Estimates of costs and benefits must be seen on a relative scale in comparison to these pertaining to existing technologies. This relative approach also allows us to provide valid findings on a smaller sample set. Providing absolute numbers for effort would require a very large sample set and result in fuzzy intervals, as effort varies greatly with the level of expertise. Providing effort relative to known tasks allows each reader to apply these values to their own experiences.

A final goal of this project pertains to the identification of gaps in information and supporting materials encountered in the course of the evaluation process. All identified gaps must be documented, required materials collected or prepared and made available. Based on these supporting materials, training webinars will be designed.

2.1.3 Evaluation Dimensions - Degrees of Freedom

For a well-founded evaluation, in addition to an in-depth understanding of the requirements, we must also know what we are evaluating against, i.e. what is the current baseline, as well as our possible axes of intervention, i.e. in which directions can we recommend modifications or further actions. The requirements we are evaluating against ensue from the previous sections. Our degrees of freedom are as follows:

- **API Specification.** While the specification for the SensorThings API Standard is fairly settled, with the V1.1 update forthcoming, the OGC API – Features remains a work-in-progress, with the core specification newly finalized and essential functionality such as filter/query still on the horizon. This challenging newness of the OGC API – Features can be seen as a benefit, as insights gained within this evaluation can be directly fed back into the OGC through the various members within the project team. The same holds true for the SensorThings API standard, although too late for the V1.1 update, having one of the OGC SWG chairs within this project can only be advantageous.
- **Data Model.** Data model simplification has been a recurring request within INSPIRE. As these simplification options have gone hand in hand with experimentation with (Geo)JSON-based approaches, it logically follows to integrate such thoughts within the API evaluation process. INSPIRE Themes suited to simplification to SF-0 and/or SF-1 will be selected, whereby initial work has commenced on the INSPIRE Theme Transport Networks - Air. Current candidates include Transport Networks - Roads and Natural Risk Zones - Flooding. Pertaining to SensorThings API, we will reflect on the additional attributes provided within the extended properties fields to determine if these are essential to our use cases and propose simplifications. Of relevance here is also enabling stakeholders more engagement pertaining to the data model. Some

⁸ <https://github.com/DataCoveEU/API4INSPIRE>

options include well-governed feedback processes or the creation of simplified models with standardized extension points.

- **Support Ecosystem.** Just as important for rapid uptake and use as the API Specification and the Data Model is the existence and completeness of the support ecosystem. A central access point to all required specifications, a vibrant community providing timely feedback, a wide spectrum of examples and code snippets can go a long way towards boosting productivity and user satisfaction. Pertaining to the two software systems integral to this exercise, GeoServer⁹ 10 and FROST¹¹, there is already a vibrant community established providing rapid developer support as well as fixes and extensions to the software; the same is true for the underlying standards OGC API – Features¹² 13 and SensorThings API¹⁴, with open GitHub repositories available for providing insights pertaining to necessary standard updates. This support ecosystem also includes transformation tools like HALE allowing data providers a simple way of transforming and providing their data or conversion tools such as LD-Proxy that can copy data from a service implementing traditional OWS to JSON based services.

2.1.4 Evaluation Components

The following components compose our evaluation methodology:

- **Stakeholder Perspectives:** based on an analysis of stakeholders and their requirements towards provision and use of data, these have been defined in order to abstract the various roles taken by the stakeholders of this process. For more details on stakeholders, please see section “2.2 Stakeholder Analysis”.
- **Evaluation Criteria:** these formalize requirements towards APIs, providing a framework suited for gauging levels of maturity of an API specification and deployments as well as providing estimates of resource requirements for their achievement. For more details on the Evaluation Criteria, please see section “2.3 Evaluation Criteria - Five Level Open Data API evaluation model”
- **Evaluation Methods:** based on a review of state-of-the-art methods for the evaluation of APIs and other software systems, a set of different methods to be utilized for interrogating representatives of the Stakeholder Perspectives on the Evaluation Criteria was selected. For more details on these methods, please see section “2.4 Evaluation Methods”.

These evaluation components then feed into the Evaluation Process (Figure 2) as described in the section below, whereby the Evaluation Methods applied to the Evaluation Criteria will be executed with representatives of the Stakeholder Perspectives both during the implementation and deployment of the services further described in “D2 Deployment strategies for standard-based APIs” as well as in the course of events more strongly addressing usage aspects.

⁹ <https://github.com/geoserver/geoserver>

¹⁰ geoserver-users@lists.sourceforge.net

¹¹ <https://github.com/FraunhoferIOSB/FROST-Server>

¹² <https://github.com/opengeospatial/ogcapi-features>

¹³ https://github.com/opengeospatial/oapi_common

¹⁴ <https://github.com/opengeospatial/sensorthings>

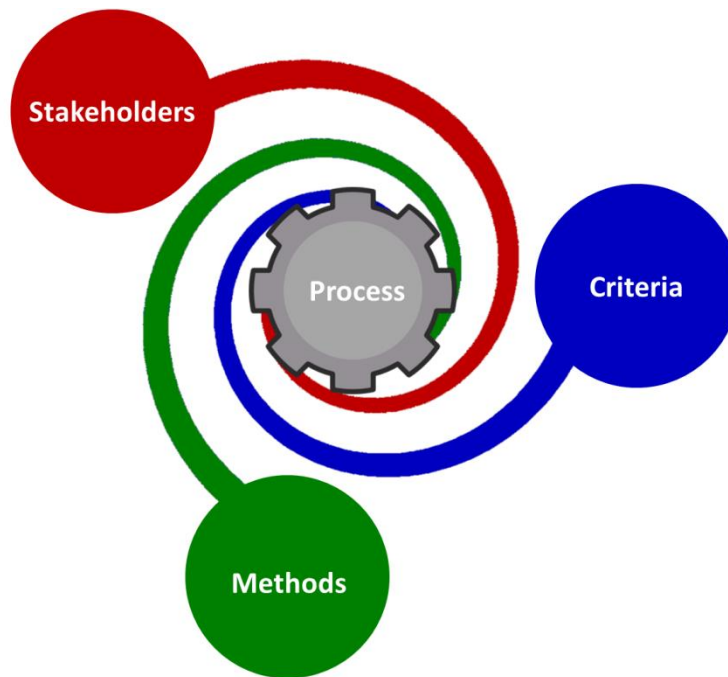


Figure 2. API4INSPIRE evaluation components

2.1.5 Evaluation Process

Based on our project goals, the evaluation dimensions identified as well as the evaluation components defined, our evaluation process will commence. This process will be tightly linked with the development and deployment processes further described under 3 Deployment strategies for standard-based APIs in order to gain insights pertaining to the development and deployment processes. Once the APIs have been deployed under the tasks comprising “D3 Technical Documentation, Code”, the evaluation steps pertaining to the user perspective will proceed, including the various events described for this purpose. This will be further documented in “D4 Assessment of the impact of APIs”

Once the evaluation process has been completed, all outputs will be merged and analysed, all gaps identified will be documented. All evaluation inputs will be reflected against the degrees of freedom identified in order to guide recommendations, quantify effort and define the contents of the supporting materials (technical guidance, software tools, tutorials, etc.) to be provided.

Based on the evaluation analysis, the outputs will be summarized and made available in various forms in order to meet the information requirements of the stakeholder groups involved.

- For administrative stakeholders, information on costs and benefits to be expected will be provided, allowing them to evaluate the potential of adoption of these technologies within their own organizations.
- For data providers, the necessary materials required to empower Member State authorities to deploy their data via the new APIs, as well as to understand the benefits and pitfalls of data simplification options will be provided.
- For data users, different levels of information will be required in order to address the different types of stakeholders.

A Webinar will be held in order to widely disseminate the outcomes of this project. All information resources and code collected during the project will be structured, documented, and made openly available. All relevant materials are being collected on the project GitHub.

2.2 Stakeholder Analysis

In this section, we provide an overview of the various stakeholder groups involved in providing and using data within the INSPIRE and beyond context. The requirements on the different stakeholder groups pertaining to both provision and use of data are analysed, ensuing implications presented in section 2.2.1 Stakeholder Overview.

Based on this overview of stakeholder requirements, different aspects of data provision and use have been identified and described, providing deeper insights into the current status quo, and helping to highlight both challenges and opportunities put on stakeholders through the INSPIRE process. These are described in sections 2.2.2 Provision Spectrum and 2.2.3 Usage Spectrum.

For the purpose of this analysis, we have refined the various aspects pertaining to the different stakeholder groups as well as their roles into a set of stakeholder perspectives. Thus, we can differentiate between a data provider making data available in comparison to the same data provider accessing data, either from within their own organization or from external sources. These are described in section 2.2.4 Stakeholder Perspectives, and serve to guide the evaluation process.

While these Stakeholder Perspectives cover the operational aspects of data provision and usage, we must also consider the administrative aspects. Without organization buy-in, most data provision and usage endeavours are doomed to fail due to resource issues. Thus, we also address administrative stakeholders as those tasked with ensuring the necessary resources for the creation and maintenance of the entire data ecosystem.

2.2.1 Stakeholder Overview

In an initial step, in order to gain a better overview of the actual requirements, we reviewed the various stakeholders to be expected pertaining to provision and use of data within INSPIRE and beyond. A related aspect is understanding the approach to data access associated with the different stakeholder types.

It is important to be aware of the fact that the same stakeholder can assume multiple roles, causing the provision vs. usage model to blur. Simultaneously, exactly this blurring of roles is where we expect the greatest benefit, as those data providers also using the API data will provide the most in depth inputs as to the synergies to be leveraged. In Table 1: Stakeholder Overview we provide an overview of the different stakeholders involved together with information on their data provision and usage modalities. From this we derive implications based in the individual requirements to help guide the further evaluation process.

Table 1: Stakeholder Overview

| | Provision | Use | Implications |
|--|--|--|---|
| Governmental Organizations providing data under INSPIRE | Provision with all requirements mandatory. Sometimes use of standard software, sometimes in-house developments for provision. | In-house use usually bypasses web services, directly accesses data sources. Often recurring processes already covered by in-house solutions. | Expect simple solutions for the continuation of their daily work, either integrated process for regular tasks, or standard formats that can be easily integrated into desktop GIS |
| Other Governmental Organizations (no INSPIRE requirements) | No provision requirements, but may also provide data regardless. Sometimes use of standard software, sometimes in-house developments for provision, whereby provided data models may be non-standardized, driven by backend DB or other requirements. | Access to data from other organizations (both within and without MS). Sometimes recurring processes already covered by in-house solutions; other times one-off studies. | Wish for simple solutions to enable their daily work. To date mostly depended on standard formats that can be easily integrated into desktop GIS. |

| | Provision | Use | Implications |
|-----------------------------|---|--|---|
| Industry | <p>No provision requirements, but may also provide data regardless.</p> <p>Sometimes use of standard software, sometimes in-house developments for provision, whereby provided data models may be non-standardized, driven by backend DB or other requirements.</p> | <p>Access to data from other organizations.</p> <p>Sometimes recurring processes already covered by in-house solutions; other times one-off studies.</p> | |
| NGOs | <p>No provision requirements, but may also provide data regardless.</p> <p>Sometimes use of standard software, sometimes in-house developments for provision, whereby provided data models may be non-standardized, driven by backend DB or other requirements.</p> | <p>Access to data from other organizations.</p> <p>Sometimes recurring processes already covered by in-house solutions; other times one-off studies.</p> | <p>Wish for simple solutions to enable their daily work. In-house processes for recurring analyses, some dependence on standard formats that can be easily integrated into desktop GIS, but also the realization that finding new answers implies creating new data tools</p> |
| Scientific Organizations | <p>No provision requirements, but may also provide data regardless.</p> <p>Sometimes use of standard software, sometimes in-house developments for provision, whereby provided data models may be non-standardized, driven by backend DB or other requirements.</p> | <p>Access to data from other organizations</p> <p>Usually new studies, thus expecting effort from data analysis & alignment</p> | <p>Some dependence on standard formats that can be easily integrated into desktop GIS, but strong realization that cutting edge research requires cutting edge tools. Limited funding for such tools, but willing students, staff and research personnel.</p> |
| Citizen Science Initiatives | <p>No provision requirements, but may also provide data regardless.</p> <p>Sometimes use of standard software, sometimes in-house developments for provision, whereby provided data models may be non-standardized, driven by backend DB or other requirements.</p> | <p>Access to data from other organizations</p> <p>Usually new studies, thus expecting effort from data analysis & alignment</p> | <p>Some dependence on standard formats that can be easily integrated into desktop GIS, but realization that research requires tools. Limited funding for such tools, but willing to accept some effort.</p> |

| | Provision | Use | Implications |
|------------------------|---|--|--|
| Interested Public | No data provision | Access to data from various organizations in a simple format, easy to view & understand, simple download formats | Simple (HTML) viewers with accompanying documentation would enable access to these resources. Simple download formats would allow non-professional users to interact with the primary data |
| EC Institutions | No provision requirements, but data centres sometimes try to align to INSPIRE data models and services. | Reporting data flows are increasingly being aligned with INSPIRE data models and services. | Modifications and extensions of the core data and service models must be supported. |
| Standardization Bodies | No provision requirements, but a vested interest in providing the relevant standards required for enablement. | No usage requirements, but a vested interest in providing the relevant standards required for enablement. | Standardization bodies must be engaged in order to ensure dynamic evolution as required by the other stakeholder groups. |

2.2.2 Provision Spectrum

In order to provide a complete system of costs and benefits to the data ecosystems being analysed, we must first disaggregate the various aspects of data provision and usage. Pertaining to data provision, the following approaches will be investigated:

- **Configuration of an existing server for the provision of the desired API.** In the case of OGC API - Features, we have selected GeoServer¹⁵ as the software of choice as this is widely deployed amongst data providers. In addition, developments towards provision of OGC API – Features are well progressed, providing us a relatively mature tool for the provision of this API. In the case of SensorThings API, we will utilize the FROST¹⁶ implementation. In some of these deployments, we will provide the full scope of data as specified within the INSPIRE Data Specifications; in others, we will experiment with the simplification options proposed by the INSPIRE MIF activity 2017.2 on alternative encodings. This work will be done in close cooperation with our associated data providers, ensuring that all APIs deployed are in line with their requirements.
- **Development of a dedicated system for provision of OGC API – Features.** In some cases, data providers will prefer to create their own implementation of the OGC API – Features directly on their local data sources in lieu of deploying unfamiliar systems. In order to gain a better overview of the costs involved in such an undertaking, a green-fields development is being done on the Austro Control Air Transport Network data by a group of motivated students. This work will give us insights into the challenges faced by “outsiders” not intimately familiar with the OGC and INSPIRE domains. Once this development has been validated on the initial data source, it will be deployed on other data provider’s data sources.
- **API by proxy on existing WFS2.** In some cases, data is already provided using WFS2, but the server software used for this cannot be upgraded to also support OGC API - Features. In this case proxy software,

¹⁵ <http://geoserver.org/>

¹⁶ <https://www.iosb.fraunhofer.de/servlet/is/82077/>

such as LD-Proxy¹⁷, may be a solution. This proxy software fetches data from the WFS2 server, and exposes the data through the OGC API - Features.

2.2.3 Usage Spectrum

Based on the stakeholder overview provided above, the following types of data usage become clear:

- **Direct access to data source (DB).** While this mostly pertains to in-house databases, there are also situations where entire databases are transferred between organizations on external discs. These solutions almost always require dedicated tooling tailored to the underlying databases. This can be a good solution for recurring tasks or when the dataset is too large for on-line transfer. Disadvantage is that external access to the dataset is not possible, and if both sides make changes to the data the two versions of the data set will go out of sync.
- **File-based data sources.** To date, many recurring and ad-hoc analyses have been performed based on data shared through semi-standardized file formats. An advantage of these formats is their simple accessibility through standard desktop GIS applications. Disadvantages are that such data cannot be interactively queried but can only be accessed as a monolithic block. In addition, the structure of the domain data contained is often not standardized. Examples of file-based data sources are Shapefiles¹⁸, SpatialLite¹⁹ databases, GeoPackage, CSV and binary formats (NetCDF, image formats).
- **Access to SOAP-Like OGC Web Services (OWS) with simple features.** Various existing OWS services, be they non-harmonized INSPIRE download services or other services made available under the Open Government Data (OGD) umbrella, provide an interesting complement to the more structured harmonized INSPIRE Services. While these services rarely utilize complex data standards in the provision of available data, instead usually providing a fairly direct image of their backend data sources, they provide access to domains not covered by other directives in a simple and accessible manner. OWS providing simple features enables integration with standard desktop GIS applications. Disadvantage is that for each data source, one must first analyse the data structure and tailor tools to correctly interpret the data.
- **Access to SOAP-Like OGC Web Services (OWS) with complex features.** Various organizations provide access to data via OWS; in some cases the INSPIRE Data Models are utilized for data provision, while in other cases either community standards or proprietary models are used. Mature data models and service specifications allow for the transparent, accessible and reusable provision of rich data models. The complexity of the data models and service specifications can also be seen as a disadvantage as the full width and breadth of the available data is experienced as overload. Integration into classic desktop GIS becomes difficult, as recently illustrated by the CanIUse INSPIRE project²⁰; an overview of which technologies can handle which encoding types has been provided there²¹.
- **Access to REST-Based APIs.** REST-Based APIs are a relatively new development in web service spectrum, providing developers with easy access to web resources. Currently, we are undergoing a wild-west phase of API development and deployment, with various semi-standardized approaches emerging; a development strongly welcomed by the developer community. REST-Based, standardized APIs pertaining to both the spatial data as well as spatio-temporal observations have newly emerged, and are the object of evaluation in this project. Alternatively, data users can configure various on-the-fly transformation tools such as LD-Proxy or pygeoapi in order to enable API-Like access to existing WFS2 based resources.

In addition to providing insights into current data usage modalities, these usage types also serve to illustrate the status quo pertaining to data provision. By classifying the data providers being interrogated during the evaluation process by their current data provision modalities, we can sharpen our insights as to how to best support and guide data providers at various levels of maturity in the most efficient usage of available resources towards data provision.

¹⁷ <https://interactive-instruments.github.io/ldproxy/>

¹⁸ <https://en.wikipedia.org/wiki/Shapefile>

¹⁹ <https://en.wikipedia.org/wiki/SpatialLite>

²⁰ <https://github.com/INSPIRE-MIF/caniuse>

²¹ <https://inspire-mif.github.io/caniuse/generator/out.html>

2.2.4 Stakeholder Perspectives

Based on this overview of relevant stakeholders and their requirements and approaches to data provision and use, we have derived the following **stakeholder perspectives** to guide us during the evaluation process:

1. **API Development** (Dev) [Provision]: Concerning the role of a software engineer, responsible to develop an implementation of the API Standard.
2. **API Deployment** (Deploy) [Provision]: Concerning the role of a data provider, responsible to publish through the API by utilizing and configuring existing implementations.
3. **API Use** (Use) [Usage]: Concerning the role of API consumers. These may be developers in the same organization owning the data, responsible for creating an application making use of in-house data. These may also be people outside the organization who are interested in the data, provided by the API. Access to the API is done either programmatically or via tools.

In addition to these three stakeholder perspectives, requirements pertaining to administrative stakeholders as described above will be accounted for.

2.3 Evaluation Criteria - Five Level Open Data API evaluation model

The technical evaluation criteria are based on the Five Level Open Data API evaluation model by Jarkko Moilanen²², which is in turn derived from Sir Tim Berners-Lee's 5-star deployment scheme for Linked Open Data. As the original 5-Level model does not reflect the standardization aspects inherent to INSPIRE and the ELISE project, additional criteria have been integrated to reflect ensuing requirements; other criteria have been modified to better correspond with the standards evaluation context. The criteria have also been compared to those included within the ISO 25010 Standard on Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE); the Suitability criteria was adopted from this source. These additions and modifications are noted in the individual criterion descriptions below.

As implied by the 5-Level methodology name, the criteria are grouped into five levels, which reflect increasing levels of maturity of both API and supporting ecosystem. Levels one through three pertain more to the maturity of the infrastructure provided by the data providers, with level one corresponding to data that has been provided with no regard to existing standards or conventions, level two pertaining to a standardized architecture with basic functionality and level three aligned with data provider requirements under INSPIRE as we know it. Levels four and five pertain more to the encompassing ecosystem, stemming from various communities.

Table 2 provides an overview of the Evaluation Criteria by level. More detail on the individual criteria is given in the following subsections.

We take these criteria as a foundation to evaluate the APIs through the stakeholder perspectives described in more detail above. The responses collected for these criteria will provide the basis for us to assess the relevance of these aspects for stakeholders assuming roles encompassed by the perspectives described above. The questions formulated for each criterion will also serve to evaluate how the effort accrued in its fulfilment compared to previous technologies, allowing us to determine potential savings through the usage of APIs.

22

https://www.europeandataportal.eu/sites/default/files/2013_api_simple_five_level_open_data_api_evaluation_model.pdf

Table 2: Overview of Evaluation Criteria

| Level 1: All find | Level 2: All use | Level 3: All trust | Level 4: All involved | Level 5: All develop |
|--|---|--|--|--|
| <p>Single Entry Point Is all information available from a single source (the portal), either directly or through links?</p> | <p>JSON or XML Does it support the use of JSON and/or XML?</p> | <p>Query and Analytics API * Does the API include Querying and Analytics?</p> | <p>SDK Availability Are API SDK's available for one or more environments?</p> | <p>Code Visible Is code visible/can be cloned?</p> |
| <p>Documentation Is updated documentation available?</p> | <p>Data License Are data license details given through the API?</p> | <p>Error Handling Is Error Handling in place and documented?</p> | <p>Code Examples Are there examples of code in one or more commonly used programming languages?</p> | <p>Bug Tracker Can Bugs, issues and suggestions be reported in a public place and is dialogue public?</p> |
| <p>Example Requests Are there examples of API requests seen as part of the documentation?</p> | <p>Terms of Use Are Terms of Use clear and easily accessible?</p> | <p>Performance and Cache * Can the API offer sufficient performance and does it support caching?</p> | <p>Community Is there a growing community to consult if needed?</p> | <p>API License/reuse Is the API's license known, are parts of the API covered by patent claims, and does it allow further development and re-use?</p> |
| <p>Example Data Are there examples of the API request returned data?</p> | <p>Embedded Metadata Does returning data include metadata?</p> | <p>Background Support Is the development and maintenance of the API supported by a big, stable entity or company?</p> | <p>Playground Is there an API Playground for testing and getting familiar with the API?</p> | <p>Development Roadmap Is the API's development roadmap known and is it visible for all?</p> |
| <p>Discoverability * Is it possible to discover deployed instances of the API based on the resources provided?</p> | <p>Authentication Does the API support authentication/authorization?</p> | <p>Availability * Is it easy to integrate into existing workflows and toolsets?</p> | <p>Linked Documentation Is documentation linked to code examples and back again?</p> | <p>Linked Data Ready * Is the data provided structured in a Linked Data ready manner, i.e. JSON-LD?</p> |

| | | | | |
|--------------------------|---|---|---|--|
| <p>Level 1: All find</p> | <p>Level 2: All use</p> <p>API Standardization *</p> <p>Is the API itself standardised, and is this specification openly available</p> <p>Suitability *</p> <p>Is the API suitable for the intended use?</p> | <p>Level 3: All trust</p> <p>API Data Validation *</p> <p>Can the data returned by the API be validated?</p> | <p>Level 4: All involved</p> <p>API Evolution *</p> <p>Can a developer, data provider or user provide feedback on issues with the data model or API functionality and are custom extensions to the API foreseen?</p> | <p>Level 5: All develop</p> <p>Test Framework available *</p> <p>Can conformance to the API be formally tested?</p> |
|--------------------------|---|---|---|--|

Note: those criteria extended to or modified from the original criteria list have been marked with an asterisk (*)

The effort required to meet each criterion is estimated on a scale of 1 to 5, with one being the least effort, and 5 being the highest. For some criteria, the effort can be shared with a larger community, or in some cases, community effort is required to meet the criterion. If this is the case, it is also indicated; in most cases, the effort will pertain to developers and deployers. Complementary to the effort involved in provision, we also quantify the benefit to users when the criteria are met. The effort and benefit estimations provided in this document stem from an initial heuristic quantification performed by the project development staff. These values will be successively refined with feedback received during the evaluation process. Effort and benefit estimates per criterion will be further refined by stakeholder role corresponding to the stakeholder perspectives described above, enriched through aspects described for the provision and usage spectra. In addition, the effort required to fulfil a criterion will be different from the effort accrued if the criterion is not fulfilled; both of these viewpoints will be integrated.

In the following sections, we present an overview of the five levels and the evaluation criteria therein. For details on how these will be posed as questions to each of our stakeholder perspectives, please see Annex A - Extended evaluation criteria.

2.3.1 Level 1: All find

This level corresponds to very basic non-standardized data provision, data is available together with simple examples. This level focuses on in-house use of APIs. Requires minimal effort in provision but is difficult to impossible to use.

Relevant questions: Is the API mainly designed for in-house use? Did the API happen by accident? Did the API evolve without relevant oversight or adherence to standards? Was the API designed based on a large set of realistic use-cases?

2.3.1.1 Single Entry Point

Is all information available from a single source (the portal), either directly or through links?

Finding all data one needs is a key requirement for any task. Ideally, there is a single place (the portal) through which all required information can be reached. For development this pertains to the API description, for deployment the data model definition, and for API use the data exposed through the API and the metadata required for understanding the data.

Effort required: 1.

Benefit provided: 4.

2.3.1.2 Documentation

Is updated documentation available?

Documentation is the main source to get information about the API. It should answer all the questions arising for all interested in the API, be it the software developer implementing the server, the publisher mapping his data to the data model, or the user requesting data. The documentation should be clear and precise, so that no "trial and error" is needed.

Effort required: 2.

Benefit provided: 5.

2.3.1.3 Example Requests

Are there examples of API requests as part of the documentation?

The interaction point with an API are requests, created by a client-(application), sent to the API implementing server, processed and the result is sent back. The request contains all information, which describes the users information needs. Example requests help with understanding the documentation from all perspectives.

Effort required: 2; Community input possible.

Benefit provided: 5.

2.3.1.4 Example Data

Are there examples of the data returned by API requests?

After processing the received request, the server returns the requested data. It's advantageous if the example data matches the example requests, and is described in detail.

Effort required: 2; Community input possible.

Benefit provided: 5.

2.3.1.5 Discoverability

Is it possible to discover deployed instances of the API based on the resources provided?

What means are available for discovering deployed instances of the API? Are dedicated services such as the OGC Catalogue Service Web (CSW) required for discovery? Can deployed instances of the API be discovered via normal search engines following W3C Data on the Web Best Practices (DWBP)²³ and Spatial Data on the Web Best Practices (SDWBP)²⁴?

Note: this criterion was added to reflect the organizational and standardization background against which this evaluation is being performed.

Effort required: 2.

Benefit provided: 5.

2.3.2 Level 2: All use

This level corresponds to basic standardized data provision. Data is available in a standardized format, basic standards based access functionality is provided. This level focuses on providing data to external users. Medium effort in provision, use is possible, but not optimal.

Relevant questions: Has the API been reviewed by external users? Has the API been designed for providing data to external users?

2.3.2.1 JSON or XML

Does the API support the use of JSON and/or XML?

Data needs to be represented in a serialized form to be transmitted. For web-based APIs, JSON and XML are established as de-facto-standard, since they're human readable and many implementations exists for various programming languages, which allows easy integration and reuse. These aspects are relevant for server development and API use, but not for the deployment.

Effort required: 1.

Benefit provided: 4.

2.3.2.2 Data License

Are data license details given through the API?

The main value of an API is provided through the data that is published through the API. Therefore, it's important that the license of the data is also available through the API itself. The data license is usually a legal document, and in most cases this document will be linked to from within certain API responses. Ideally, the data license is based on a standard licensing scheme such as CC BY.

Effort required: 1.

Benefit provided: 3.

2.3.2.3 Terms of Use

Are Terms of Use clear and easily accessible?

In addition to data license, the terms of use should be available. A data provider should be able to specify how and with which constraints an API service can be used, and it should be clear to users of the API where to find

23 <https://www.w3.org/TR/dwbp/>
24 <https://www.w3.org/TR/sdw-bp/>

this information. The terms of use are usually described in a legal document, and in most cases this document will be linked to from within certain API responses.

Effort required: 1.

Benefit provided: 3.

2.3.2.4 Embedded Metadata

Does returning data include metadata?

Getting data is often not sufficient. Additional metadata is needed to use and interpret the data correctly. For example, what is represented by the data? What are the units? The API must allow the developer to enable the provider to provide the metadata required by the user to interpret the data.

Effort required: 3 – 4.

Benefit provided: 4.

2.3.2.5 Authentication

Does the API support authentication/authorization?

Not all data should be available publically. To limit access to authorized users, authentication/authorization (auth*) mechanisms exist. To ease integration, the auth* methods should be based on existing, well-known protocols (e.g. OAuth²⁵ or OpenID-Connect²⁶). After authenticating, the authorization mechanisms define exactly which data the user is allowed to see.

Effort required: 2 – 4.

Benefit provided: 3.

2.3.2.6 API Standardization

Is the API itself standardised, and is this specification openly available?

Note: this criterion was added to reflect the organizational and standardization background against which this evaluation is being performed.

For an API to become widespread, it is helpful if the API is formally adopted as a standard by a well-respected, international standards body. Furthermore, an API standard rarely stands alone. Most standards refer to other standards for specific aspect. For example, in most standards, whenever a date or time is used, the encoding is done according the ISO 8601 standard. Referencing existing standards usually reduces the effort required for implementing the standard and makes it easier for clients to use the standard, since they can use common libraries that implement these referenced standards. It also reduces the effort of mapping data models, since standardised building blocks are likely to already be in use.

Effort required: 5; Community input required.

Benefit provided: 5.

2.3.2.7 Suitability

Is the API suitable for the intended use?

Note: this criterion was added based on the criteria available from within the ISO 25101 Standard.

For an API to be deployed and used, it has to be suitable for the intended use case. From the deployment perspective, this means the API has to be implemented in server software that fits in the deployment landscape of the data provider and that can be connected to, or loaded with, the data that the data provider intends to publish. From the user perspective, this means that the user must be able to request, in a suitably efficient manner, the data that he needs.

Effort required: 3.

²⁵ <https://en.wikipedia.org/wiki/OAuth>

²⁶ https://en.wikipedia.org/wiki/OpenID_Connect

Benefit provided: 4.

2.3.3 Level 3: All trust

This level corresponds to mature standardized data provision. Encompassing data is available in complex standardized format, powerful and mature standards based access functionality is provided. Fairly large effort in provision, use is well supported.

Relevant questions: Has the API been designed for use in a diverse set of use cases, in a diverse set of environments? Can the API support the complexity of data models required for real-world use cases? Has the API been designed for use with large data sets?

2.3.3.1 Query and Analytics API

Does the API include Querying and Analytics?

Note: this criterion was modified from the original to include query functionality together with analytics

APIs are often used to get data from a service. Usually only a subset of the available information is of interest. Therefore, the API should contain querying and analytic capabilities. First, this includes a suitably powerful filter mechanism to limit the response to those parts of the data that the user is interested in. Second, this includes a mechanism in the API to do basic analytical calculations, e.g. some aggregation functions.

Effort required: 4.

Benefit provided: 5.

2.3.3.2 Error Handling

Is Error Handling in place and documented?

While using an API errors might occur. Either there's an issue with the server itself, or the data sent by the user isn't correct or doesn't match the available data. To allow a user to handle these errors, the API should specify how errors are reported back to the user. At the same time, the error message should not expose sensitive internal information about the server deployment.

Effort required: 2.

Benefit provided: 4.

2.3.3.3 Performance and Cache

Can the API offer sufficient performance and does it support caching?

Note: this criterion was modified from the original to include performance functionality together with caching

APIs may have inherent performance bottlenecks that become apparent when deploying and using the API with large data sets or a large number of users. To increase performance and efficiency, caching mechanisms may be used.

Effort required: 2 – 4.

Benefit provided: 3.

2.3.3.4 Background Support

Is the development and maintenance of the API supported by a big, stable entity or company?

Deciding to use a specific API requires investments on all sides (dev, deploy, use). Thus, for the future development of the API itself and the implementation is important, to be sure that the API will still be relevant in the future. A big entity or company in the background increases this probability.

Effort required: 1 – 5; Community input possible.

Benefit provided: 4.

2.3.3.5 Availability

Is it easy to integrate into existing workflows and toolsets?

Many users and domain experts have well established workflows that use existing tools and (desktop) software packages. Switching to a different data source or API may break these workflows, which would greatly reduce the incentive for users to switch to the new API. These tools can include those used by providers to import data from external or primary sources into the local data infrastructure, or those used by data users to visualise or otherwise use the data.

Note: this criterion was modified from the original to include availability of relevant tooling.

Effort required: 4; Community input required.

Benefit provided: 4.

2.3.3.6 API Data Validation

Can the data returned by the API be validated?

For interoperability it is important that the data returned by an API conforms to the data model defined by the API. It is beneficial if there is a way to automatically check this, for example by checking the JSON against a JSON Schema, or XML against an XML Schema Definition.

Note: this criterion was added to reflect the organizational and standardization background against which this evaluation is being performed.

Effort required: 3; Community input possible.

Benefit provided: 4

2.3.4 Level 4: All involved

This level pertains more to the maturity of the wider ecosystem built up around the standardized specifications. The aspects described go beyond the control of the individual data providers and thus must be supported through diverse stakeholder communities. Achieving this level facilitates both the provision and usage perspectives.

Relevant questions: Can support for the API be seen as a community effort?

2.3.4.1 SDK Availability

Are API SDK's available for one or more environments?

Software Development Kits (SDKs) simplify the interaction with the API on the development and client side. They contain libraries that reduce the amount of code that needs to be created to interact with the API and help reuse existing work and integrating the API in new contexts.

Effort required: 3; Community input possible.

Benefit provided: 2.

2.3.4.2 Code Examples

Are there examples of code in one or more commonly used programming languages?

Like SDKs, code examples simplify and clarify the interaction with an API. They show in detail how certain aspects of the API should be used and can often directly be executed to see the covered features live in action.

Effort required: 3; Community input possible.

Benefit provided: 4.

2.3.4.3 Community

Is there a growing community to consult if needed?

A big, active and friendly community can be, in addition to the documentation, an additional source of information. Questions, best-practices and issues can be discussed within a community to spread available knowledge and provide support.

Effort required: –; Community input required.

Benefit provided: 4.

2.3.4.4 Playground

Is there an API Playground for testing and getting familiar with the API?

Learning by doing and practically testing the usage of the API helps getting more insights. A playground helps with this. Such a playground can range from a public server that anyone can access, to a Docker image that can be quickly deployed with standard settings, to a one-click-install installation package that can run on a desktop PC.

Effort required: 2; Community input possible.

Benefit provided: 3.

2.3.4.5 Linked Documentation

Is documentation linked to code examples and back again?

A documentation that links to code examples, example data and request (and back) helps the reader to better understand the API.

Effort required: 2; Community input possible.

Benefit provided: 3.

2.3.4.6 API Evolution

Can a developer, data provider or user provide feedback on issues with the data model or API functionality and are custom extensions to the API foreseen?

Note: this criterion was added to reflect the organizational and standardization background against which this evaluation is being performed.

APIs are rarely perfect and finished in their first incarnation. They also always have to strike a balance between the diverse requirements of many different use-cases on the one hand, and complexity on the other. Because of this, it is important that developers, providers and users have a way to provide feedback on issues, deficiencies and lack of clarity in the API. It is important that these issues are addressed in future versions of the API. Similarly, it is very helpful if the API has clear extension points so that the API can be extended for certain use cases that require functionality that is not in the API.

Effort required: 2; Community input required.

Benefit provided: 2.

2.3.5 Level 5: All develop

This level describes a well developed and mature ecosystem built up around the standardized specifications. The aspects described go far beyond the control of the individual data providers and thus must be supported through diverse stakeholder communities. Achieving this level facilitates both the provision and usage perspectives.

Relevant questions: Can (further) development of the API be seen as a community effort?

2.3.5.1 Code Visible

Is code visible/can be cloned?

Having access to the source code of a reference implementation allows a deeper understanding of the implementation. It offers the possibility to check if specific behaviour was intended or is a bug. Available open-source code with a suitable license offers the possibility to add own changes, so that there is no dependency on a third-party.

Effort required: 5; Community input required.

Benefit provided: 3.

2.3.5.2 Bug Tracker

Can Bugs, issues and suggestions be reported in a public place and is this dialogue public?

Though an API is not software and can thus not have “bugs” in the traditional sense, an API can still have inconsistencies, errors and unclear definitions. Often these issues are not noticed until the API is applied in specific use cases, or implemented by multiple people. Having a Bug Tracker publically available offers a place, where to report issues and to track discussions and solutions. Having such a system openly available allows input from a wider community, helping the system to evolve to support a wider user community. It also makes it easier to collaborate on extensions.

Effort required: 1; Community input possible.

Benefit provided: 4.

2.3.5.3 API License/reuse

Is the API's license known, are parts of the API covered by patent claims, and does it allow further development and re-use?

Parts of APIs can be covered by patent claims, making it impossible to implement the API without paying royalties. The license of the API is important and might be a blocker, if the API needs to be re-used or if further development of the API is required. Ideally, if a license is required this should be based on a standard licensing scheme such as CC BY.

Effort required: 2; Community input possible.

Benefit provided: 3.

2.3.5.4 Development Roadmap

Is the API's development roadmap known and is it visible for all?

A roadmap can help to understand the further development direction of the API and to know what to expect in the (near) future.

Effort required: 3; Community input possible.

Benefit provided: 2.

2.3.5.5 Linked Data Ready

Is the data provided structured in a Linked Data ready manner, i.e. JSON-LD?

Note: this criterion was added to reflect emerging technological advances to be expected within the stakeholder community.

Linked Data is a technology that looks very promising and that has been on the horizon for some time now, with parts and concepts of it finding their way into APIs and data models. While it is not yet practical to have an API that fully employs all Linked Data principles, it is possible to design the API and data models in a way to allows Linked Data adoption in the future.

Effort required: 3; Community input required.

Benefit provided: 3.

2.3.5.6 Test Framework available

Can conformance to the API be formally tested?

Note: this criterion was added to reflect the organizational and standardization background against which this evaluation is being performed.

Test Frameworks can be used to verify the implementation and deployment of the API. This helps interoperability by insuring the deployed services implement the API correctly.

Effort required: 4; Community input required.

Benefit provided: 3.

2.3.6 SQuaRE: ISO 25010 Model

Our main evaluation criteria is the “Five Level Open Data API evaluation model” described above. To validate this model, additional evaluation catalogues were investigated. In the context of software product quality the “ISO 25010 Standard on Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE)” is well established. It defines properties for a software product (without naming a concrete implementation) which should be considered to have a high-quality product. Although those properties were developed for evaluating a software product, they can be used for evaluating an API, too. To prevent having an additional evaluation model in this project, the ISO standard is used to validate our already defined criteria and to check the completeness of the “Five Level model”.

As a result our evaluation criteria were mapped to the categories of the ISO standard (see Table 6 in Annex B - SQuaRE: ISO 25010 Model). In addition we double-checked that all categories of the ISO standard are mapped to at least one criteria of the Five Level model. The categories *Context coverage* and *Compatibility (technical)* of the ISO standard were considered irrelevant. *Suitability* was added to Level 2, as an additional criterion.

2.4 Evaluation Methods

For the evaluation of the degree of achievement of the evaluation criteria defined pertaining to each of the stakeholder perspectives, a set of evaluation types have been defined. These range from Heuristic Expert Evaluation over Peer Review of critical functionality to Interviews and Questionnaires performed in the framework of various events. Details on the different evaluation types as well as on events being organized in order to allow us to engage with the various stakeholder groups is provided in the following subsection 2.4.1 Evaluation Types.

The different evaluation types have been matched with both the stakeholder perspectives as well as the evaluation criteria, with the most suitable evaluation types assigned to each criterion paired with a stakeholder perspective. Detailed information on how the evaluation types will be applied towards the different stakeholder perspectives can be found in subsections 2.4.2 API Development and Deployment 2.4.3 API Use. Explicit alignment of evaluation types with the evaluation criteria and stakeholder perspectives can be found in Table 3: Evaluation types for the stakeholder perspectives develop and deploy and Table 4: Evaluation types for the stakeholder perspective use. Based on these alignment tables, the explicit content of the interview and questionnaire questions will be derived, whereby the questions defined for each evaluation criterion are described in Annex A - Extended evaluation criteria.

A great deal of insight pertaining to the development and deployment processes will be gleaned from our data providers. The data sources and provision technologies have been selected to not only provide insights into direct development and deployment requirements, but also to allow for comparison between competing technological solutions. For selected data providers, the same data source will be exposed through alternative technologies, allowing for direct comparisons pertaining to both provision and usage efforts. Based on these interviews, we will gain insights as to the effort required in reaching the successively more complex levels.

Events are being organized in order to allow us to directly engage with the wider user community. The various data sources provided by the development and deployment stakeholders linked to this project will be made available; information on the provision process will be provided together with various usage examples. The participants will then be given the opportunity to interact and experiment directly with the APIs and other services provided to gain a better understanding of the potential of these technologies as related to their daily work. During these events questionnaires will be provided to collect feedback from participants, individuals will be selected for more in depth interviews. More information on the events being organized is available from the following subsection Events within subsection 2.4.1.3 Stakeholder Evaluation.

2.4.1 Evaluation Types

2.4.1.1 Heuristic Expert Evaluation

The Heuristic Expert Evaluation is the least structured of the planned evaluation types. Under this methodology, we rely on the experience of the domain experts involved within the API4INSPIRE Project, both the project team as well as data provider staff will be included. While some of this input will be collected during dedicated meetings, we expect many of these insights to be gained on-the-fly while performing the tasks required to develop, deploy and use the two APIs being evaluated. Such interactively identified insights will be collected as issues within the project GitHub at <https://github.com/DataCoveEU/API4INSPIRE>. These will be merged with the insights gained from the more formalized evaluation meetings, and will provide a well-founded basis for our subsequent analysis.

The Heuristic Expert Evaluation will be applied to all interaction types described above. For some criteria, this evaluation type will suffice to provide well-founded input. For other criteria, this evaluation step will help to highlight issues that must be followed up on by the other methods listed.

2.4.1.2 Peer Review

The API Usability Peer Review process proposed by Farooq will be a valuable tool for gaining focused insights on specific aspects of the API usability. While this methodology has been designed for the evaluation of the features of the emerging APIs, we believe that it can be effectively applied within the context of evaluating existing API Specifications. The original process foresaw the following roles:

1. Feature Owner: responsible for the specification and development of a specific functionality
2. Feature Area Manager: responsible for the wider functionality area, can put the individual functionality into context
3. Usability Engineer: responsible for evaluating usability, coordinates process
4. Reviewers: organizational peers who will provide feedback.

In the original process, once the Usability Engineer has organized a review, the Feature Owner presents the new functionality in order to allow rapid uptake by the Reviewers. The Reviewers provide feedback, the Usability Engineer collects and structures this feedback, the Feature Area Manager assures that all feedback and ensuing activities remain aligned with the requirements of the wider feature area.

As the API Usability Peer Review process was custom tailored to the evaluation of an API under development, we must first adapt this process to our current evaluation target pertaining to the deployment and usage of a standardized API. As we are not evaluating the explicit implementations but instead only evaluating the functionality according to the standard, we believe that we can merge the roles of Feature Owner and Feature Area Manager without negatively impacting the robustness of the methodology, whereby we will refer to this role as the Feature Manager. Thus, within the API Peer Review, the following roles will be required:

1. Feature Manager: responsible for the description of a specific functionality, as well as ensuring alignment with the wider context.
2. Usability Engineer: responsible for evaluating usability, coordinates process
3. Reviewers: organizational peers who will provide feedback.

Before the review is started, the Feature Manager and Usability Engineer will prepare basic documentation on the feature to be evaluated, if relevant examples. In addition, questions to be posed will be documented.

Once the review commences, the Feature Manager will have 20 minutes to present the functionality, explain how it is to be utilized, known strengths and weaknesses. Once the functionality has been presented, the prepared questions are posed to the reviewers. The Usability Engineer will collect all insights from the ensuing dialog between the Feature Manager and Reviewers. Towards the end of this block, there should also be a question foreseen allowing reviewers to provide input on strengths and weaknesses identified but not covered by the other questions (AOB, but for questions, AOQ). This question and answer session should last about 60 minutes. At the end, the Usability Engineer has 10 minutes to present the insights taken and finalized these with the reviewers.

After the review has been performed, the Feature Manager and Usability Engineer come together for a post mortem meeting, discuss the outcomes of the review in detail and formalize the output.

2.4.1.3 Stakeholder Evaluation

While we have a well-balanced team of developers and data providers both within our internal project team as well as our associated data providers, we must go beyond this select group in order to gain deeper insights into the requirements of the wider stakeholder community.

In contrast to our internal and associated teams the stakeholder community will mostly not be familiar with the APIs being presented nor the data model variants utilized. Before we proceed to interrogate this community via questionnaires or interviews, we must first provide them with all relevant information required to gain the necessary insights into the proposed technologies and data models. Thus this evaluation type will be undertaken within the framework of various events, during which the necessary information will be disseminated. The questionnaires will be made available to all participants, while we will select individuals for more in depth questioning in the form of an interview.

2.4.1.4 Events

Foremost pertaining to the usage aspects of the APIs under evaluation, various events will be organized to allow us to directly engage with potential stakeholders, disseminate information on the current status of these APIs and collect feedback based on their experiences. The participants in these events will form the pool of experts who will be further interrogated via questionnaires and interviews as described in the sections below.

One challenge faced in the organization of events is posed by the tight project duration. Many relevant events initially planned such as the regular hackathons organized by the German cities of Munich or Hamburg do not temporally convene with our requirements. Participation in events cannot be guaranteed as we must await the outcome of the submission process where we have applied.

Events to be conducted in the scope of this study are planned to include, whereby the later points are fall-backs for the case that the first two points prove not viable:

- INSPIRE/api APithon at the INSPIRE Conference in Dubrovnik (12-15 May: WS submitted, further organizational aspects discussed with JRC) In addition to API usability aspects, in Dubrovnik we will provide information on our development and deployment activities, as well as allowing participants to provide their own existing WFS2 solutions via LD-Proxy
- FOSS4G Europe (13-17. July) API Hackathon will be submitted as a WS
- An alternative for gaining feedback on API usage would be a dedicated in-house event at Fraunhofer IOSB
- A similar alternative would be the organization of a Webinar. The organizers present the available APIs together with existing code examples, the participants are given the opportunity to experiment and ask questions
- While formal EGU (3-8 May) participation is no longer possible, we are considering leveraging the high concentration of relevant experts in Vienna the week before Dubrovnik through the organization of a side event or round table.

2.4.1.5 Questionnaire

Questionnaires will be created in order to gain insight into various evaluation criteria, whereby the questions to be posed will be constrained to those where we expect answers that fill well to the questionnaire format. For a full list of questions pertaining to the evaluation criteria, please see the Annex A - Extended evaluation criteria. Most questionnaire questions will be limited to the following response types to allow for wide scale analysis of the outcomes:

- Binary yes/no responses
- Multiple choice responses (both single response and multiple response versions)
- Word Clouds

We propose the usage of an online survey tool for the questionnaires, whereby the project team can provide such functionality via Mentimeter²⁷. At present we foresee one questionnaire that also covers provision aspects; should such an approach lead to unnecessary questions (i.e. it is not possible to branch the questionnaire to only pose usage questions if the respondent does not provide data) separate questionnaires will be created for deployment and use.

2.4.1.6 Interview

While many of the evaluation criteria can be reduced to simple questions suited for a questionnaire as described above, other criteria will require more complex feedback. For this purpose, participants will be selected from our events based on their level of engagement and the more complex evaluation criteria discussed with them in the form of a structured interview, with the interview questions being derived from the questions pertaining to development, deployment and usage provided together with the full list of criteria in Annex A - Extended evaluation criteria.

²⁷ <https://www.mentimeter.com/>

2.4.2 API Development and Deployment

As we only have three API development teams, the evaluation types will be limited to expert heuristic evaluation and interview as described above. Creation of a questionnaire for such a small stakeholder group would not be efficient, but could be done if a similar evaluation is performed in a wider context. Should the need arise we may introduce the Peer Review evaluation type, but at present we believe that the evaluation types posed suffice. The evaluation aims to provide answers to the detailed criteria questions pertaining to development provided in the Annex to this document.

Pertaining to API deployment, the evaluation types expert heuristic evaluation and interview will be used similarly to the API development evaluation. In addition, questionnaires will be utilized in order to address a wider audience of interested data providers that are taking notice of our activities, be it through our events or through word-of-mouth. The evaluation aims to provide answers to the detailed criteria questions pertaining to deployment provided in the Annex to this document. The interview questions will take administrative and infrastructural aspects into account, illustrating how existing investments in infrastructure and applications can be leveraged. Table 3 shows which evaluation types are to be applied to which evaluation criteria:

Table 3: Evaluation types for the stakeholder perspectives develop and deploy

| Stakeholder Perspective: | API Development | | API Deployment | | |
|---------------------------|-----------------------------|-----------|-----------------------------|---------------|-----------|
| | Heuristic Expert Evaluation | Interview | Heuristic Expert Evaluation | Questionnaire | Interview |
| Criteria: | | | | | |
| Level 1: All find | | | | | |
| Single Entry Point | X | | X | | |
| Documentation | X | X | X | X | X |
| Example Requests | X | X | X | X | X |
| Example Data | X | X | X | X | X |
| Discoverability | X | X | X | X | |
| Level 2: All use | | | | | |
| JSON or XML | X | | NA | NA | NA |
| Data License | X | | X | X | |
| Terms of Use | X | | X | X | |
| Embedded Metadata | X | | X | | X |
| Authentication | X | X | X | X | X |
| * API Standardization | X | | X | | X |
| * Suitability | | | X | X | X |
| Level 3: All trust | | | | | |
| * Query and Analytics API | X | X | X | | |

| Stakeholder Perspective: Criteria: | API Development | | API Deployment | | |
|---|-----------------------------|-----------|-----------------------------|---------------|-----------|
| | Heuristic Expert Evaluation | Interview | Heuristic Expert Evaluation | Questionnaire | Interview |
| Error Handling | X | X | X | | |
| * Performance and Cache | X | X | X | | X |
| Background Support | X | | X | X | |
| * Availability | NA | NA | X | X | X |
| * API Data Validation | X | X | X | X | |
| Level 4: All involved | | | | | |
| SDK Availability | X | X | NA | NA | NA |
| Code Examples | X | X | NA | NA | NA |
| Community | X | | X | X | |
| Playground | X | | X | X | |
| Linked Documentation | X | | X | X | |
| * API Evolution | X | | X | | X |
| Level 5: All develop | | | | | |
| Code Visible | X | | X | X | X |
| Bug Tracker | X | | X | X | |
| API License/reuse | X | | X | X | |
| Development Roadmap | X | | X | X | X |
| * Linked Data Ready | X | | X | | |
| * Test Framework available | X | X | X | | |

2.4.3 API Use

As API usability is the strongest argument for their uptake, we will investigate this aspect with a wide range of evaluation types. In order to gain deep insights into suspected weaknesses of the APIs, in addition to Heuristic Expert Evaluation we will apply the Peer Review process described above with participants from among our data providers, development teams and associated experts.

Various events as described above are planned that will allow us to present the APIs to a wider audience and collect feedback after they have had the opportunity to interact with the APIs and gain their own experience. Questionnaires will be used to address the wider audience, while interviews will be conducted with select event participants.

In Table 4 we show which evaluation types are to be applied to which evaluation criteria:

Table 4: Evaluation types for the stakeholder perspective use

| Stakeholder Perspective: Criteria: | API Use | | | |
|---|----------------------|-------------|---------------|-----------|
| | Heuristic Evaluation | Peer Review | Questionnaire | Interview |
| Level 1: All find | | | | |
| Single Entry Point | X | | | |
| Documentation | X | | X | X |
| Example Requests | X | | X | X |
| Example Data | X | | X | X |
| Discoverability | X | X | X | |
| Level 2: All use | | | | |
| JSON or XML | X | | | |
| Data License | X | X | | |
| Terms of Use | X | | | |
| Embedded Metadata | X | X | X | |
| Authentication | X | X | X | |
| * API Standardization | X | | X | |
| * Suitability | | X | X | X |
| Level 3: All trust | | | | |
| * Query and Analytics API | X | X | X | X |
| Error Handling | X | X | X | |
| * Performance and Cache | X | X | X | |
| Background Support | X | | X | |
| * Availability | X | | X | |
| * API Data Validation | X | | X | |
| Level 4: All involved | | | | |
| SDK Availability | X | | X | |
| Code Examples | X | | X | |
| Community | X | | X | X |
| Playground | X | | X | |

| Stakeholder Perspective: | API Use | | | |
|----------------------------|----------------------|-------------|---------------|-----------|
| Criteria: | Heuristic Evaluation | Peer Review | Questionnaire | Interview |
| Linked Documentation | X | | X | |
| * API Evolution | X | | X | |
| Level 5: All develop | | | | |
| Code Visible | X | | | X |
| Bug Tracker | X | | X | X |
| API License/reuse | X | | | X |
| Development Roadmap | X | | | X |
| * Linked Data Ready | NA | NA | NA | NA |
| * Test Framework available | X | | | |

2.5 Evaluation Process

In this section, we describe the individual steps of our evaluation process in detail. We begin with a description of how the evaluation methodology initially foreseen for this task has been further reflected against the project goals, the methodologies refined as required. We go on to sketch the consultation process, describing the different evaluation types being applied to the different stakeholder perspectives in order to gain insights pertaining to our evaluation goals. Once the consultation has been completed, the analysis of the outputs will commence, providing insights as to strengths and weaknesses of the new technologies under evaluation as well as potential measures. In a final section, we describe how the insights gained will be used to generate the necessary materials required to enable MS stakeholders to assess the implications of adopting these technologies as well as deploy and utilize them within their own organizations.

Figure 3 provides an overview of the Evaluation Process. The consultation step has been split by stakeholder perspective, as these different perspectives require quite different approaches to the evaluation process. All evaluation steps are described in greater detail in the sections below.

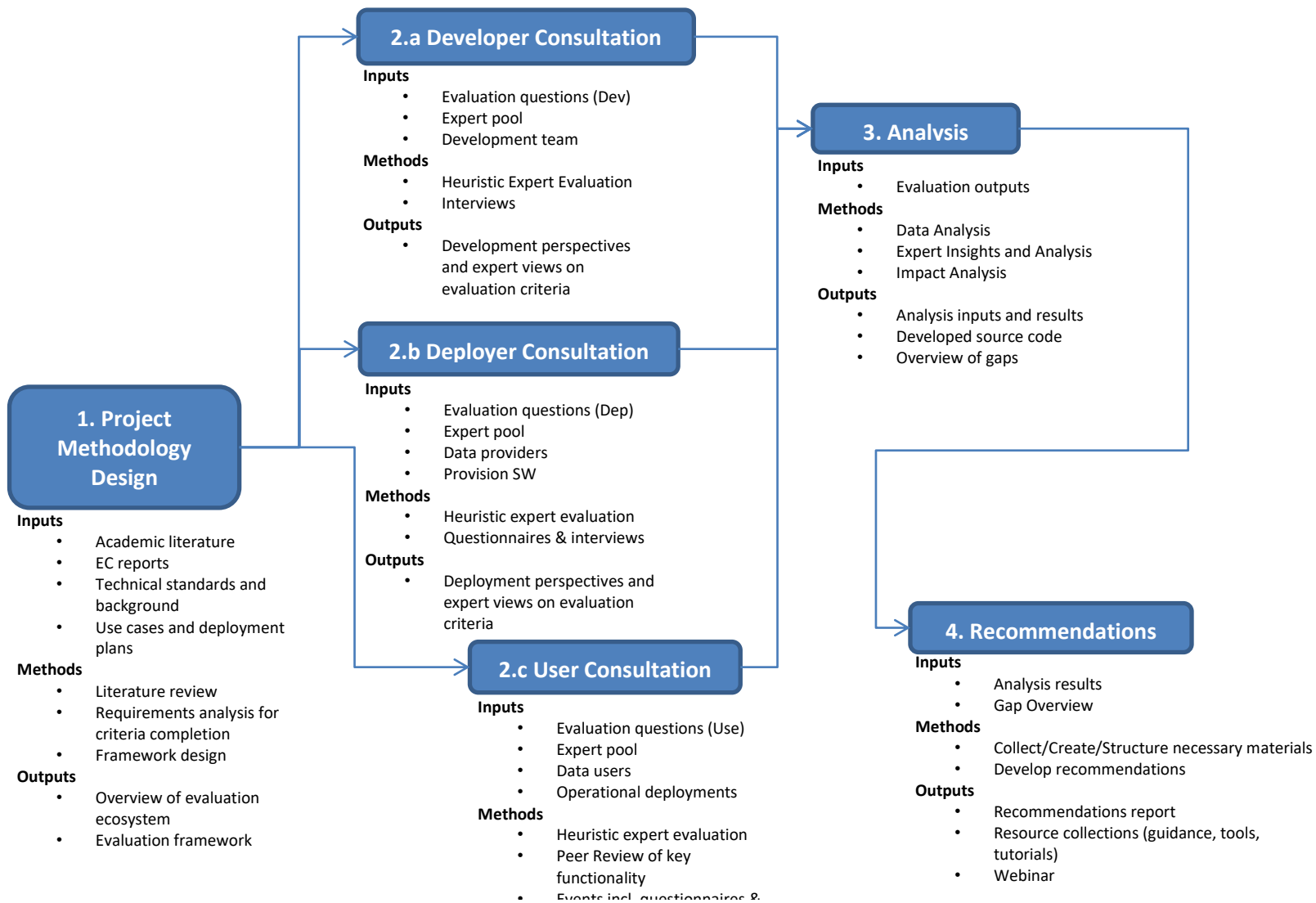


Figure 3: Overview of the evaluation process. The consultation step has been split by stakeholder perspective, as these different perspectives require quite different approaches to the evaluation process. All evaluation steps are described in greater detail in the sections below.

2.5.1 Project Methodology Design

As described in the sections above, the initial evaluation process foreseen was refined based on relevant literature and the experience of the project team. In order to gain deeper insights into the stakeholder community and their requirements, we first put together a stakeholder overview, whereby this information was based both on our long experience within the INSPIRE and wider governmental data sector complemented with insights from relevant reports. This list was extended through information on typical provision and usage patterns pertaining to the stakeholder group together with implications stemming from these requirements. Based on this list, we then derived the core stakeholder perspectives that serve to guide this evaluation.

Based on the stakeholder overview with provision and usage requirements, we then further refined both the provision and usage spectra, detailing the diverse provision and usage options available. In addition to deepening our understanding of stakeholder requirements, this analysis also provides a good overview of the existing landscape, allowing us to evaluate the impact of the new APIs and data models against known quantities and efforts.

In order to further guide our evaluation process, we also investigated and documented the degrees of freedom available for remediation of issues identified. This will help guide us in identifying potential solution space while giving us the wisdom to accept the things we cannot change. In line with our agile approach, we then revisited our goals and refined them in light of these insights.

The Five Level Open Data API evaluation model by Jarkko Moilanen was then refined based on the requirements identified. Further criteria were added, both from other evaluation schemes such as described in ISO 25010 as well as based on our understanding of the wider evaluation domain, whereby the main extension focus was to cover the standardization aspects missing in Moilanen's work. In some cases existing criteria were modified to better reflect our requirements. In addition, we defined more generalized criteria to indicate general achievement of a specific level.

In parallel to the work on the evaluation criteria, we also refined the evaluation types foreseen in the project concept. Some evaluation types were modified to better reflect the evaluation goals specific to this project, as most methodologies available are tailored for the evaluation of a concrete software implementation. Paired with this work was determining which evaluation type is most suited for gaining insights pertaining to a specific criterion applied to the perspective of a specific stakeholder perspective. This alignment is provided together with the description of the different evaluation types. In addition, events have been foreseen in order to attract a wider user community, and gain insights from their interactions with the APIs being made available by this project.

Once the evaluation criteria were aligned with the requirements of this project, we then derived questions pertaining to each level and individual criterion, tailored to reflect the insights of the stakeholder perspective being addressed. These questions will be further refined in accordance with the evaluation types deemed suitable for the evaluation of the specific criterion being addressed as different evaluation types allow for different types of interactions, and thus different complexities in both the questions posed and the types of answers expected. All these additions and modifications to the evaluation criteria have been documented and provided within this document; full information on potential questions has been shifted to Annex A - Extended evaluation criteria for easier reference.

2.5.2 Consultations

While the consultations pertaining to API use will be grouped around our proposed events, the consultations with the developers and the data provider staff involved in deploying the new APIs will take place in an ongoing manner during the deployment process. More details on the deployment process will be provided in D2 Deployment strategies for standard based APIs. In addition to the more fine-grained criteria stemming from the 5-level evaluation model, the consultations will also include effort required in achieving these individual levels.

All gaps pertaining to documentation and other essential resources identified during the development, deployment and use processes will be collected on the project GitHub as issues marked with the label "documentation missing", providing a dynamically up to date overview of all such gaps encountered. All required materials will be collected and collated on the GitHub Wiki, whereby the contents of this wiki can later be transferred to a target site of the customer's choice.

2.5.2.1 Developer Consultation

Pertaining to developer consultation, we have an interesting resource pool to interrogate. On the one side, we have the developers of two of the leading API provision systems GeoServer and FROST within our project team. These colleagues from Fraunhofer IOSB and GeoSolutions will provide well-founded information on their experiences in implementing professional solutions for the provision of these APIs.

In order to complement these insights with fresh perspectives, we have also triggered the development of a simplified green-fields implementation of the OGC API – Features tailored to the provision of simple features corresponding to the SF-0 feature standard. This implementation is being done by a team of two highly motivated students currently undergoing vocational training in Information Technology at the Höhere Technische Lehranstalt Spengergasse, a school for higher technical education located in Vienna. The initial implementation is being done on datasets provided by Austro Control, both as SQLite and PostGIS data sources.

As the number of involved developers is naturally limited, the evaluation types applied will be far more interactive, consisting of ongoing Heuristic Expert Evaluation paired with more structured interviews aligned with the evaluation criteria. In addition, the developers will be requested to provide insights into the efforts required for implementation, both in absolute terms as well as in relation to known efforts pertaining to existing technologies.

2.5.2.2 Deployer Consultation

The six institutions integrated into this project as data providers will provide us with insights pertaining to the process of deploying existing solutions for the provision of the APIs under evaluation. As all of these organizations are presently involved in the INSPIRE data provision process, they can provide well-founded input on the deployment effort of both the new APIs as well as the simplified data models in comparison to their already accrued efforts pertaining to existing technologies.

Similar to the developer consultation, as the number of involved parties is necessarily limited, the evaluation types applied will be far more interactive. Most responses to the evaluation criteria will be obtained via ongoing Heuristic Expert Evaluation paired with more structured Interviews aligned with the evaluation criteria. However, we also plan on engaging further potential data providers in the course of our events. Questionnaires are foreseen in order to gain access to this additional information, which will be integrated to the insights gleaned from our internal teams.

2.5.2.3 User Consultation

Gaining access to the wider user community is one of the challenges of this project. For this purpose, we are organizing various events where the participants will be able to engage with operational examples of the various constellations of APIs and data models being provided by our data providers. Simple tools and scripts for access and use of the available data will be presented, giving the participants the opportunity to interact and experiment with these APIs in a hands-on manner. In addition, project staff will be available for interested users to provide support in understanding and using these resources.

As we see the greatest potential for reduction of required effort pertaining to the usability of the emerging APIs and simplified data models, we will apply the Peer Review evaluation type to the methodologies utilized to evaluate selected functionality. In the user consultation we will focus more strongly on the questionnaires as a method of gaining insights from a larger stakeholder group while identifying the more engaged participants and pulling them aside for more in depth interviews. These insights will be rounded off by our team of experts providing insights based on their experience.

2.5.3 Analysis

Once the APIs have been deployed and evaluated, the analysis of the evaluation outputs will commence. Before we describe the analysis in detail, we would like to revisit the various dimensions that define the information space created by this evaluation:

— Stakeholder Perspectives: development, deployment and usage aspects are included

- Provision alternatives: green-fields development vs. configuration of existing software. Full details of this dimension are provided in section 2.2.2 Provision Spectrum.
- Usage options: accessing APIs via custom software or integrating into existing tools. Full details of this dimension are provided in section 2.2.3 Usage Spectrum.

- APIs: OGC API – Features and SensorThings API, comparison to previous generation OWS
- Data Models: Complex features as defined in the INSPIRE data specifications as well as simplification options
- Available documents and other community resources

Requirements and opportunities related to these dimensions have been collected in accordance with the evaluation criteria, providing not only basic data such as the extent to which the criterion is fulfilled by a specific API or data model but also information providing deeper insights into the ramifications of this criterion and highlight suitable measures for potential future support activities and resources.

Efforts in provision and use have been evaluated in relation to

- the attainment of the evaluation levels
- the APIs in comparison to OWS
- simple vs complex data models

These inputs must be reflected against the degrees of freedom identified in order to guide recommendations, quantify effort and define the contents of the supporting materials (technical guidance, software tools, tutorials, etc.) to be provided.

The core feedback unit will pertain to a specific evaluation criterion as well as a specific stakeholder perspective; e.g. the criterion Embedded Metadata pertaining to the Deployment Perspective. Depending on the evaluation types utilized, different types of feedback will be merged to provide consistent information including fulfilment of criterion, effort information (effort required for provision or effort incurred due to its lack during use) and recommendations for supporting material on this feedback unit.

Once all information has been collated per feedback unit, we will merge this information along the analysis axes in order to generate overview information pertaining to the different aspects being analysed. In the examples given in Figure 4: Analysis overview of level of achievement, we compare OGC API – Features with SensorThings API for the Stakeholder Perspective User as well as provision and use aspects pertaining to OGC API – Features. The colours, as described in the key, denote the following values:

- Yes: clear agreement that the criterion is fully met.
- Partly: either only certain aspects of the criterion have been fulfilled.
- No: clear agreement that the criterion has not been met.
- Unsure: the stakeholders interrogated were not able to provide a satisfactory answer to the criterion, or there is strong divergence in the feedback to this criterion.
- No Data: the criterion was deemed not applicable to the evaluation dimension.

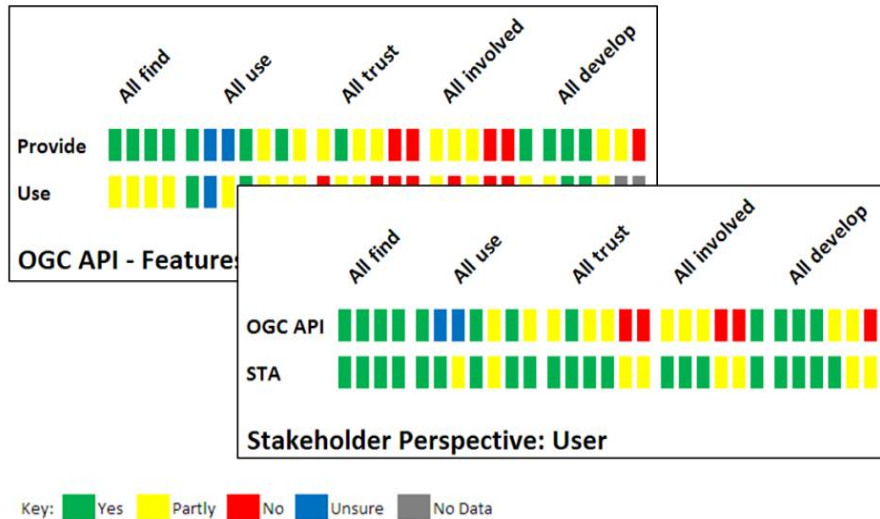


Figure 4: Analysis overview of level of achievement

Effort information collected will first be merged per stakeholder perspective and then evaluated against alternatives along the dimensions described pertaining to effort evaluation above. In addition, effort will be compared between provision and usage stakeholder perspectives, whereby we expect to find a Sweet-Spot somewhere in the middle of the effort spectrum, not too much effort required for provision while supporting the most important usability concerns (See Figure 5: balance between deployment and use effort, please note that the current diagram is a heuristic estimation serving only illustrative purposes. Updated figures will be provided in "D5 Conclusions and recommendations for MS authorities"). As mentioned above, effort will only be qualitatively estimated on a scale of 1 to 5, with one being the least effort, and 5 being the highest. Effort information will also be differentiated into effort relating to initial uptake and effort relating to day to day use.

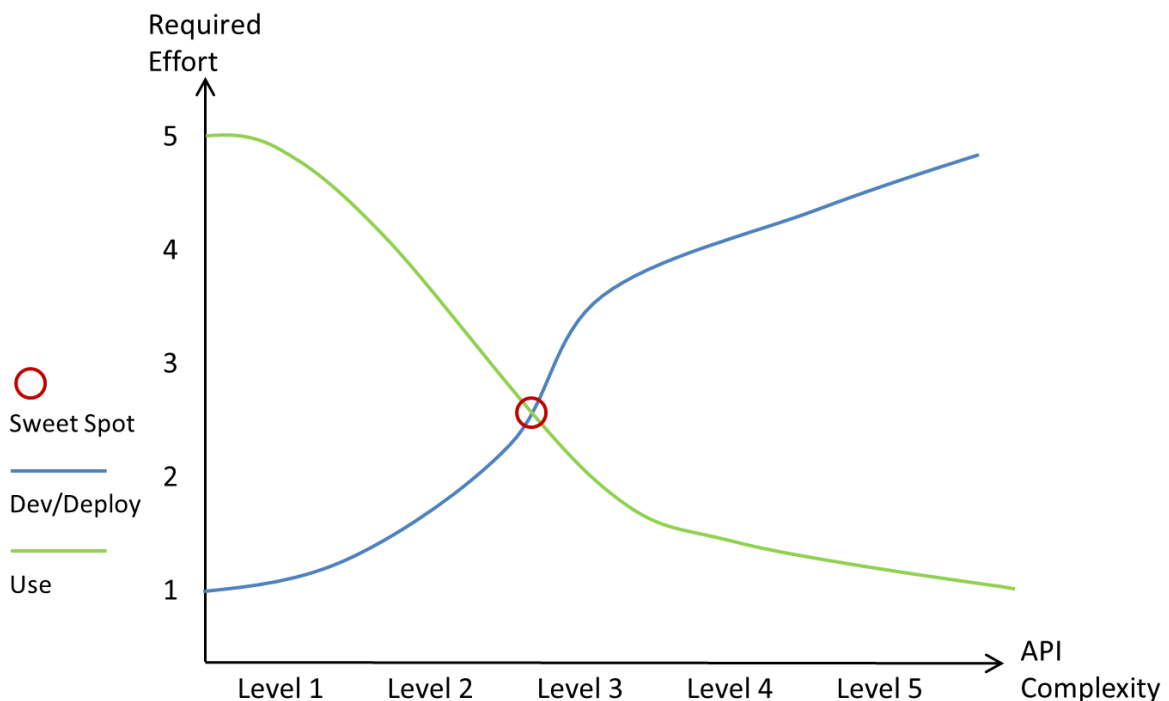


Figure 5: balance between deployment and use effort

Information on information gaps and other resource deficiencies collected through various aspects of the evaluation process will be collected and structured. This will help in the formulation of measures to guide the recommendation and training process.

2.5.4 Recommendations

Based on the evaluation analysis, the outputs will be summarized and made available in various forms in order to meet the information requirements of the stakeholder groups involved.

For administrative stakeholders concerned with providing the necessary resources for deployment and use of APIs, information on costs and benefits to be expected will be provided, allowing them to evaluate the potential of adoption of these technologies within their own organizations. Note that effort (costs) will not be explicitly quantified, but instead provided on a graduated scale in relation to known efforts. This is in line with the fact that an evaluation is always a comparison against something else, whereby this foil being evaluated against may be abstract criteria or existing systems such as previous generation OWS. This approach allows administrators to apply these relative results to the level of expertise available within their organizations based on experience in the provision of existing systems.

For data providers, the necessary materials required to empower Member State authorities to deploy their data via the new APIs, as well as to understand the benefits and pitfalls of data simplification options will be provided. These materials will include a rich collection of reference links, tutorials and technical guidance documents, providing a central access point for all required information. The various pieces of information collected within the GitHub Wiki in order to support the development and deployment teams will form the core of this resource. Where possible, we will endeavour to collect existing resources; where gaps have been identified, resources will be created either through the project team or triggered via other initiatives.

For data users, different levels of information will be required in order to address the different types of stakeholders. Based on feedback from events, information will be regularly integrated into the GitHub Wiki providing a core for the final resource collection. As described above pertaining to data providers, existing resources will be complemented as required.

A Webinar will be held in order to widely disseminate the outcomes of this project, whereby we may foresee some thematic splitting in order not to overload the Webinar format. Separate Webinars pertaining to provision of a specific API may be beneficial to empower Member State authorities to deploy and utilize the OGC APIs. Training material will be provided as required.

3 Deployment strategies for standard-based APIs

3.1 Introduction

This section reports on the results of Task 2 specified in the Technical Specification of the project. The project partners together with the data providers have researched deployment strategies of the chosen OGC APIs to be evaluated as described in detail in the previous section “2 Methodology for evaluation of standard based APIs”. Fraunhofer IOSB together with DataCove e.U. have developed a methodology for the successful deployment of new standards, whereby exemplary use cases guide the process of data mapping and query optimization until the required functionality is met. We built on this successful strategy, making minor modifications as required.

Different strategies were required depending on the maturity of the individual data providers. In some cases, fully INSPIRE compliant WFS2 services have long been deployed, so the endpoint need only be upgraded to OGC API - Features; should project resources suffice, alternative options for provision of OGC API - Features based on existing WFS2 deployments (i.e. LD-Proxy) were explored in order to determine if functionality and performance criteria could be considered equivalent. In other cases, while there are some basic OWS services in operation, these may not be fully aligned with the requirements ensuing from INSPIRE and the existing mapping may need to be extended and the configuration adjusted accordingly if the mapping differences are significant. On the other end of the spectrum, data sources were leveraged, which has not yet been available. In these cases, a basic harmonization process was performed, focusing on making the data available through the new APIs first, and implementing INSPIRE requirements where it makes sense.

For the steps #5 to #7, commonly utilized open source software has been applied, enabling a smooth transition to the new APIs and wide-scale uptake. Specifically: GeoServer as OGC API - Features implementation, and FROST with the STA 1.1 extension as required INSPIRE as SensorThings API implementation.

3.2 Data Providers

During the preparatory stage of the API4INSPIRE project, a set of data providers was selected to provide interesting data sources for experimentation and evaluation of the new APIs as well as to provide feedback on both on the effort required for provision of the new APIs as well as for insights as to the usability of these services.

3.2.1 Selection Criteria

The following criteria guided our selection process:

- Co-location of complementary datasets enabling Use Cases integrating multiple sources across data providers and Member States
- Balance of different INSPIRE Themes and data models
- Balance between OGC API - Features and OGC SensorThings API
- Wide range of development levels at the different data providers

The core criterion for identifying potential data providers lay in the suitability of the available data for implementation of realistic use cases. Co-location of datasets spatial scope was an essential criterion to ensure that data sources can be evaluated not only by themselves, but also pertaining to how well they can be integrated with data from other API sources.

Pertaining to the balance between types of data being provided as well as API specifications being utilized, care has been taken to complement highly dynamic sources typically provided via OGC SensorThings API with more static spatial sources provided via OGC API - Features. For selected data sources, the same data holdings will be exposed via alternate technologies, providing users with the possibility of direct comparison of API functionality.

In order to assure that the evaluation being performed pertains to a wide spectrum of potential data providers, data providers were also selected to reflect the wide range of stakeholder types described within the stakeholder analysis performed within “D1 - Methodology for evaluation of standard based APIs”. The usage types detailed in section “3.1.3 Usage Spectrum” not only illustrate the status quo pertaining to data usage, they also provide the starting point upon which the APIs will be deployed. Data providers have been selected to reflect the following existing usage and access types:

- Direct access to data source (DB)

- File-based data Sources
- Access to previous SOA+XML style OGC Web Services (OWS) with simple Features
- Access to previous SOA+XML style OGC Web Services (OWS) with complex Features
- Access to (non-standard) REST-Based APIs

3.2.2 Data Nests

As mentioned above, care was taken to identify complementary sets of colocated data, allowing users to integrate the data made available from the APIs provided for the creation of applications. Based on this requirement together with the other selection criteria described, six different data providers across Europe maintaining relevant data sources were contacted and cooperation agreements reached. These agreements allow us to provide the following Data Nests:

3.2.2.1 Airy Austria

In Austria, we have two quite advanced data providers integrated within API4INSPIRE, providing both spatial data via various OGC service types as well as dynamic measurement data, allowing users to integrate over both API types. In addition, air quality data has been added in an ad-hoc manner, initially from the Austrian Environment Agency (Umweltbundesamt), but then rapidly extended across central Europa via access to data from the European Environment Agency.

- **Austro Control** (ACG), responsible for air traffic control within Austria, has long had nearly all datasets relevant for the INSPIRE Theme Transport Networks - Air online via WFS in strict accordance with the INSPIRE data specifications. In addition, they have defined a solid URI based identifier scheme, allowing for direct resolution of the identifier URI to the underlying data object. Within API4INSPIRE, an OGC API endpoint will be configured in parallel to the WFS endpoint; additionally, the stand-alone OGC API - Features implementation for provision of simple features corresponding to SF-0 will be deployed on this data source. Thus, participants will be able to experiment with diverse provision options and evaluate the strengths and weaknesses of each of the three different systems serving the same dataset.
- **Zentralanstalt für Meteorologie und Geodynamik** (ZAMG), the Austrian agency for meteorology and geodynamics is responsible for the continuous measurement of meteorological parameters, as well as the provision of this data to the World Meteorological Organization (WMO). They are currently finalizing services for INSPIRE compliant WFS provision of this data, and will add an additional OGC API - Features endpoint to this configuration. In addition, the same dataset will also be provided in parallel via SensorThings API, underscoring the strengths of sensor centric services for the provision of dynamic measurement data.
- **Ad-hoc Austrian Air-Quality API** - while not formally planned, this additional source has been triggered by the Corona virus outbreak and the relationship with air quality. As near-real-time Austrian air-quality data was already available in accordance with INSPIRE and European reporting requirements via existing OGC WFS and SOS services, it was a fairly simple task to harvest these endpoints, transform the data as required and insert into a SensorThings API instance. This endpoint is being dynamically extended as additional data sources become available from sources such as the European Environment Agency. Details are available at <http://datacove.eu/ad-hoc-air-quality/>

Potential Use Cases in this context pertain to linking locations from air transport networks with data stemming from meteorological and air quality monitoring stations.

3.2.2.2 Urban Data Platform Hamburg

The Urban Data Platform of Hamburg Germany is supporting us through provision of their Smart City Sensors - deployed on SensorThings API. These range from car charging stations over the bike stations of "StadtRad" to various data of the Energy Campus of Hamburg University of Applied Science. A small selection:

Charging **Stations:**
[https://iot.hamburg.de/v1.0/Things?\\$filter=substringof\(%27Lade%27,name\)&\\$expand=Locations,Datastreams/Observations\(\\$orderby=phenomenonTime%20desc;\\$top=1\)&\\$count=true](https://iot.hamburg.de/v1.0/Things?$filter=substringof(%27Lade%27,name)&$expand=Locations,Datastreams/Observations($orderby=phenomenonTime%20desc;$top=1)&$count=true)

StadtRad **Locations** **and** **available** **Bikes:**
[https://iot.hamburg.de/v1.0/Things?\\$filter=properties/serviceName%20eq%20%27STA%20StadtRad%27;\\$ex](https://iot.hamburg.de/v1.0/Things?$filter=properties/serviceName%20eq%20%27STA%20StadtRad%27;$ex)

[pand=Locations\(\\$select=location\),Datastreams\(\\$expand=Observations\(\\$select=phenomenonTime,result;\\$orderby=phenomenonTime%20desc;\\$top=1\);\\$filter=properties/type%20eq%202\)&\\$count=true](https://iot.hamburg.de/v1.0/Things?$filter=name%20eq%20%27Energie%20Campus%20Hamburg%20University%20of%20Applied%20Sciences%27&$expand=Locations,Datastreams)

Data **from** **the** **Energy** **Campus:**
[https://iot.hamburg.de/v1.0/Things?\\$filter=name%20eq%20%27Energie%20Campus%20Hamburg%20University%20of%20Applied%20Sciences%27&\\$expand=Locations,Datastreams](https://iot.hamburg.de/v1.0/Things?$filter=name%20eq%20%27Energie%20Campus%20Hamburg%20University%20of%20Applied%20Sciences%27&$expand=Locations,Datastreams)

In addition, the City of Hamburg maintains over 400 spatial datasets, many already available online in various formats. As the Smart City Sensors would be very interesting in combination with routing information (and while INSPIRE TN-RO is available, there are no routing algorithms to date), we propose to complement the APIs provided with data stemming from OpenStreetMap (OSM)²⁸.

Conclusion: Lots of data on the bottom, applications on the top, and some APIs in between.

Potential Use Cases in this context pertain to Smart City applications, e.g. providing routing information to the nearest charging station. In addition, users can experiment with concepts stemming from building management systems utilizing the data from the Energy Campus.

3.2.2.3 Franco-Germanic Flow

For our third data nest, we've pulled together data from both the German and French sides of the Rhine River, integrating data from the German Bundesland of Baden-Württemberg (LUBW) with that stemming from the French Geological Survey (BRGM) and the French Office for Biodiversity (OFB) (via its environmental information systems research centre - INSIDE). This data covers spatial sources ranging from the basic river network information covered by the INSPIRE Theme Hydrography with additional river features supplied by the INSPIRE Theme Transport Networks - Water over water measurement stations provided in accordance with the INSPIRE Theme Environmental Monitoring Facilities to known flood risk zones provided under the INSPIRE Theme Natural Risk Zones, that will be exposed via OGC API - Features. This data will be complemented by dynamic data provided via SensorThings pertaining to both water quality and quantity.

Groundwater data will also be provided in the form of Hydrogeological units, their monitoring facilities served by OGC API - Features and their associated raw quantity observations provided by SensorThings API.

One interesting aspect of this combination of datasets will pertain to the overlaps in data maintained by different MS, with the French River Networks and Aquifers extending into Germany. A further duplication will be created pertaining to Environmental Monitoring Facilities, as we aim to provide these via both the OGC API - Features as well as SensorThings API; as far as possible these parallel datasets will provide cross references.

Potential Use Cases in this context pertain mostly to flooding within the Rhine catchment area, including both surface and ground water, but can also extend to navigability of French water transport networks based on water levels.

3.3 Deployment Methodology

Fraunhofer IOSB together with DataCove have developed a seven-step methodology for the successful deployment of new standards:

1. Definition of exemplary use cases
2. Mapping of required data to the data model of the standard
3. Derive queries from use cases aligned with query functionality of the API
4. Define expected performance of these queries
5. Either convert and import the data into a server/database which implements the API or create the necessary configuration files for accessing existing data sources
 - (a) If required, configure the service to the data model to be provided
6. Execute and measure the specified queries

²⁸ <https://ec.europa.eu/jrc/en/publication/comparing-inspire-and-openstreetmap-data-how-make-most-out-two-worlds>

7. Optimize queries where necessary

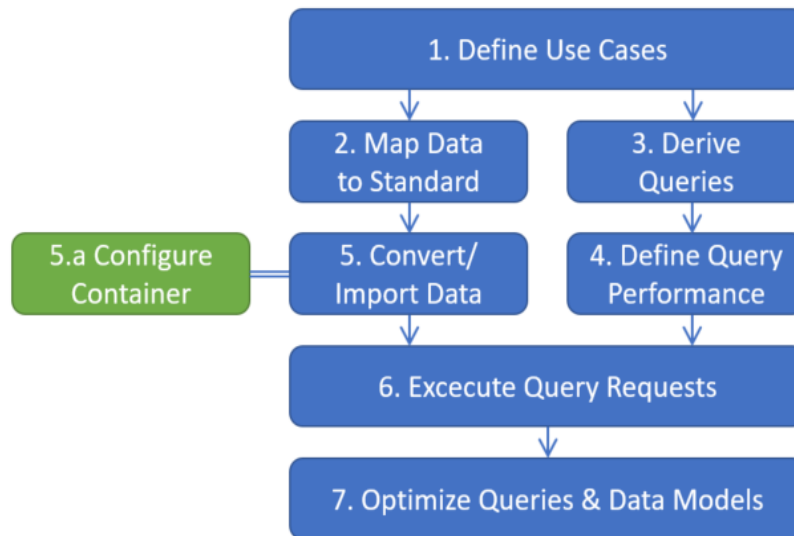


Figure 6: Deployment Methodology

While this methodology was initially defined for the deployment and benchmarking of SensorThings API, only minor modifications are required to extend this successful methodology towards provision of OGC API - Features. Modifications of this strategy will be required depending on the maturity of the individual data providers. The following list shows the current state-of-play among our data providers, as well as the deployment strategy selected:

1. Fully INSPIRE compliant services have been deployed [ZAMG, ACG, UBA]
 - (a) GeoServer in cooperating data providers: In this case, the existing deployment can be maintained, only the OGC API - Features extension needs to be added to the GeoServer Deployment.
 - (b) Deegree or other servers: for sources where we cannot easily add an additional OGC API - Features endpoint, conversion tools such as ld-proxy will be utilized.
 - (c) For the Austrian Air Quality service, we selected an alternative route, harvesting the available data from both the WFS2 and SOS2 endpoints and inserting into a SensorThings Server.
2. Spatial data is available within the data providers organization, but not or only partially exposed in a non-standardized manner. For this purpose, we have created reusable theme specific configurations for GeoServer in the form of preconfigured Docker images. In these cases, a basic harmonization process will be performed, focusing on making the data available through the new APIs first, and implementing INSPIRE requirements where it makes sense [BRGM, OFB]
 - (a) Where possible, GeoServer with the OGC API - Features extension will be deployed within the data providers infrastructure.
 - (b) Where this is not possible such deployments will be hosted within the project
3. SensorThings endpoints are available, but not fully aligned with INSPIRE requirements [BRGM, CH]
 - (a) Where possible, the additional data elements required by INSPIRE will be added to the SensorThings deployments in cooperation with data providers staff.
 - (b) Should provision of full INSPIRE requirements not be possible, these endpoints will still be included, whereby these will allow a comparison of usability between base SensorThings vs. SensorThings including the INSPIRE extensions.
4. Dynamic data is available within the data providers organization, but not or only partially exposed in a non-standardized manner [BRGM, OFB, LUBW, EEA]
 - (a) Where possible, the FROST SensorThings implementation will be deployed within the data providers infrastructure.
 - (b) Where this is not possible such deployments will be hosted within the project

In addition, a standalone OGC API - Features implementation, only supporting provision of simple features in accordance with the OGC SF-0 profile, is being developed within this project, providing insights into difficulties encountered by developers not familiar with the various underlying standards. This implementation will be deployed on selected datasets in addition to the options described above, allowing users to experiment with the strengths and weaknesses of such a simplified system.

The extent to which individual data provider staff will be integrated in the deployment work will be strongly dependent on the level of expertise available at that data provider. In the case of highly technical data providers such as Austro Control or BRGM, we will be working hand-in-hand with data provider staff on upgrading their existing deployments to the new APIs. In addition, staff from these institutions will participate in the heuristic expert evaluation as well as the peer reviews. For other data providers such as the French Biodiversity Office OFB, far more support will be required; in those cases we will need to provide at least initial configurations together with documentation on how to extend this core configuration, providing immediate feedback on the quality of the training and guidance materials being provided under D5.

Initial data analysis and mapping will be done on systems of IOSB or GeoSolutions, while the final deployment used for evaluation is done on the systems of the data provider. If the data provider cannot provide infrastructure to deploy the services required in this project, the services will be deployed on a Kubernetes cluster at Fraunhofer IOSB and maintained at least one year past contract finalization.

Although the actual deployments in the project will be done using FROST-Server and GeoServer, much of the deployment methodology, like the data mapping procedures and container deployment technology, is not implementation specific. The evaluation will foremost focus on these general aspects, implementation specific aspects will be clearly flagged as such.

The technologies listed here are described in greater detail in the sections below.

3.3.1 GeoServer Deployment

GeoServer is a free and open source project designed for geospatial data interoperability and is an implementation of several OGC standards such as WFS, WMS, WCS and more recently the OGC API, which includes OGC API - Features. GeoServer is built on top of the GeoTools library and integrates natively with GeoWebCache, providing a flexible and easy to use tile cache mechanism. GeoWebCache implements several standards including the WMTS service.

In GeoServer, modularity and flexibility are first class citizens, with the default package caring for most common needs and extensions supporting a variety of additional functionality. The former includes the most common data stores (e.g. Shapefile, GeoPackage, PostgreSQL and GeoTIFF) and OGC services (e.g. WFS, WMS and WMTS).

GeoServer offers a large amount of extension points allowing anyone to add new functionality in a modular way. Some of those extensions are contributed back to the GeoServer Community, becoming official GeoServer plugins: at present around eighty plug-ins are available for GeoServer. These plug-ins extend GeoServer in a variety of ways: adding support for new data sources, adding new services or extending existing ones, adding new security methods, and so on.

Plug-ins can be divided into two main categories: community modules and extensions. Community modules are generally considered experimental in nature and can be undergoing significant development. Once a community module development is stable and has proven to be useful it can be promoted to extension. To become an extension a community module must have an official maintainer, pass the code quality requirements and have official documentation. If an extension is judged to be important enough or is commonly used, it can be promoted to a GeoServer core module.

Three GeoServer plugins will be particularly relevant for this project: INSPIRE extension, OGC API community module and App-Schema extension. The INSPIRE extension allows GeoServer to be compliant with a number of INSPIRE Services, for example, the View Service and Download Service. When this extension is present, new UI elements allowing the configuration of extra metadata and INSPIRE specific options are available; relevant OGC services outputs will also be modified in accordance with the INSPIRE specifications. In addition, we will experiment with the JSON-LD plugin, as this allows for stronger linkages between features and better indexing of data by search engines crawlers which should enhance data reuse.

The OGC API community module exposes a variety of services based on the new OGC API commons and the OGC API - Features 1.0 service, delivering feature data. The OGC API - Features 1.0 is thus a replacement for the existing WFS services, although GeoServer will still support the existing WFS services (1.0.0, 1.1.0 and 2.0.0); it will also be possible to use them alongside with the OGC API - Features.

GeoServer core only allows the publishing of features compliant with level SF-0 of OGC Simple Features profile, the App-Schema extension adds to GeoServer the capability of publishing features compliant with level SF-1 and SF-2. This extension brings also the possibility of building features by defining the mappings between a data source, typically a relational database, and a target GML schema. For most of the use cases App-Schema can use the database as is, although for certain corner cases SQL views must be created prior to configuration.

It is worth mentioning that although several data source integrations exist with App-Schema, the most mature and feature rich one is the PostgreSQL \ PostGIS integration. Hence PostgreSQL \ PostGIS will be the target data source for the GeoServer deployments in the context of this project.

3.3.1.1 OGC API - Features in GeoServer

OGC is defining a new set of services to replace the currently used OWS, that date back 20 years. These new services are collectively identified as OGC APIs, and are based on different concepts compared to the old OWS based services, in particular:

- They are Web APIs, sometimes referred to as RESTful services;
- Each service is described by an OpenAPI document;
- Each of them is geared towards JSON based representations of resources, at least for meta information, while allowing other representations as extensions;
- HTML representation is also promoted as a way to make the service easier to learn and access from a browser;
- Each service has a minimal core, and numerous optional extensions to add functionality.

For example, OGC API – Features 1.0 core does not mandate any output format (GeoJSON being recommended), only supports CRS84, provides limited filtering abilities (time, space, single attribute equality), and requires no schema for the data being published. Extra coordinate systems support, full-fledged filtering and transactions are currently being developed as extensions and will be delivered to the public sometime in the future.

The service is based on a small set of resources:

- /, the landing page, providing access to all other resources
- An OpenAPI service description, which is linked from the landing page, and does not have a fixed location, though many implementations place it at /api
- /conformance, a set of conformance classes, a list of URIs identifying the capabilities of the service
- /collections, providing a list of available collections (previously known as feature types)
- /collections/{collectionId}, providing a description of the single collection
- /collections/{collectionId}/items, providing access to the actual data, often in GeoJSON format, although no formats are mandated by the specification
- /collections/{collectionId}/items/{itemId}, providing access to a single feature

The GeoServer implementation provides full core support; all resources are available either as JSON or HTML. A minimal CQL filtering extension has been added, based on the work done at the 2019 Washington STAC and OGC API sprint, while support for the CRS extension is still not present, but incoming in the next few months. As the specification for OGC API Part 3 Common Query Language is still very much a work-in-progress with results not expected until the final quarter of 2020, filter functionality will not be covered within the evaluation of OGC API.

3.3.1.2 App-Schema

GeoServer distinguishes between Simple Features, features whose properties may contain only values of XML simple types, and Complex Features, features whose properties may contain values of XML complex types and link other features. The former roughly corresponds to compliance level SF-0 of OGC's Simple Features profile (OGC® 10-100r3), while the latter corresponds to level SF-1. By default, GeoServer only supports simple features. In the most common use case, GeoServer is connected to a data source such as a relational database and each table containing geographic data is automatically mapped to a simple feature type.

Things become more complicated when the physical data model is more complex. For example, if multiple relational database tables are required for a single layer, simple features are often not enough; thus, we must create complex features, which support nested objects (list of objects containing other objects or other lists of objects). Publishing complex features requires more complex configuration work from the user, at least a target GML schema and App-Schema mappings between the data source and the target GML schema must be provided.

App-Schema will produce a stream of complex features that will be used by the GML and GeoJSON encoders, which are used by WFS and OGC API – Features, to produce the respective responses. The produced GML responses will match the target GML schema that was used to create the App-Schema mappings. The GeoJSON responses will be produced based on a set of rules defined based on the original target GML. It is worth mentioning that even if we are only interested in the GeoJSON responses, the App-Schema mappings still need to be defined between a data source and target GML schema.

App-Schema will also be utilized to interpret queries performed against the published complex features. In most situations, App-Schema will be able to interpret the query and translate it to one or more SQL queries that will be sent to the database to retrieve the necessary data. In certain cases, App-Schema will not be able to translate the query to SQL; in those situations, all the data will be retrieved from the database and the filter executed in memory. It should be noted that this requires extra memory consumption and will add a significant performance penalty.

3.3.1.3 New guidelines for modern standards

The new guidelines based on modern standards, GeoJSON and OGC API – Features, that are being proposed within the INSPIRE community will be considered during this work, foremost:

- [Alternative encodings for INSPIRE, the GeoJSON encoding;](#)
- [Setting up an INSPIRE Download service based on the OGC API-Features standard.](#)

The implications of these new guidelines' implementations are not only purely technical and raise concerns regarding the reusability of the modelling and implementation efforts already made by INSPIRE implementers.

Ideally it should be possible to migrate existing implementations to these new API's, leveraging on the investments already made. At the same time, new implementations should be free to adopt these new guidelines without being tied to the complexity of previous data models. This raises a few technical challenges that will be investigated during this project.

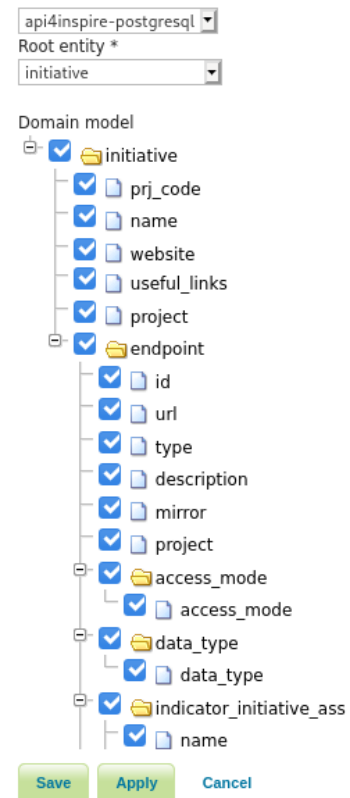
The implementability of those guidelines will also be evaluated, using GeoServer as a sandbox. We already foreseen a few enhancements that may be done to GeoServer to simplify the publishing of complex data models.

3.3.1.4 New provision approach – SmartDataLoader and Templating

Based on experience in providing data utilizing GeoServer App Schema, a powerful but often painful process, we have created an extension allowing for an alternative more intuitive approach towards data provision, that we hope will ease the data provision process and enable data holders to expose more data more effectively.

The first step of this approach has been implemented as the [SmartDataLoader](#) Extension to GeoServer. Similar to the existing functionality allowing a user to expose the content of a database table as a Simple Feature directly in GeoServer, SmartDataLoader extends this approach to Complex Features. The extension enables GeoServer to read a database structure, starting from a root table specified and traversing foreign key relations to further tables, then automatically generate the App-Schema mappings and target local GML schema²⁹ required by App-Schema. The user can interactively perform pruning, excluding relations or columns that are not relevant until all required data is exposed. Once this configuration is complete, the SmartDataLoader output is deployed as an App-Schema datastore, the output available both as WFS and OAPIF

This functionality is complemented by the GeoServer [Features-Templating](#) Extension also developed under this project, allowing users to customize the desired output based on a template using a *what you see is what you get* approach. All base text from the template is utilized in data provision, only dynamic aspects of the data must be configured. This will significantly simplify the publishing of complex data and provide a very flexible customization for output without a performance penalty. At present this functionality is only available for GeoJSON and JSON-LD, but work is progressing on extending this functionality to XML and HTML formats.



²⁹ The generated target local GML schema is not meant to be used for data exchange or to be public, it should be seen as an internal configuration file for App-Schema.

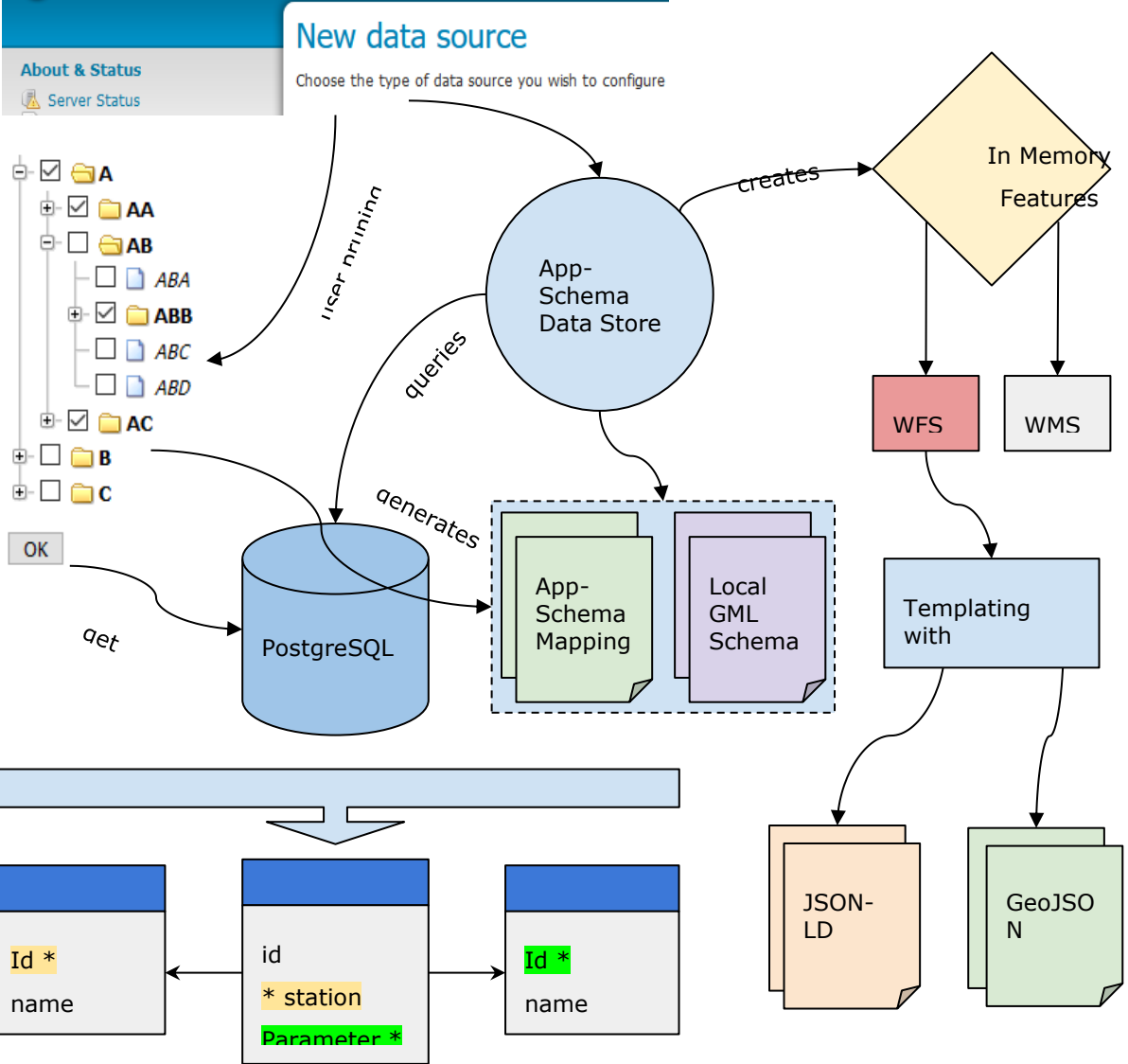


Figure 7: GeoServer New Smart App-Schema Data Source

3.3.1.5 Preconfigured GeoServer Black Boxes in Docker

In typical GeoServer deployments, the GeoServer instance is configured to access the data provider's native database source via the App-Schema configuration files. While this is surely the best mode of deployment pertaining to efficient data provision of up-to-date data, the downside is that these configurations can rarely be reused as they strongly reflect the data providers in-house infrastructure.

In order to allow for rapid deployment, we have had success in the utilization of preconfigured GeoServer Black Boxes. Under this deployment paradigm, the source database accessed by GeoServer is structured in close alignment with the final GML output format; an App Schema configuration provided that allows GeoServer to provide these FeatureTypes. This configuration is then packaged for simple deployment via docker.

The data to be exposed can then be provided to this preconfigured GeoServer Black Box in two manners:

- Import: The required data can be transformed and imported directly into the foreseen database tables. The advantage of this option is that the GeoServer deployment can be hosted on separate infrastructure, which has security advantages; the disadvantage of this option pertains to highly dynamic data, as this would require regular updates to the system.

- Views: As an alternative to data import, the required data can also be made available in the form of database views that mimic the structure of the database tables foreseen in the GeoServer Black Box configuration. Depending on the size of the dataset to be provided, one may need to utilize materialized views for performance criteria. The advantage of this option is that dynamic data is automatically updated; the disadvantage of this option pertains to large datasets that could lead to performance issues with the views as well as security considerations.

Alternatively, these preconfigured GeoServer Black Boxes can also be used as a starting point for deployment directly linked with a data provider's data sources. While the modification of the App Schema configuration does require a degree of technical experience, it is far simpler than having to start from scratch as most of the difficult aspects of the configuration have already been correctly performed.

3.3.2 OGC API – Features Development - OGC API Simple

In order to ascertain the complexity of developing an OGC API - Features conformant API on top of existing data holdings, a stand-alone development of the OGC API will be implemented. As this exercise should reflect the issues encountered by non-experts attempting to create correct APIs based on the existing standards and guidance documents, we have started a collaboration with a vocational informatics school in Vienna. A team of interested students has been created, and implementation work initiated based on the following simple specification requirements:

- Deploy on an existing geospatial database
- Manage connections to geospatial databases based on PostGIS and SpatiaLite DB
- Expose columns as properties in SF-0
- Select the relevant geometry column and expose that as Geometry
- Map table name to collection name if desired
- Map column names to properties names if desired
- Hide or show tables and/or columns if desired
- Provide all through a user interface, configuration details stored as json config files
- Additional collection types can be created through SQL statements
- Filtering functionality (limit, bbox, feature properties and combinations)
- Display configured collections in a simple map view
- Display collections and items as HTML and JSON as far as reasonable
- Docker-enabled installation

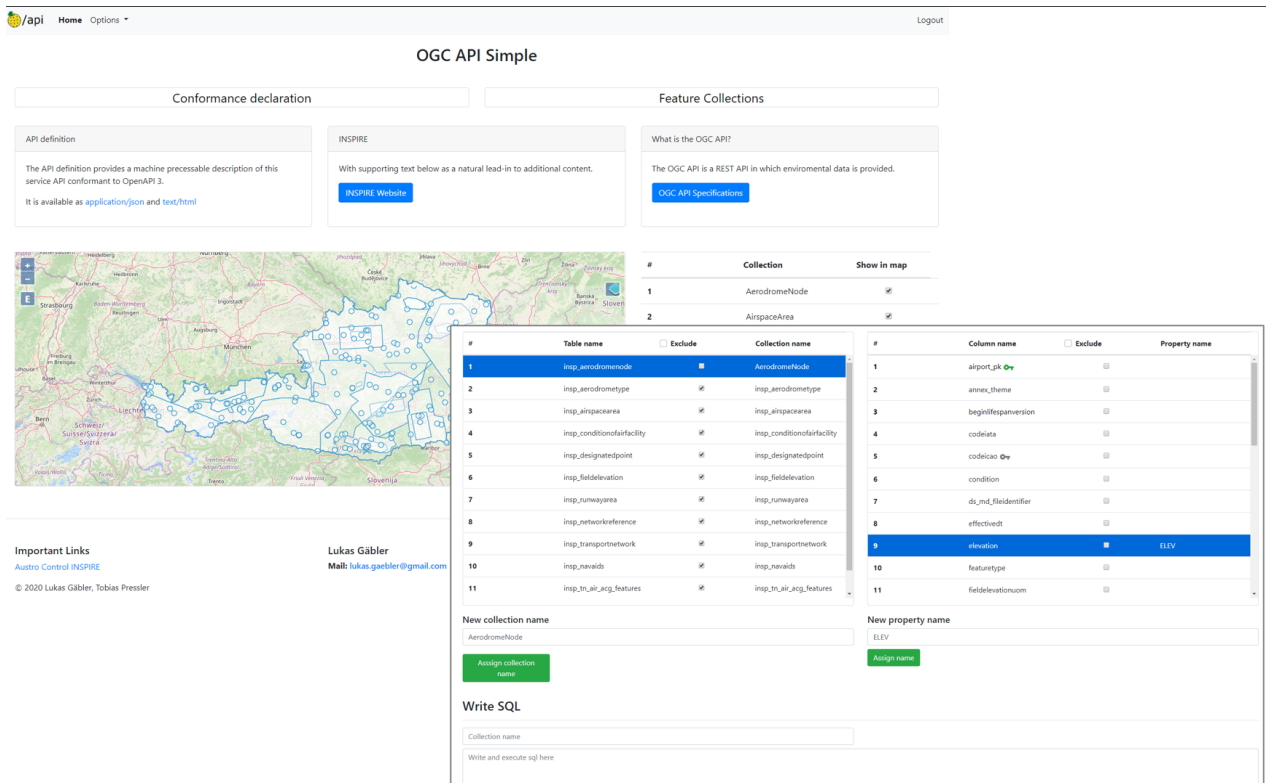


Figure 8: OGC API - Simple Interface

The application will be a Java Servlet and will be developed and tested using the Apache Tomcat web server. The backend of the OGC API Simple will be developed as a Java Spring application and the frontend will use the TypeScript based web framework Angular (v.8). The web interface shown in Figure 8: OGC API - Simple Interface will get the data to display from the Java Spring REST API which will be implemented according to the OGC API Core specification and some more REST paths (e.g.: for the authentication).

This simple OGC API demonstrator will be created in an iterative manner. In an initial step, the collection name will correspond to the name of the table in the database, while the individual attribute names will be the same as the columns of the selected table. In further iterations, it will be possible to configure these names as required. All source code is available via the project GitHub under the OGCAPIsimple folder.

For encoding, this implementation will rely on simple feature encoding standards and only provide GeoJSON and JSON encoded data. At present, this implementation is limited to PostGIS and Spatialite databases. Austro Control has kindly provided dumps of their repositories for this development work. While this work will foremost concentrate on Austro Control Air Transport Data, other data sources will also be demonstrated.

3.3.3 LD-Proxy

In order to leverage existing WFS2 implementations not stemming from GeoServer deployments as well as GeoServer deployments where we cannot include the OGC API extension, we will provide an instance of LD-

Proxy that can perform on-the-fly conversion. Adding services to LD-Proxy is a very simple undertaking, all one needs to provide in order to configure a new service interface is a name and the URL of the WFS2 Endpoint.

After the Service Endpoint has been added to the LD-Proxy instance, a list of the feature types available from that end point are displayed.

The first time one selects a feature type, LD-Proxy requests information on the feature types being provided

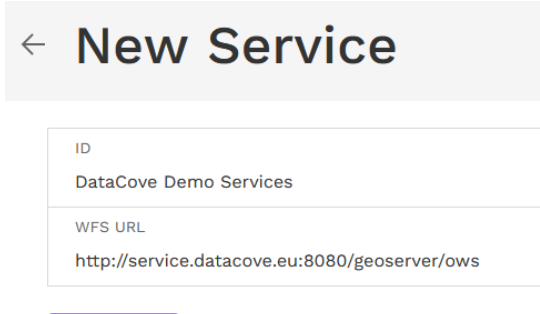


Figure 9: LD-Proxy - add new service

using the WFS DescribeFeatureType request. Please note that the functionality of this service request is essential for LD-Proxy to function, some WFS servers unfortunately do not correctly support this.

Once LD-Proxy has retrieved all relevant information on the feature types being provided by the underlying WFS2, the structure of the feature types can be displayed. In the example in Figure 6, the attributes for the type tn-a:AirNode are displayed.

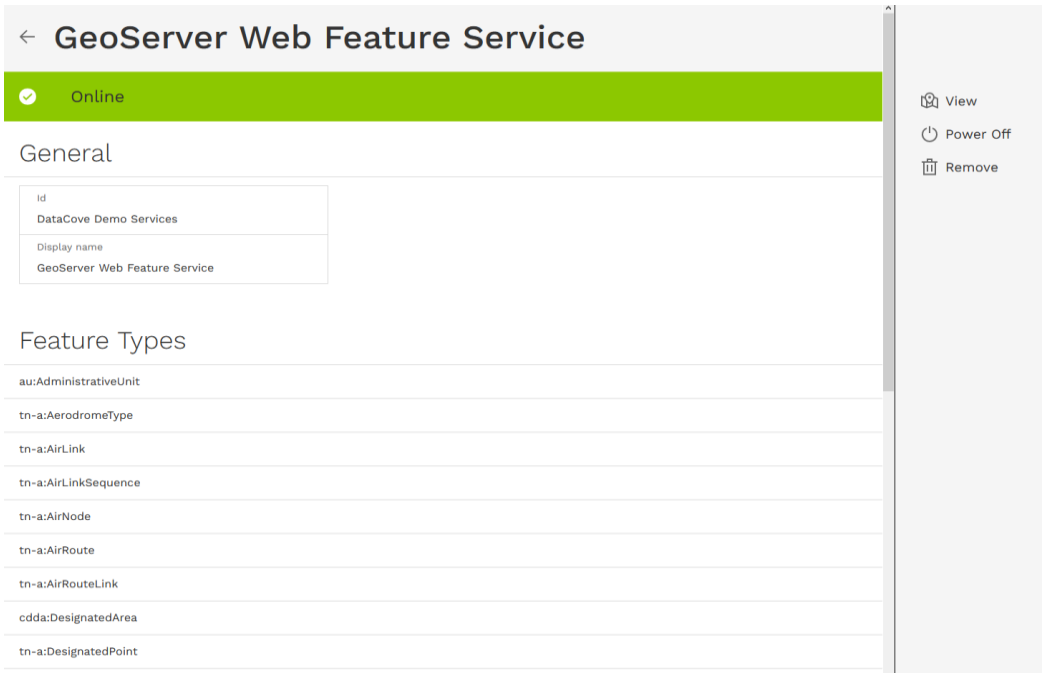


Figure 10: LD-Proxy - show feature types

← tn-a:AirNode

General

| | |
|--------------|--------------|
| Id | AirNode |
| Display name | tn-a:AirNode |

Mapping

| |
|---|
| — tn-a:AirNode |
| <input type="checkbox"/> @gml:id |
| <input type="checkbox"/> net:beginLifespanVersion |
| — net:inspireId |
| — base:Identifier |
| <input type="checkbox"/> base:localId |
| <input type="checkbox"/> base:namespace |
| <input type="checkbox"/> base:versionId |
| <input type="checkbox"/> net:endLifespanVersion |
| <input type="checkbox"/> net:geometry |
| — tn:geographicalName |
| — gn:GeographicalName |
| <input type="checkbox"/> gn:language |

Figure 11: LD-Proxy - feature attributes

Where applicable, information on the data type of the individual attributes are mapped to common vocabularies such as schema.org. This is of special relevance when accessing the available data in JSON-LD format.

| |
|---|
| LD Type |
| http://schema.org/Place |

Figure 12: LD-Proxy - schema mapping

Once the LD-Proxy endpoint has been configured, one can then proceed to view or access the data in GML, GeoJSON or JSON-LD format.

tn-a:DesignatedPoint

Filter

« < 1 > »

bemki.100.2018-07-26

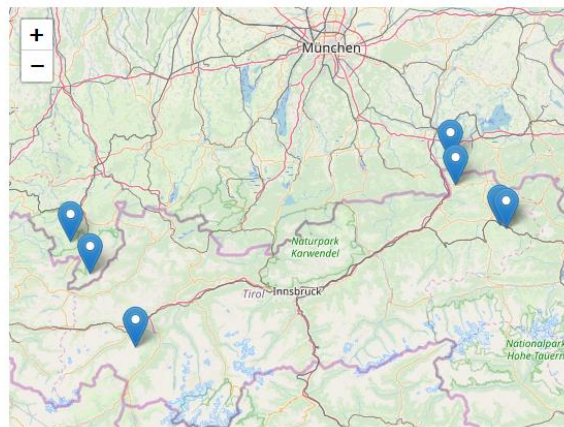
beginLifespanVersion Monday, 23 October 2017, 22:00:00
inspireId.localId 100
inspireId.namespace http://guid.datacove.eu/TNA_MD/tn.DesignatedPoint
inspireId.versionId 2018-07-26
validFrom Wednesday, 25 July 2018, 22:00:00
significantPoint true
designator BEMKI

beras.121.2018-07-26

beginLifespanVersion Monday, 23 October 2017, 22:00:00
inspireId.localId 121
inspireId.namespace http://guid.datacove.eu/TNA_MD/tn.DesignatedPoint
inspireId.versionId 2018-07-26
validFrom Wednesday, 25 July 2018, 22:00:00
significantPoint true
designator BERAS

birgi.122.2018-07-26

beginLifespanVersion Monday, 23 October 2017, 22:00:00
inspireId.localId 122
inspireId.namespace http://guid.datacove.eu/TNA_MD/tn.DesignatedPoint
inspireId.versionId 2018-07-26
validFrom Wednesday, 25 July 2018, 22:00:00



```
JSON Rohdaten Kopfzeilen
Speichern Kopieren Alle einklappen Alle ausklappen JSON durchsuchen
type: "FeatureCollection"
features:
  e:
    type: "Feature"
    id: "bemki.100.2018-07-26"
    geometry:
      type: "Point"
      coordinates:
        0: 10.183317
        1: 47.448136
    properties:
      beginLifespanVersion: "2017-10-23T22:00:00Z"
      inspireId.localId: "100"
      inspireId.namespace: "http://guid.datacove.eu/TNA_MD/tn.DesignatedPoint"
      inspireId.versionId: "2018-07-26"
      validFrom: "2018-07-25T22:00:00Z"
      designator: "BEMKI"
  i:
    type: "Feature"
    id: "beras.121.2018-07-26"
    geometry:
      type: "Point"
      coordinates:
        0: 12.503366
        1: 47.486322
```

Figure 13: LD-Proxy - TN-A Data

3.3.4 FROST Deployment

For data storage, FROST uses the PostgreSQL database server with PostGIS geospatial extension, whereby the data model must be aligned with the underlying data model put forth in the STA standard. Additional attributes as required by INSPIRE can be supplied in extended properties sections, as described in “Extending INSPIRE to the Internet of Things through SensorThings API”³⁰. These extended properties sections are available in the 1.1 version of the STA standard, which FROST already implements.

3.3.4.1 FROST-Maintained Database

By default, FROST creates and maintains its own tables in the database, and users do not need to deal with the details of these tables besides adding use-case specific indices to speed up specific queries required for the use case. When using FROST (or any other SensorThings API server implementation) in this way, all interactions

³⁰ Extending INSPIRE to the Internet of Things through SensorThings API, <https://www.mdpi.com/2076-3263/8/6/221>

with the data can happen through the API itself. New entities are added and updates are made through the API, and data is read through the API.

3.3.4.2 Custom Database

Is it also possible to not have FROST maintain its own database tables, but instead mimic those tables using database views and have FROST operate on those views? The big advantage of this approach is that any existing applications and procedures can remain in place. Any new data automatically appears in the SensorThings API interface and does not need to be duplicated.

For most deployments, we have been successful by creating database views on the source data aligned with the database tables expected by FROST; however, in the case of massive data holdings, it has been shown necessary, for performance purposes, to create a physical copy of the source data, either using materialized views or transfer the data to FROST database tables.

3.3.4.3 Deployment Options

The FROST server software is deployed separately from the database, and can be deployed in several ways:

- As java “war” file that can be deployed on an existing Tomcat or other servlet container. This option is ideal for organisations that already have other services running on Tomcat and thus have experience administering Tomcat.
- Using the Docker image, on any Docker environment. The FROST Docker images allow for an easy way to deploy and update FROST. In a Docker environment on a cluster it is possible to set up automatic horizontal scaling.
- Using the Helm³¹ package manager on a Kubernetes cluster. This option is the easiest for those organisations that use Kubernetes to manage their Docker environment.

3.4 Data Provider Specific Deployment

Within the API4INSPIRE project, we have brought together a set of different data providers with a wide spectrum of levels of expertise and organizational infrastructure. This allows us to illustrate various deployment options for both OGC API - Features as well as SensorThings API. At the one end of this spectrum, we find data providers such as Austro Control, who have long fulfilled their INSPIRE Requirements pertaining to provision of GML encoded data via WFS utilizing GeoServer and only needed to add the OGC API - Features extension to their existing infrastructure, or the City of Hamburg with a rich smart sensor landscape long deployed on SensorThings API. At the other end of the spectrum are data providers who have to date relied on internal or national data structures and formats, where the necessary data sources must first be identified, mapped to INSPIRE requirements, the data transformed as required by the data models, and finally the necessary software for the selected API deployed. Going beyond this foreseen spectrum of data provision options, based on current events triggered by the Corona virus crisis, we’ve started leveraging existing non-harmonized data sources on *in situ* air quality measurements from both national and European data sources, transforming and providing this data via SensorThings API.

In the following sections, we detail the deployment methodology to be utilized for which datasets stemming from which data providers.

3.4.1 Austrian Meteorological Agency – ZAMG (AT)

The Austrian Meteorological Agency has over a century of experience in the collection and administration of meteorological data. Thus, this institution has a wide range of expertise at its disposal pertaining to all aspects of meteorological data management and provision. Information on meteorological measurement stations as well as the data ensuing from these stations is currently available in the form of a WFS2 deployment utilizing GeoServer. This endpoint will be extended in cooperation with ZAMG staff to also provide an OGC API endpoint. In addition, SensorThings API will be deployed at ZAMG.

- Meteorological Measurement Stations [EF]: SensorThings API and OGC API - Features

³¹ <https://helm.sh/>

— Meteorological Data [AC/MF]: SensorThings API and OGC API - Features

3.4.2 Austro Control – ACG (AT)

Austro Control (German: Österreichische Gesellschaft für Zivilluftfahrt) is the air navigation services provider that controls Austrian airspace. Its location and jurisdiction is Vienna, with the physical offices also being located in Vienna. Austro Control is the official and certified air navigation service provider for Austrian airspace. In this role, Austro Control manages all data pertaining to air traffic networks in Austria, and currently provides this data via an INSPIRE compliant endpoint running on GeoServer. This endpoint has been extended by Austro Control staff to also support OGC API - Features. In addition, Austro Control is sponsoring the development of a simplified OGC API - Features implementation that supports the provision of data conforming with the SF-0 simple features specification; further information and source code is available via the project GitHub.

— Air Traffic Network [TN-A]: Currently available as WFS2 and OGC API - Features on GeoServer. In addition, this data will be made available via the simple features implementation of OGC API - Features.

While the WFS2 and OGC API - Features endpoints will continue to be hosted by Austro Control via their corporate site, the simple features implementation of OGC API - Features will be hosted by Fraunhofer IOSB.

Currently available endpoints:

WFS2: <https://sdigeo-free.austrocontrol.at/geoserver/tn-a/wfs?service=WFS&version=2.0.0&request=GetCapabilities>

OGC API: <https://inspire.austrocontrol.at/ogcapi/ogc/features>

3.4.3 Umweltbundesamt - UBA (AT) & European Environment Agency - EEA (EU)

The provision of air quality data was not initially foreseen within the API4INSPIRE project. However, due to recent events pertaining to the spread of Covid-19, this data source has been added in an ad-hoc manner without integration of the data providers. This was possible in part due to the fact that the data has already been harmonized due to the European Air Quality Directive (AQD) data formats, that were created through the extension of the INSPIRE Environmental Monitoring Facilities Theme.

In an initial step, air quality data was harvested from the near-real-time services offered by Umweltbundesamt, the Austrian Environment Agency, that provides monitoring stations via WFS2 and near-real-time measurement data from these stations via OGC SOS. This data was inserted into a SensorThings API server via native SensorThings API requests, the endpoint has been freely published.

The importer runs in two stages. First it queries the WFS to find all the Stations (Things), Processes (Sensors), ObservedProperties, SamplingPoints (Datastreams), etc. and creates the entries in the SensorThings service. This needs to be done only once, but it can be done again when the source data changes, since the importer is smart enough to update data that has already been imported. The second step is to import the actual Observations from the SOS. On an hourly basis it fetches, for each Datastream, the new Observations since the last Observation that was imported. On a daily basis it fetches all Observations of the last three days, to check if any Observations were changed after they had been imported.

Based on increasing interest by other European Union Member States, we then turned to the sources offered by the European Environment Agency. This task proved to be the bigger challenge; while the concepts from the AQD data formats have been maintained, the structure foreseen by the data models was removed, with the resulting data only available in not-standardized csv formats. While this did cause some issues in the import process, they could finally be resolved.

Since the source data are published in a different manner, the importer also works slightly differently. For the EEA data, the importer starts from the CSV files containing the Observations. When importing an Observation, the importer checks if the Datastream and other metadata entities already exist for this Observation. If not, the importer downloads the CSV file containing the station metadata for all stations in Europe, and uses this data to generate the missing metadata entities.

The total number of Observations, for the ObservedProperties SO₂, PM_{2.5}, PM₁₀, NO, O₃, CO, CO₂, NO₂ and 'NO_x as NO₂' in the countries of Austria, Switzerland, Germany, Italy and France, for the time period of 2018 to now is about 100 million. These are grouped into 8185 Datastreams, for 2117 measurement stations (Things).

At present, we are in negotiations with other national data providers interested in adopting SensorThings API for direct provision of air quality data from their respective institutions.

Currently available endpoints:

STA: <https://airquality-frost.docker01.ilt-dmz.iosb.fraunhofer.de/v1.1>

Viewer: <https://wg-brgm.docker01.ilt-dmz.iosb.fraunhofer.de/servlet/is/121/>

3.4.4 City of Hamburg – CH (DE)

The City of Hamburg has long seen the potential of Smart City technology, and long been involved in diverse Smart City initiatives, successively extending their smart sensor infrastructure to an ever widening usage area. These efforts have gained the city accolades such as being Germany's smartest city. While formally much of this data goes beyond the formal INSPIRE specifications, we believe that it nicely illustrates further potential of such a consistent European SDI merging into an Urban Data Platform and highlights the challenges of integration between the formally specified core of INSPIRE with additional data sources.

- Smart City Sensors [EF++]: A great deal of real-time information pertaining to mobility within Hamburg is already available via SensorThings API. This includes data on the availability of charging stations and data stemming from the StadtRad CityBike initiative.
- Energy Campus [EF++]: while stemming more from building management, the data from the Energy Campus of the Hamburg University of Applied Sciences provides a wide field for experimentation with the SensorThings API.
- Road Networks [TN-R]: This data is currently available via an INSPIRE compliant WFS2 from the regional authorities. While we'd initially hoped to make this data available via LD-Proxy this has not been possible due to issues in the underlying WFS configuration. As an alternative to this source, we are proposing the use of data from OpenStreetMap (OSM), which has the added benefit of providing routing functionality that will be beneficial to the planned use cases.

Currently available endpoints:

STA: <https://iot.hamburg.de/v1.0>

Viewer: <https://wg-brgm.docker01.ilt-dmz.iosb.fraunhofer.de/servlet/is/110/>

3.4.5 French Geological Survey – BRGM (FR)

BRGM has long been involved in pushing the envelope pertaining to the possibilities of environmental data provision. They are intensely involved in various OGC initiatives pertaining to the exploration of novel data provision methodologies as well as providing valuable input to the core standardization activities within the OGC. BRGM staff was integral in the creation of relevant INSPIRE data specifications, and this organization has provided funding for many of the tools the INSPIRE community is coming to rely upon pertaining to access to and processing of complex spatial features. Based on this expertise, they will also be providing support towards the French Office for Biodiversity, that does not have such strong resources at its disposal, but hosts complementary datasets.

BRGM is the French Geological survey but its activities go way beyond the "traditional" aspects of Geology and Mineral Resources. Its Risk division has an important activity and BRGM IT division hosts the French national silo for risks (<https://www.georisques.gouv.fr/>).

Its Environmental & Ecotechnologies division provides expertise on Groundwater but also on Ecotechnologies, Soils, etc. Associated with this, the BRGM IT division hosts many data portals of the French Water Information System but also for the national polluted soils Information System on behalf of the French ministry of environment.

Thus, the available data sources pertain to geological structures as well as the subsurface aspects of water such as the location of aquifers and the measurement of groundwater quality and quantity. The pertinent INSPIRE Themes for this data are Geology (Hydrogeology package and its equivalent in the international OGC standard GroundWaterML2 - OGC 16-032r2 - where BRGM and JRC pushed the necessary elements for INSPIRE compliance) for the spatial feature types, Environmental Monitoring Facilities for the complementary groundwater quality and quantity measurements.

For the provision of OGC API - Features, we will start with preconfigured GeoServer Black Boxes in Docker, whereby there are plans to successively modify the configuration to directly access the native data sources available within BRGM.

For the provision of dynamic measurement data via SensorThings API, BRGM started in 2018 with porting its infrastructure collecting raw Groundwater observation from GPRS sensors and exposing it according to O&M from OGC:SOS to OGC SensorThings API. This activity is directly contributing to the API4INSPIRE project.

Within API4INSPIRE, the following datasets will be made available via the following APIs:

- Groundwater Quantity [EF]: both OGC API - Features and SensorThings API
- Hydrogeological Unit [GE & or OGC:GroundWaterML2]: OGC API - Features
- Flood Risk [NZ or AM]: OGC API - Features; candidate for SF-0 encoding

Currently available endpoints:

STA: <https://sensorthings.brgm-rec.fr/SensorThingsGroundWater/v1.0/>

Viewer: <https://wg-brgm.docker01.ilt-dmz.iosb.fraunhofer.de/servlet/is/101/>

3.4.6 French Office for Biodiversity (OFB) – INSIDE - environmental information systems research centre (FR)

The French Office for Biodiversity is a very new institution, founded on 1 January 2020 to protect and restore biodiversity in metropolitan France and overseas territories. It merges the French Agency for Biodiversity with the French national agency for wildlife. The public institution is under the tutelage of the French Environment and Agriculture ministers. As a result it is in charge of three main French Information Systems: the Water Information System, the Marine Information System, the Biodiversity Information System

The Agency has 5 complementary missions:

- making available knowledge of, and research and expertise on species, and their uses
- environmental and wildlife health policing
- support for public policies
- assisting protected area managers
- supporting stakeholders and mobilising civil society

Jointly with BRGM, OFB set up the environmental information systems research centre called 'pole INSIDE'. It aims to push standards, practices and an identified reference open-source implementations up to a maturity level that enables them to be deployed in production by the three Information Systems driven by OFB as well as its partners (BRGM, IFREMER, OFB, MNHN,...).

Its activity and roadmap can be openly accessed through its GitHub (<https://github.com/INSIDE-information-systems/Roadmap/projects/2>).

"INSIDE" teams members will support the provision of OFB data to the API4INSPIRE project.

For the provision of OGC API - Features, we will start with preconfigured GeoServer Black Boxes in Docker, whereby there are plans to successively modify the configuration to directly access the native data sources available within OFB.

For the provision of dynamic measurement data via SensorThings API:

- Surface water quantity : a new ST API instance is being set up on top of the national flows (XML Sandre based). It will be fed using ST API REST & MQTT,
- Surface water quality : in 2018-2019 INSIDE research center already worked with DataCove and Fraunhofer IOSB on exposing the national water quality database according to ST API (https://github.com/INSIDE-information-systems/SensorThingsAPI/tree/master/Surface_Water_Quality_Bigdatavolume_Benchmarking). This work will contribute to API4INSPIRE

For the purpose of this work, the following datasets will be made available via the following APIs:

- Surface Water Quality & Quantity [EF & OM]: both OGC API - Features and SensorThings API
- Hydrological Networks [HY]: OGC API - Features
- Water Transport Networks [TN-W]: OGC API - Features; possible candidate for SF-0 encoding

Currently available endpoints:

STA Surface Quantity: <https://sensorthings.brgm-rec.fr/SensorThingsFlood/v1.0>

STA Surface Quality: <https://sensorthings-wq.brgm-rec.fr/FROST-Server/v1.0>

Viewer: <https://wg-brgm.docker01.ilt-dmz.iosb.fraunhofer.de/servlet/is/101/>

3.4.7 Environment Agency Baden-Württemberg – LUBW (DE)

The Landesanstalt für Umwelt, Messungen und Naturschutz Baden-Württemberg (LUBW) or Baden-Württemberg State Institute for the Environment, Survey and Nature Conservation is a central institution of the German federal state of Baden-Württemberg, whose legal responsibilities lie in the fields of environmental conservation, health and safety and consumer protection and thus they support the state authorities at a technical level in nature conservation and radiation protection. Surface Water Quality & Quantity [EF]: SensorThings API

- Hydrological Network [HY]: SensorThings API and OGC API - Features
- Surface Water Quality [EF & OM]: both OGC API - Features and SensorThings API

For the provision of the spatial data pertaining to the hydrological network in Baden-Württemberg we will utilize a preconfigured GeoServer Black Box in Docker, while additionally making this information available via SensorThings API. The dynamic Surface Water Quality & Quantity data will be provided via SensorThings API. As Fraunhofer IOSB is responsible for the administration and management of LUBW's water resources, the APIs from this data provider will be configured and deployed on Fraunhofer IOSB Infrastructure.

The data is available in an internal expert system used to manage everything water-related that the LUBW is responsible for. The system consists of a database and a fat-client used to access this database. To export the relevant data from this expert system into a server implementing the SensorThings API, the fat-client has been extended with export functionality. This export function first checks if the required metadata entities (Things, ObservedProperties, etc.) exist, and creates them if they are missing, or updates them if they are out of date. Then the exporter checks which Observations need to be exported or updated.

Currently available endpoints:

STA: <https://lubw-frost.docker01.ilt-dmz.iosb.fraunhofer.de/v1.1>

Viewer: <https://wg-brgm.docker01.ilt-dmz.iosb.fraunhofer.de/servlet/is/114/>

3.4.8 Summary Overview

The following table provides a summary overview of all data providers with available datasets.

| | Features | Observations | INSPIRE Themes |
|-------------|--------------------------|--------------------------|------------------|
| ZAMG | Not public WFS2 endpoint | Not public WFS2 endpoint | AC/MF |
| ACG | Public WFS2 endpoint | NA | TN-A |
| UBA | Public WFS2 endpoint | Public SOS endpoint | EF & OM |
| EEA | Public CSV access | Public CSV access | EF & OM |
| CH | Public WFS2 endpoints | Public STA endpoint | TN-R, Smart City |

| | | | |
|-------------|-------------------------|----------------------------|--|
| BRGM | Public WFS2 endpoints | Public SOS & STA endpoints | EF, GE & or OGC:GroundWaterML2, NZ or AM |
| OFB | Internal DB, non public | Internal DB, non public | EF & OM, HY, TN-W |
| LUBW | Internal DB, non public | Internal DB, non public | EF, HY |

Table 5: Overview Data Providers and Datasets

3.5 Conclusions

In the sections above, we have described the methodologies we have defined for the deployment and configuration of both OGC API - Features as well as SensorThings API on a wide variety of stakeholder infrastructure. These deployment plans pertain to both existing software systems as well as to new developments being triggered by the API4INSPIRE Project such as the OGC API Simple currently being developed. In addition, both the GeoServer system for the provision of OGC API - Features as well as the FROST SensorThings API Server are being extended as additional requirements emerge.

Pertaining to the SensorThings API we have made rapid progress in endpoint deployment, with many of the planned endpoints already operational. This was in part due to preexisting work with our data providers, but also due to the power of standardized data models as illustrated in our ad-hoc air quality activity. In parallel to the work on API4INSPIRE, we will also monitor related activities pertaining to STA such as the newly emerged COVID-STA available at <http://193.196.138.56:8080/STACOVID/>. One major advantage of the FROST SensorThings API implementation is the utilization of JSON-Blobs for the storage of all additional properties. Where traditional servers and RDBMS require the creation of a dedicated database schema and subsequent configuration of the server to this structure, FROST allows full access to properties stored in the JSON-Blob format without additional configuration.

Provision of OGC API - Features is more challenging as this standard is much fresher, and software still in the process of development. Necessary extensions to GeoServer for simple provision of OGC API - Features have been agreed upon as described in the sections above, and will be implemented within the next weeks and months. While this development work progresses, we will continue with the configuration of App Schema based on existing GML Schemas for provision of complex data sources. Once the new Smart App-Schema data store has been implemented, we will commence testing this new development for simpler provision; some selected endpoints will be reimplemented with this technology in order to allow for comparison of effort between the two alternative configuration approaches. In addition, our newly developed OGC API - Simple will be deployed on several endpoints to allow us to explore the potential of SF-0 encoding for provision of at least simplified datasets.

3.6 Next Steps

A challenge in the next weeks and months will be ensuring enough end user engagement as more and more planned face-to-face meetings and events are being postponed or cancelled. This is one of the reasons we've widened the scope of provided data with activities such as the Ad-hoc Air-Quality API or the COVID-STA in order to best leverage current events and engage potential communities. In addition, we are currently exploring virtualization methods, providing alternatives to physical meetings in the form of Webinars and other virtual events and challenges. This work will be continued in the next weeks and months in order to assure a good outcome to API4INSPIRE regardless.

4 Technical documentation, source code and working prototype Information system and documentation

4.1 Introduction

This section reports on the documentation and source code created in the scope of the API4INSPIRE project. The documentation is divided into technical documentation, that describes how to use and deploy the APIs, and prototype documentation, that describes the prototypes that are deployed.

All documentation is hosted on the API4INSPIRE documentation website, that can be found at: <https://datacoveeu.github.io/API4INSPIRE/>

The documentation site will be further updated as the project continues.

4.2 Technical Documentation

4.2.1 OGC API – Features

The documentation for OGC API – Features describes the API itself, how to deploy a service using an existing GeoServer installation, and how to deploy a pre-configured “black-box” GeoServer.

— [OGC API - Features Overview](#)

— [GeoServer](#)

- [GeoServer App Schema Mapping](#)
- [Mapping - Example Overview](#)
- [Mapping File Base](#)
- [Basic Feature Mapping](#)
- [Feature Chaining Mapping](#)
- [Identifiers](#)

— [Preconfigured GeoServer "Black Boxes"](#)

- [GeoServer Black Box Configuration for HY-N and TN-W](#)
- [GeoServer Black Box Configuration for SU-V](#)
- [GeoServer Black Box Configuration for AM](#)
- [GeoServer Black Box Configuration for NZ](#)

4.2.2 SensorThings API

The SensorThings API documentation describes the API, with a list of example queries, and links to implementations of clients and servers.

— [OGC SensorThings API](#)

— [The Data Model](#)

— [Basic Requests](#)

— [Tailoring Responses](#)

— [Filtering Entities](#)

— [Expanding Entities](#)

— [Example Queries](#)

— [Implementations](#)

4.3 Working Prototype Documentation

The information about the deployed prototypes can be found under the header “Data Nests”.

— Data Nests

- [Airy Austria](#)
- [Urban Data Platform Hamburg](#)
- [Franco-Germanic Flow](#)
- [Ad-Hoc Sources](#)

4.4 Source Code

4.4.1 FROST-Server

Any code and documentation developed in the project, that is not project specific, will flow back into the development version of FROST (hosted on GitHub), to be included into the next stable release.

Extensions that are (partly) developed in the context of the API4INSPIRE project include:

— GeoJSON result format:

- <https://fraunhoferiosb.github.io/FROST-Server/extensions/GeoJSON-ResultFormat.html>

— Custom Entity Linking extension:

- <https://fraunhoferiosb.github.io/FROST-Server/extensions/EntityLinking.html>

4.4.2 SensorThings API Tools

Several implementation-independent tools have been created that are useful when deploying a SensorThings API instance:

— Tool for Importing GeoJSON into SensorThings:

- <https://github.com/hylkevds/FROST-GeoJsonImporter>

— Tool for integrating SensorThings data into web map tools (OpenLayers & Leaflet)

- <https://github.com/DataCoveEU/STAM>

— SensorThings Processor, for generating aggregate values from sensor data:

- <https://github.com/FraunhoferIOSB/SensorThingsProcessor>

— SensorThings Importer, for importing CSV and other data:

- <https://github.com/FraunhoferIOSB/SensorThingsImporter>

4.4.3 GeoServer

GeoServer deployment(s) will strive to reuse the existing App-Schema configuration, or at least to use it as a starting point, to publish the necessary features through OGC API – Features and to encode the features in the selected formats, e.g. GeoJSON. A guide for upgrading existing systems will be provided, describing in which situations the existing configurations, mainly the App-Schema mappings, can be used as is or which modifications will be needed.

All the developed code and documentation, that is not specific to this project, will be contributed back to the GeoServer project (hosted on GitHub) following the contributions guidelines. In practice, this means that all contributions will be bound to community acceptance and review, backwards compatibility with the existing functionalities will need to be maintained. The target release for the implemented improvements and fixes is GeoServer 2.18.0, which is not officially planned yet but should happen around September 2020. Possible backports will need to be evaluated case by case.

4.4.4 OGC API Simple

OGC API Simple is a green-fields implementation of the OGC API – Features. It can be used on top of an existing SQLite or PostGIS database:

<https://github.com/DataCoveEU/API4INSPIRE/tree/master/OGCAPISimple>

5 API Evaluations

This section summarises the lessons learnt of the executed project tasks evaluating both OGC API - Features (OAPIF) and OGC SensorThings API (STA) and provides recommendations to Member States authorities on how they can best take advantage of these APIs and integrate these within their data infrastructure. Based on the identification of relevant stakeholders and their requirements and approaches to data provision and use, several Stakeholder Perspectives have been derived which guided us during the evaluation process. This prepared for a close interaction with the stakeholders during the project. As a main source we could lean on the experience of the domain experts involved within the API4INSPIRE Project, both the project team as well as data provider staff, which resulted in Heuristic Expert Evaluations.

As face-to-face Hackathons and similar events could not be executed due to the on-going COVID crisis alternatives had to be considered. To gather additional information, questionnaires that had been developed were transformed to online survey format issued to interested domain experts in webinars. Furthermore, interaction partners have been selected and more complex evaluation criteria discussed with them in the form of a structured Interviews. Finally, project partners provided their insights as heuristic expert opinions.

As a further compensation for the reduced user interaction due to COVID restrictions, we expanded the foreseen data sources being provided to include various COVID relevant datasets that are openly available but not currently available in a standardized format. These datasets include real-time air quality data across Europe, daily COVID case data and European Demography data for context to the COVID case numbers, and generated interest from external data users for these API endpoints, providing us with additional external feedback.

Additionally, evaluation dimensions that represent potential action areas were defined. These degrees of freedom have been identified as threefold. The API Specification to identify potential modifications to be communicated back to the underlying standards bodies; the Data Model to identify potential modifications to the underlying data model to be communicated back to the underlying standards bodies; as well as what aspects of the Supporting Ecosystem could be enhanced to better support uptake and use of the API.

In the sections below, we present the outcomes of our evaluations of both OAPIF and STA against each of the Stakeholder Perspectives described above. Please note that due to both logistical constraints ensuing from the Covid-19 pandemic as well as the not yet fully developed nature of OAPIF, it was not possible to perform an evaluation of its usage. Thus, this aspect is missing from the sections below. However, we are confident that the insights obtained from the other Stakeholder Perspectives together with our experience with OGC APIs as well as use of GeoJSON, we can still paint a full picture of the potential of this API.

5.1 Evaluation Outcomes Development OGC API - Features

5.1.1 General Information on API Experience

In the following sections, we will describe the outcome of our evaluation process pertaining to development of an OGC API - Features compliant server, whereby we applied the Evaluation Methods to the Evaluation Criteria taking under consideration all Stakeholder Perspectives. The figure below provides a first overview as to the evaluation outcomes



Stakeholder Perspective: Development

Figure 14: OAPIF Development Perspective Evaluation Overview

The evaluation outcome reflects the emerging nature of OAPIF. While core functionality has been defined, essential parts of this standard such as Filtering and the “Common Query Language” as well as transactions

are still work in progress. The same applies to “OGC API - Common”, that will provide the general underpinning for all OGC APIs, as shown in the roadmap³² excerpt in the following figure:

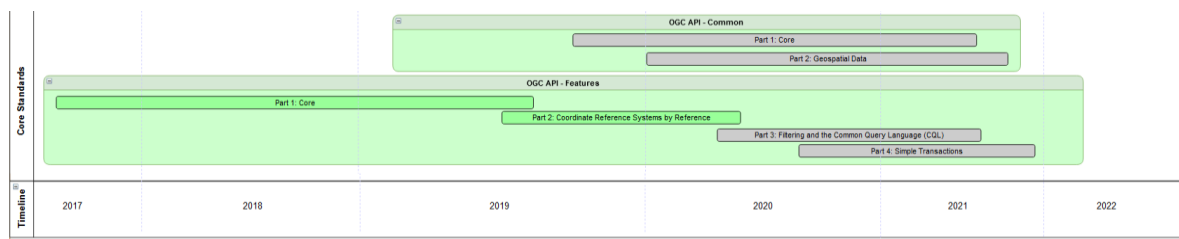


Figure 15: OGC API Development Roadmap

The most commonly encountered issue across all evaluation levels pertains to the lack of examples for all aspects of API development. While some examples of both API code as well as encoded data exist, most are the result of initial prototypes, and have often not been validated and are usually poorly documented. These examples are difficult to find, as there is no central point providing structured information.

A serious deficit of OAPIF at this time pertains to query and analytics functionality. This is at least in part due to the fact that this part of the multipart standard has yet to be finalized. Not clear at present is if this will support queries across multiple linked objects such as supported by STA, or if it will only allow queries pertaining to individual feature types, thus reducing its effectiveness when providing linked data resources.

On a positive note, none of the issues identified had to do with the nature of how the API itself has been structured. Thus, we are confident that once the remaining standards parts relevant for the implementation and use of OAPIF has been finalized, this standard will be well suited for the provision of spatial data.

Detailed information on this evaluation can be found in the accompanying Excel sheet DevelopmentOAF_All.xlsx

5.1.2 Level 1: All find

This level corresponds to very basic non-standardized data provision, data is available together with simple examples. This level focuses on in-house use of APIs. Requires minimal effort in provision but is difficult to impossible to use.

All find



The emerging nature of the OGC API - Features standard becomes clearly visible at this primal evaluation level. The basic criterion for this level has been met, as the API was designed by OGC to serve a diverse set of spatial data provision requirements, based on various use cases provided via OGC and developed through participants at hackathons and events.

Issues have been identified pertaining to the single entry point providing all required documentation and links to standards, as well as the documentation itself. At present, only the core functionality of OGC API - Features has been defined; thus only draft standards texts are available for much of the required functionality, and tutorials are yet to be created.

The lack of sample requests was noted as a serious issue at this level. While some examples are emerging from the various prototypes being created, there is no definitive source, and the examples may also contain errors. The same point was made for data, whereby we could partially ameliorate this issue in the course of the API4INSPIRE project, creating our own API endpoints to serve as examples for further work.

The final issue at this level pertains to discoverability requirements. The way the API is currently designed, it mostly relies on a complementary external metadata catalogue for the provision of administrative metadata concepts. The HTML representation allows for automatic discovery via crawlers. An OpenAPI definition is also available, documenting the details of each resource and which HTTP method can be used on it.

³² <https://ogcapi.ogc.org/apiroadmap.html>

5.1.3 Level 2: All use

This level corresponds to basic standardized data provision. Data is available in a standardized format, basic standards based access functionality is provided. This level focuses on providing data to external users. Medium effort in provision, use is possible, but not optimal.

The evaluation of the Level 2 criteria provided a more positive picture, with only minor issues being identified. The evaluation results were purely positive pertaining to the general criteria as the API was designed in a professional manner by a respected standardization organization, complying with Open API specifications. Both JSON and XML encoding are standard encodings that will allow for widespread uptake of this API.

Issues were identified pertaining to the provision of data license information as well as terms of use. OAPIF allows linking to a data license using a “license” relation, but support for this is currently not available in the implementation, thus this functionality must be covered within the provided data or external metadata catalogue.

“OGC API – Features” allows linking to metadata sheets using the “describedBy” relation; the current implementation being evaluated does not yet support this. The above relation can be used for multiple purposes, including providing a schema for the data (supported). A single, clear meaning of the link to metadata is currently not available in the specification.



5.1.4 Level 3: All trust

This level corresponds to mature standardized data provision. Encompassing data is available in complex standardized format, powerful and mature standards based access functionality is provided. Fairly large effort in provision, use is well supported.

Again at this level, the basic criteria pertaining to standardization of the API and its development by a wide community lead to positive evaluations. The API itself can serve data of any level of complexity, dependent on the utilized data models. In particular, “OGC API – Features” poses no limitation on the data being returned, the results can be simple, complex, schema oriented or schemaless. Background support is available as this API has been developed by OGC, being implemented by various vendors; OGC members include a mix of small and large organizations.

Issues have been identified pertaining to query and analytics due to the emerging nature of the API. The specification for “OGC API - Features - Part 3: Filtering and the Common Query Language (CQL)” is still in the process of being drafted, planned to be finalized 2021. At present, only very simple query functionality is available. Thus one cannot determine the implementability of this functionality until Part 3 is finalized, but based on the current drafts should be simple to map to SQL Queries on the underlying data source. While there are no frameworks foreseen, query functionality needs to be implemented on underlying data query modalities.

Error handling is currently similarly undefined, with only basic HTTP behaviour specified. At present only 200: success, 404: not found and 400: bad request. While formally correct, this level of error handling is not really useful. A similar deficit pertains to performance and caching functionality, with nothing foreseen within the API specification. Having the API provide validity for data objects would enable external caching by web servers. Scalability of the systems is not a problem as data requests are stateless, allowing for multiple servers in parallel with load balancing.

Finally, data validation remains an issue as not foreseen at present. Final decisions on JSON encoding as well as use of JSON Schema still in progress.

5.1.5 Level 4: All involved

This level pertains more to the maturity of the wider ecosystem built up around the standardized specifications. The aspects described go beyond the control of the individual data providers and thus must be supported through diverse stakeholder communities. Achieving this level facilitates both the provision and usage perspectives.



Level 4 is the second level where serious issues have been identified towards the use of OGC API - Features as it stands today. At present, there is no support available for the development of this API beyond what is available pertaining to OpenAPI itself. Further deficits have been identified pertaining to the lack of a playground in which developers can experiment with an existing system, as well as the provision of linked documentation integrated with valid examples. This is related to the lack of documentation and examples already identified in previous levels, steps should be taken to provide such support before this API is utilized at a wider level.



Related to this issue is the lack of code examples. While code from existing prototypes is available, the implementations are not meant to be didactical, they are production oriented and often complex. Transfer to another programming language is possible, but not easy.

Positive feedback was provided on all criteria pertaining to the community nature of this development. As the API is still in the process of development, the community is currently very active. Most OGC Standards are developed via GitHub, as this allows for interaction, also for non-OGC members. A Gitter channel is also available. In addition, periodic code sprints are organized to foster the development of standards and verify they are possible and practical to implement. However, most of the participants have been involved long term in standardization efforts, it's required to have a suitable set of skills and knowledge to productively participate in it.

5.1.6 Level 5: All develop

This level describes a well developed and mature ecosystem built up around the standardized specifications. The aspects described go far beyond the control of the individual data providers and thus must be supported through diverse stakeholder communities. Achieving this level facilitates both the provision and usage perspectives.

All develop

Level 5 again provided a more positive picture, with only minor issues identified. Due to the OGC infrastructure, most community requirements are covered, issues can be discussed and solutions proposed via the relevant GitHub repositories. While OGC Licenses are not aligned with CC-BY, the OGC Licences are well known. An example can be found at the very beginning of the OGC API - Features specification: <http://docs.openeospatial.org/is/17-069r3/17-069r3.html>.



The API has been designed in such a manner as to allow for the provision of JSON-LD, and also generally makes strong use of links to integrate with other endpoints. A test suite for the core OGC API is available on GitHub (<https://github.com/openeospatial/ets-ogcapi-features10>). The OGC CITE engine (<https://cite.openeospatial.org/teamengine/>) can be used to test an implementation that is reachable from the internet. The tests can also be run offline.

Some issues were identified pertaining to the visibility of the API code. While various developments are open source and make their code available, as much of this code has been developed through rapid prototyping and hackathons, it is now always well structured or easy to interpret. A roadmap is available for the major building blocks, for more fine grained information one must go through the details of the issues on GitHub. To a certain extent yes, but what actually gets developed also depends on community requirements and SWG members availability, so a specific timeline cannot be determined.

5.2 Evaluation Outcomes Deployment OGC API - Features

5.2.1 General Information on API Experience

In the following sections, we will describe the outcome of our evaluation process pertaining to the deployment of OAPIF, whereby we applied the Evaluation Methods to the Evaluation Criteria taking under consideration all Stakeholder Perspectives. The figure below provides a first overview as to the evaluation outcomes.



Stakeholder Perspective: Deployment

Figure 16: OAPIF Deployment Perspective Evaluation Overview

The main issues identified pertaining to the deployment of OAPIF are due to the lack of documentation and examples available for this emerging API. While this is in part due to the emerging nature of this API specification, it is also related to a wider issue often encountered in the usage of OGC standards. The standards texts are concise but terse, listing explicit requirements, but not really connecting the dots between these requirements to show actual usage. Simple tutorials complementing these standards would be most valuable. In addition, there is at present a lack of examples. Based on the survey response, examples, tutorials and a playground would be the most essential additions to the existing resources.

While the API foresees links to external resources, the only explicit linking to additional metainformation pertains to licenses. Terms of use and additional metadata must be linked from within the data being provided as this is not supported by the API

Query and filtering functionality are currently not defined, as this standard part is not yet finalized. This functionality should be revisited once specified, to assure that all requirements are met. It will be interesting if this upcoming standard will also allow for queries pertaining to associated features, or if filtering will be limited to the content of individual feature types.

Finally, error handling is poorly defined within this API, with only the most basic HTTP error codes being provided. There is no additional support for identification of data errors, or issues in the underlying data models.

Detailed information on this evaluation can be found in the accompanying Excel sheet `DeploymentOAF_All.xlsx`

5.2.2 Level 1: All find

This level corresponds to very basic non-standardized data provision, data is available together with simple examples. This level focuses on in-house use of APIs. Requires minimal effort in provision but is difficult to impossible to use.

The issues encountered in the evaluation of Level 1 pertaining to deployment revolved around the lack of maturity of the documentation and examples. As only the core functionality of this API has been specified to date while the complementary parts continue to emerge, the available documentation and examples are spotty at best, while in most cases, the technicians responsible for deployment must rely on the standards text itself where available, as well as trying to understand the existing drafts for the upcoming parts.

The lack of documentation and examples was nicely expressed during the survey accompanying our ELISE Webinar on OAF, with “Examples” and “more Examples” dominating the word cloud illustrating the survey response to the question of “What further documentation would you require?”

What further documentation would you require?



Figure 17: User Feedback - Examples Required

5.2.3 Level 2: All use

This level corresponds to basic standardized data provision. Data is available in a standardized format, basic standards based access functionality is provided. This level focuses on providing data to external users. Medium effort in provision, use is possible, but not optimal.

The issues identified pertaining to the Level 2 criteria on API deployment were similar to those identified pertaining to development. While the API itself is well standardized, and deemed quite suitable for purpose, the lack of mechanisms for the provision of additional metadata on the resources being provided was deemed troublesome. The way the API has been defined, additional metadata must be provided by an external metadata catalogue, with no mechanisms foreseen for direct provision of additional metadata, data licensing information or information on terms of use directly within the API.



Authentication is also not covered within the standard, devolving these concerns to external levels such as web servers. While these mechanisms are well suited to basic authentication, they break down when fine grained rights management directly dependent on individual data concepts is required. As diverse authentication mechanisms are required by many data providers, additional work in this area, especially pertaining to rights on the object level, should be explored.

5.2.4 Level 3: All trust

This level corresponds to mature standardized data provision. Encompassing data is available in complex standardized format, powerful and mature standards based access functionality is provided. Fairly large effort in provision, use is well supported.

All trust



The two positively evaluated criteria at Level 3 pertained to the standardized nature of the API as well as its support for the provision of complex data models. Some issues were identified with the underlying INSPIRE data models, that are not always seen as fit for purpose. The available background support was also seen as very positive, with many organizations involved in the development of this standard and supporting software.

Similar issues pertaining to query and analytics functionality of the API were reported pertaining to deployment as already mentioned under development. Once this standard part has been finalized, this question should be revisited to assure that all filter and query requirements are fully supported. A concern at present pertains to the possibility to query across associated features, as this functionality was not foreseen under WFS2 and no indication of such support has been seen in the draft versions. This is in part due to the fact that data models are not included in the API specification.

Error handling was seen as a serious deficit, as only the most basic error responses have been defined. While these basic HTTP errors do indicate that an error occurred, they mostly leave it to the user to determine the exact nature of the error, making work with the API an often painful process. At present it is unclear if this will be rectified in further parts of the OGC API Standards family. A similar issue was identified pertaining to data validation, currently only possible on XML outputs of the API.

Caching mechanisms are also not covered by the standard, but due to the stateless nature of the API, this functionality can be devolved to external HTTP Servers.

5.2.5 Level 4: All involved

This level pertains more to the maturity of the wider ecosystem built up around the standardized specifications. The aspects described go beyond the control of the individual data providers and thus must be supported through diverse stakeholder communities. Achieving this level facilitates both the provision and usage perspectives.

As to be expected, the maturity of the wider ecosystem is not yet given as the standards themselves are still emerging, the community is mostly focused on technical aspects, thus at present little support on pragmatic deployment issues. This issue has been exacerbated through the shut-down of the INSPIRE Forum, thus devolving such issues to generic portals beyond the INSPIRE Scope.



While there are many emerging prototypes, it would be nice to have an openly available reference implementation as a definitive example. The API4INSPIRE project has made some inroads in this area, creating both documentation as well as demo endpoints. However, such work requires good abstraction capabilities, as examples are only provided for a few selected themes.

The API provides basic (Swagger) info in the /api page. Otherwise one must know where to search for additional information as many of the required standards documents not yet finalized.

Issues pertaining to the data model are related with deeper issues pertaining to INSPIRE data models, thus not responsibility of the API. Such issues must be first resolved within the INSPIRE Conceptual models. The API itself can serve any data model, the identified issues must be resolved at a different level.

5.2.6 Level 5: All develop

This level describes a well developed and mature ecosystem built up around the standardized specifications. The aspects described go far beyond the control of the individual data providers and thus must be supported through diverse stakeholder communities. Achieving this level facilitates both the provision and usage perspectives.

Interesting is that at Level 5, the API conforms very well to the evaluation criteria defined. This is mostly due to the framework provided by the OGC for standards development, covering most aspects pertaining to this evaluation level. The development and extension of the API standard is done in a collaborative manner, with good tools in place for interaction and reporting of issues. OGC provides a development roadmap that is regularly updated to reflect actual progress and new aspects being standardized.



In addition, this API is well suited for the requirements of Linked Data. The API itself relies strongly on links, and this is further reflected in the support for JSON-LD.

The only issue encountered in the Level 5 criteria pertained to the availability of a test framework. While CITE tests are available, we do not see these as suited for deployers, as a great deal of technical expertise is required. Thus, API deployers must rely on the developers to do this level of validation for them.

5.3 Evaluation Outcomes Development SensorThings API (STA)

5.3.1 General Information on API Experience

In the following sections, we will describe the outcome of our evaluation process pertaining to development of a STA compliant server, whereby we applied the Evaluation Methods to the Evaluation Criteria taking under

consideration all Stakeholder Perspectives. The figure below provides a first overview as to the evaluation outcomes.



Stakeholder Perspective: Development

Figure 18: STA Development Perspective Evaluation Overview

The evaluation of STA pertaining to development provides a quite positive view of this standard in the context of INSPIRE. Some deficits have been identified pertaining to provision of data license information and terms of use as these concepts are not covered by the API standard. Authentication mechanisms are devolved to external HTTP mechanisms, not allowing for fine-grained access rights on the data level.

Error handling could definitely be developed further, as at present only the most basic HTTP errors are covered by the standard itself. Thus, all detailed error handling must be foreseen by the developers. A similar issue pertains to data validation, as no JSON schema files are foreseen at present.

While STA is well suited for linked data approaches due to its basic design, integration of JSON-LD is currently under investigation, but no issues have been encountered to date prohibiting its use in the STA context.

Detailed information on this evaluation can be found in the accompanying Excel sheet DevelopmentSTA_All.xlsx

5.3.2 Level 1: All find

This level corresponds to very basic non-standardized data provision, data is available together with simple examples. This level focuses on in-house use of APIs. Requires minimal effort in provision but is difficult to impossible to use.

Pertaining to Level 1, the only issues identified pertained to the single entry point and discoverability of the API. While there is a great deal of documentation available for STA, at present information is still a bit divided and not linked directly from the API endpoints. However, the number of examples and tutorials available make up for this deficit.

A further issue pertains to discoverability of STA endpoints. As no HTML representations of the data being provided are foreseen, it is difficult for standard web search engines to index the content of the STA endpoint. An external catalogue service is required for easy discoverability of STA endpoints.

5.3.3 Level 2: All use

This level corresponds to basic standardized data provision. Data is available in a standardized format, basic standards based access functionality is provided. This level focuses on providing data to external users. Medium effort in provision, use is possible, but not optimal.

Again, STA shows the power of a more mature well standardized OGC API. The issues encountered at this level pertained mostly to mechanisms for the provision of data license information as well as terms of use, currently not foreseen by the API. If such concepts are to be provided, they must be contained within the data payload of the objects being provided. At present, no extensions for this information have been foreseen in the underlying data model, so developers cannot add additional support.

A further issue identified pertains to authentication, where STA relies solely on external HTTP mechanisms. While these do provide support for most authentication requirements, they do not support

fine-grained authentication and access on the data level. Extensions to the STA API would be required for such functionality if required.

5.3.4 Level 3: All trust

This level corresponds to mature standardized data provision. Encompassing data is available in complex standardized format, powerful and mature standards based access functionality is provided. Fairly large effort in provision, use is well supported.

All trust



Query and analytics are one of the great strengths of STA, with the creation of very powerful and complex queries being integral to the underlying OData standard. One very valuable functionality in this context is the ability to create queries across multiple associated features that compose the integrated data model of STA. This allows for the provision of a highly normalized data model while at the same time ensuring that all query and filter requirements are fulfilled.

Some deficits were noted pertaining to error handling, with only the most basic HTTP error codes foreseen by the standard. Server developers are now investigating how to best integrate further error handling support, e.g. providing better error messages for incorrectly formulated queries.

Performance and caching are also not specified within STA, but provision of validity times for basic requests could help caching by external HTTP servers. Horizontal scaling is possible as data requests are stateless.

Data validation is not possible at present, as no JSON Schema has been created for the underlying O&M based STA data model.

5.3.5 Level 4: All involved

This level pertains more to the maturity of the wider ecosystem built up around the standardized specifications. The aspects described go beyond the control of the individual data providers and thus must be supported through diverse stakeholder communities. Achieving this level facilitates both the provision and usage perspectives.

The maturity of the wider STA ecosystem becomes quite visible at Level 4, with the only deficit identified pertaining to the linked documentation criterion. While the documentation is linked to many examples, there is no direct link from the API landing page to these resources.

Apart from the lack of a direct link to the documentation and further resources, this Level is fulfilled very well for STA, with many examples, playgrounds and a vibrant community established around this standard.

All involved



5.3.6 Level 5: All develop

This level describes a well developed and mature ecosystem built up around the standardized specifications. The aspects described go far beyond the control of the individual data providers and thus must be supported through diverse stakeholder communities. Achieving this level facilitates both the provision and usage perspectives.

All develop



As expected, only a few minor issues were identified at this level. Pertaining to the development roadmap, while it is available for the major building blocks, for more fine grained information one must go through the issues on GitHub. The development priorities usually depend on community requirements, thus if major extensions are required, this will also require resources for this work.

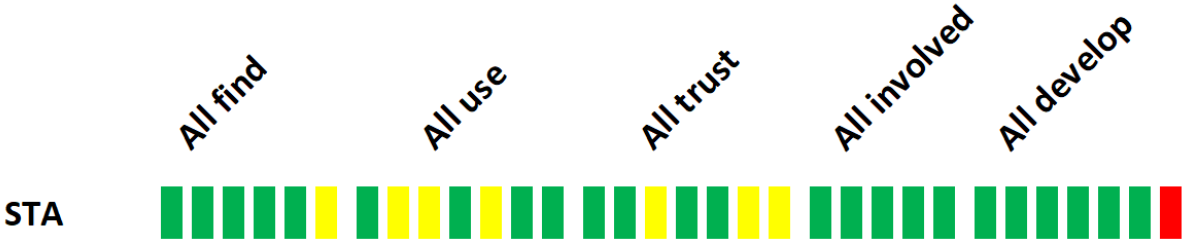
Pertaining to linked data requirements, the basic structure of STA is well suited, making strong usage of links contained within the underlying data model. Investigations of the usage of JSON-LD are ongoing, but need to be finalized before extending the standard to this serialization language.

A test suite for the core SensorThings API is available on GitHub (<https://github.com/opengeospatial/ets-sta10>). The OGC CITE engine (<https://cite.opengeospatial.org/teamengine/>) can be used to test an implementation that is reachable from the internet. The tests can also be run offline.

5.4 Evaluation Outcomes Deployment STA

5.4.1 General Information on API Experience

In the following sections, we will describe the outcome of our evaluation process pertaining to the deployment of STA, whereby we applied the Evaluation Methods to the Evaluation Criteria taking under consideration all Stakeholder Perspectives. The figure below provides a first overview as to the evaluation outcomes.



Stakeholder Perspective: Deployment

Figure 19: STA Deployment Perspective Evaluation Overview

Based on our evaluation criteria pertaining to API deployment, STA is a quite mature API, fulfilling most of the requirements posed.

Issues identified pertained mostly to the provision of additional metadata, pertaining both to discovery of the API endpoints as well as to the standardized provision of information on data licenses and terms of use. Authentication mechanisms are also not foreseen within the API specification.

Further issues noted pertain to tools supporting data ingestion to a STA system. While the underlying data model is quite simple at core, its flexibility requires in depth thought before structuring the required data. Validation of the data itself as well as the underlying API framework is not foreseen at present.

Detailed information on this evaluation can be found in the accompanying Excel sheet DevelopmentSTA_All.xlsx

5.4.2 Level 1: All find

This level corresponds to very basic non-standardized data provision, data is available together with simple examples. This level focuses on in-house use of APIs. Requires minimal effort in provision but is difficult to impossible to use.

All find

Pertaining to the requirements laid down by evaluation Level 1 towards STA, the only issue identified pertained to discoverability, as there are no integrated mechanisms foreseen for provision. An external metadata catalogue must be utilized in order to assure the discoverability of the API endpoint and the data resources made available.

All general requirements pertaining to documentation, example requests and example data were well supported by existing resources, and further complemented by documentation created in the course of the API4INSPIRE project.

5.4.3 Level 2: All use

This level corresponds to basic standardized data provision. Data is available in a standardized format, basic standards based access functionality is provided. This level focuses on providing data to external users. Medium effort in provision, use is possible, but not optimal.

Pertaining to Level 2 of the evaluation criteria, issues were identified pertaining to the provision of licensing information as well as information on the terms of use pertaining to the data being provided. There are at

present no mechanisms for this, all requirements must be fulfilled through provision of this information within the extension points foreseen within the 1.1 version of the underlying data model.


Authentication is also not foreseen in the STA specification, thus one must rely on existing mechanisms provided by HTTP levels. While sufficient for most use cases, these mechanisms do not allow for fine grained access and rights pertaining to individual data objects.



STA does provide a great deal of direct metadata pertaining to the measurement data being provided, as this is integral to the underlying data model, making STA uniquely suited for the provision of dynamic data. This is in line with the general suitability of STA, largely due to its standardized background and the wide community behind this standard.

5.4.4 Level 3: All trust

This level corresponds to mature standardized data provision. Encompassing data is available in complex standardized format, powerful and mature standards based access functionality is provided. Fairly large effort in provision, use is well supported.



The main issue identified pertaining to Level 4 of the evaluation criteria pertains to the integration into existing workflows. At present, tools for mapping existing data sources to the STA data model are still emerging. As the STA data model is very flexible, allowing for various options for the explicit structuring of the data, care must be taken to assure that the structure selected is also well suited to the expected queries on this data source.

Data validation is also not foreseen, as at present no JSON Schemas have been created for the underlying data types that comprise the STA data model. Otherwise STA fulfils all requirements pertaining to this level of API maturity.

5.4.5 Level 4: All involved

This level pertains more to the maturity of the wider ecosystem built up around the standardized specifications. The aspects described go beyond the control of the individual data providers and thus must be supported through diverse stakeholder communities. Achieving this level facilitates both the provision and usage perspectives.


The fact that STA is a mature and well adopted standard is reflected in the evaluation outcomes pertaining to the criteria laid down for Level 4. A community has grown around this standard, with multiple implementations available utilizing various technologies.

Extensive documentation is available, together with various sites offering playground functionality linked to this documentation. Additional requirements identified within the INSPIRE process have been added to the 1.1 version of the standard, illustrating how the API evolves through community input.



5.4.6 Level 5: All develop

This level describes a well developed and mature ecosystem built up around the standardized specifications. The aspects described go far beyond the control of the individual data providers and thus must be supported through diverse stakeholder communities. Achieving this level facilitates both the provision and usage perspectives.



Again the maturity of the STA standard is well reflected in the evaluation outcomes pertaining to the criteria posed by Level 5 of our evaluation schema. As this standard is from OGC, a rich development environment is available pertaining to this standard, including bug tracking functionality.

There are many implementations of this standard to choose from, and all required information is readily available. The only issue identified within the criteria at this level pertained to test frameworks. While CITE tests are available from OGC, these are not easy for deployers to handle.

5.5 Evaluation Outcomes Use STA

5.5.1 General Information on API Experience

In the following sections, we will describe the outcome of our evaluation process pertaining to STA usage, whereby we applied the Evaluation Methods to the Evaluation Criteria taking under consideration all Stakeholder Perspectives. The figure below provides a first overview as to the evaluation outcomes.



Stakeholder Perspective: User

Figure 20: STA User Perspective Evaluation Overview

As already detailed in the sections above pertaining to development and deployment of STA, this standard is quite mature, and has a large and active supporting community. Again, the issues identified pertained mostly to the provision of additional administrative metadata required for discovery and use (data licensing and terms of use information), as this is not foreseen by the API specification, such information must be explicitly provided within the data payload.

Additional issues identified pertained to error handling that is at present poorly specified within the API standard, devolving this task to implementers and custom solutions. This also pertains to data validation and the lack of a test framework. Thus, users must trust that the systems they are accessing have been created correctly.

Detailed information on this evaluation can be found in the accompanying Excel sheet UseSTA_All.xlsx

5.5.2 Level 1: All find

This level corresponds to very basic non-standardized data provision, data is available together with simple examples. This level focuses on in-house use of APIs. Requires minimal effort in provision but is difficult to impossible to use.

All find

As Level 1 of our evaluation criteria pertain to the most primal aspects of APIs, it was not surprising that STA easily fulfilled all requirements pertaining to this level. Interview responses with experienced STA developers painted a highly positive view of this level, as in contrast to the more typical home-spun APIs expected at this level, STA is a professional standard

The questionnaire responses were not quite as positive, with some participants not immediately able to identify the necessary resources. However, the majority of the participants answered positively, thus the assumption is that the negative responses were due to the tight time constraints placed on the Webinar context. This assumption was confirmed by the interview responses.

5.5.3 Level 2: All use

This level corresponds to basic standardized data provision. Data is available in a standardized format, basic standards based access functionality is provided. This level focuses on providing data to external users. Medium effort in provision, use is possible, but not optimal.

Based on the interview responses, STA fulfils most criteria posed to Level 2. Due to the standardized nature of STA, this API has been designed and thoroughly reviewed by international experts. The nature of the underlying STA data model ensures that all relevant metadata concepts pertaining to the observations being exposed are provided. Due to the underlying O&M model, formalized as ISO 19156, STA is highly suited for the provision of dynamic data on spatial features.



Issues were identified pertaining to the provision of information on data licenses as well as terms of use pertaining to the data. While such information can be provided within the data payload of the features being provided by this endpoint, no conventions for this provision are foreseen by the API specification. A related issue pertains to authentication, for which STA relies on external mechanisms provided by the web server; while such an approach covers many cases, it does not allow for fine-grained access mechanisms directly pertaining to the data provided.

The questionnaire responses confirmed the favourable impression of STA as relates to the criteria of Level 2. The vast majority of participants found the API very intuitive and well suited for real-world use cases; the overhead involved in assuring that all relevant information is concisely provided was deemed well worth the ensuing functionality.

5.5.4 Level 3: All trust

This level corresponds to mature standardized data provision. Encompassing data is available in complex standardized format, powerful and mature standards based access functionality is provided. Fairly large effort in provision, use is well supported.

Within Level 3, one deficit identified pertained to data validation, that is currently not foreseen for OGC APIs utilizing JSON encoding; work on the utilization of JSON Schema is progressing, but has not yet been finalized. This pertains less to the API itself, more to current development paradigms that deem validation to be unnecessary

A further issue identified pertained to error responses. While certain error responses are clearly specified in the standard, this does not pertain to all aspects of user interaction with a STA endpoint, often leaving users guessing. For instance, the standard does not describe how a service should respond if a user makes an invalid filter request. Though all implementations return errors that are sufficiently clear for a human user to find the problem, the lack of a standardised error message schema makes it impossible for client software to reliably pass errors on to the user.

This feedback was again reflected in the questionnaire responses. While the questions targeting the degree to which the API fulfils requirements pertaining to API requests and filter functionality, the issues pertaining to error handling in STA were of concern to all Webinar participants.

5.5.5 Level 4: All involved

This level pertains more to the maturity of the wider ecosystem built up around the standardized specifications. The aspects described go beyond the control of the individual data providers and thus must be supported through diverse stakeholder communities. Achieving this level facilitates both the provision and usage perspectives.



In the years since the formal approval of STA as an OGC Standard early 2016, diverse implementations of STA Servers, both proprietary and open source, have emerged and been deployed across a wide range of usage areas. This has in turn engendered a vibrant community, with a wide array of resources available. This was largely reflected in the interview results, with the only deficit identified pertaining to the available documentation and how well this was linked to examples and a playground; this deficit was largely remedied within the API4INSPIRE Project through the creation of a rich documentation interlinked with examples and a playground. While the API landing page provides machine readable documentation, no links are foreseen for further human readable documentation or tutorials from API endpoints.

All involved



This feedback was mirrored in the questionnaire results, where participants reported no problems accessing documentation and playgrounds, but no direct link from the API endpoints to further textual documentation.

5.5.6 Level 5: All develop

This level describes a well developed and mature ecosystem built up around the standardized specifications. The aspects described go far beyond the control of the individual data providers and thus must be supported through diverse stakeholder communities. Achieving this level facilitates both the provision and usage perspectives.

All develop

The maturity of the STA specification is well reflected in the evaluation results pertaining to Level 5 of the usage evaluation criteria. As this standard is from OGC, a rich development environment is available pertaining to this standard, including bug tracking functionality.

The only issue identified pertains to the lack of a test framework to assure that the API itself is performing in accordance with specifications.



6 Conclusions

6.1 Evaluation Overview

Both APIs have recently been endorsed as INSPIRE Good Practices for deployment as INSPIRE download services and are valuable additions to the INSPIRE ecosystem. As they are both developed by OGC, their development process is well monitored by many interested parties and the overall quality of the standards is assured.

The diagram below provides a comparison of all evaluations performed. Note that some fields in the diagram below are grey - this was necessary to assure alignment between evaluations as not all evaluation criteria pertained to all Stakeholder Perspectives.

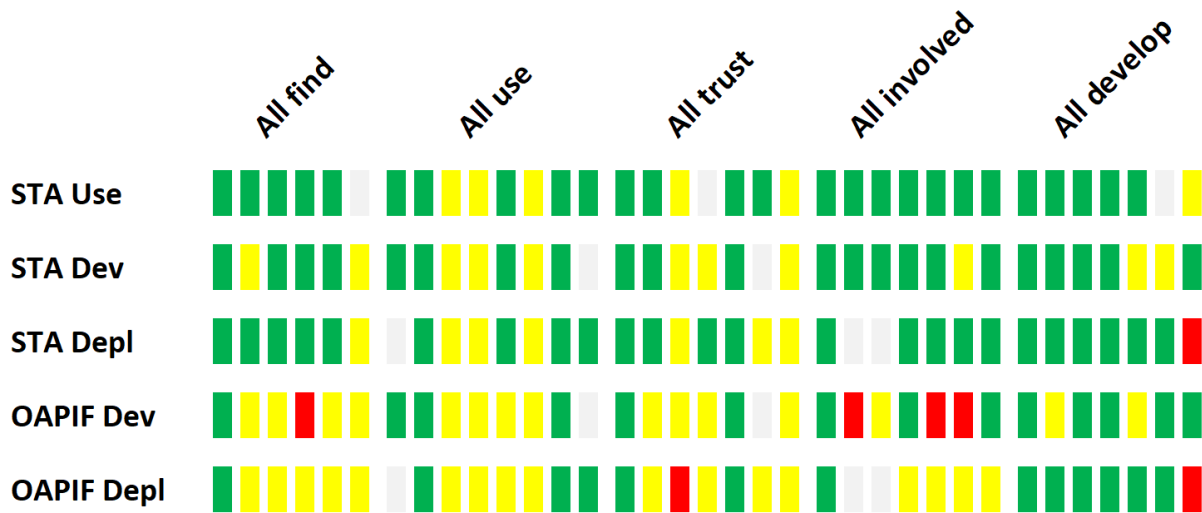


Figure 21: Evaluation Overview

As an alternative view, here we also provide all evaluation results in a numeric form.

| | All find | All use | All trust | All involved | All develop |
|------------|-------------|-----------------|---------------|---------------|---------------|
| STA Use | 1 1 1 1 1 | 1 1 2 2 1 2 1 1 | 1 1 2 1 1 2 | 1 1 1 1 1 1 1 | 1 1 1 1 1 1 2 |
| STA Dev | 1 2 1 1 1 2 | 1 1 2 2 1 2 1 | 1 1 2 2 1 2 | 1 1 1 1 1 2 1 | 1 1 1 1 2 2 1 |
| STA Depl | 1 1 1 1 1 2 | 1 2 2 1 2 1 1 | 1 1 2 1 1 2 2 | 1 1 1 1 1 | 1 1 1 1 1 1 3 |
| OAPIF Dev | 1 2 2 3 2 2 | 1 1 2 2 2 2 1 | 1 2 2 2 1 2 | 1 3 2 1 3 3 1 | 1 2 1 1 2 1 1 |
| OAPIF Depl | 1 2 2 2 2 2 | 1 2 2 2 2 1 1 | 1 2 3 2 1 2 2 | 1 2 2 2 2 | 1 1 1 1 1 1 3 |

Figure 22: Evaluation Overview Numeric (1 Best, 3 Worst)

STA is a mature and proven standard that has been in use for more than five years. STA has been developed based on the OGC SWE Suite of standards, leveraging the decades of experience pertaining to structuring and provision of measurement data as encapsulated within the O&M standard, while extending its scope towards the Internet of Things (IoT) and relevant sensor protocols such as MQTT. Based on the user requirements pertaining to spatially enabled sensor measurements, the API was specified utilizing the ISO/IEC³³ approved OASIS Open Data Protocol (OData), that defines a set of best practices for building and consuming RESTful APIs.

³³ <https://www.oasis-open.org/2017/02/21/iso-iec-jtc-1-approves-oasis-odata-standard-for-open-data-exchange/>

One essential functionality provided by this API approach is enabling queries across multiple associated classes, thus greatly reducing the number of requests required to identify and retrieve the required data; OData libraries can be utilized to support such functionality on existing databases. There are multiple implementations of STA available, both open-source and closed-source, on both server- and client-side.

OAPIF is a young standard created on the basis of Open API, and still in the process of being specified. The core of this standard has been finalized, together with Part 2: Coordinate Reference Systems by Reference, that opens up the scope of spatial data being provided to data encoded in coordinate reference systems beyond WGS84. Based on the current roadmap, additional standards parts pertaining to Filtering and the Common Query Language (CQL, Part 3) and Simple Transactions (Part 4) should be finalized by the end of 2021, rounding out the API functionality. At present, it does not yet completely replace WFS, but looks very promising and is already the simpler alternative for many use cases. Implementation of the basic server functionality is relatively easy, as demonstrated by the OGC-API Simple implementation that was developed in the context of this project. However, as much of the more advanced functionality that is planned for the additional parts of the standard has yet to be specified, it is not possible to comment on the implementation complexity of the full standard, especially pertaining to query/search functionality that goes beyond simple bounding-box or time queries as foreseen in the Core of this standard. To what degree complex queries and filtering across multiple associated features as provided by the OData functionality underlying STA will be supported is not clear at this time. Planned work on integration of JSON Schema will also be relevant.

The utilization of both API will surely profit from recent developments towards increased semantics. While STA relies on the underlying semantics of the O&M data model that has been entrenched within this standard, OAPIF is exploring the utilization of JSON Schema for the provision of explicit information on the semantic structure of the data being exposed. In addition, linked data developments that have led to the creation of JSON-LD, and recently a GeoJSON-LD, are being investigated for inclusion within both APIs, providing users with the best of both worlds: simple data access through the provision of GeoJSON data paired with rich semantics in the -LD context header, allowing data users to better understand the data being provided, and set it into wider context.

6.2 General Issues

In the course of deploying both OAPIF and STA across diverse datasets made available by our data providers, several general issues were exposed. While resolution of these issues was not within the scope of the evaluation, we provide them here as we believe additional investigation would be most valuable.

6.2.1 Cross-Border Alignment

An important general topic that is not new to the API standards under evaluation here, but became visible due to the nature of the datasets being utilized within this project, pertains to cross-border aspects of the data being exposed, how to deal with cross-border identification of features that cross or define these borders. While we could expose data from both sides of the French-German border in a compatible manner, for true interoperability, the following issues must first be solved:

- Matching: while both datasets contain spatial features that depict segments of the Rhine River, there are at present no mechanisms in place for determining that these actually describe the same river. The creation of matching services that access available API endpoints and provide possible matched pairs would be most valuable.
- Identification: at present there is no transnational identification system for spatial objects that fall under the jurisdiction of multiple nationalities. The OGC Interoperability Experiments ELFIE and SELFIE have started work exploring how multiple digital representations of one real-world-object can be connected through provision of a URI for the real-world-object; this could provide guidance for future developments on this point.
- Referencing: the specification of a mechanism for referencing between different digital representations pertaining to the same or related real-world-objects must be further explored. At present OAPIF provides links to alternative encodings, but only pertaining to the media type of the encoding. Integration of profile information with links information could allow for linkage to alternative representations, also those hosted by other data providers.

6.2.2 Linking OAPIF and STA

A new issue that has emerged due to the subtle differences between these two new OGC APIs is how to deal with Features (e.g. Environmental Monitoring Stations, Rivers, Lakes, Aquifers) that are represented in both APIs; how to define and implement links between the Features being provided by these two APIs.

To illustrate this issue, we take an example from our Data Nests, where we've been at work exposing information on both surface- and ground-water bodies, together with information on Environmental Monitoring Facilities that provide data on these Features. For the provision of data on water bodies, the data specifications from the INSPIRE Themes Hydrography and Transport Networks Water have been utilized. In addition, all water monitoring stations pertaining to both surface- and ground-water have been provided in accordance with the INSPIRE Environmental Monitoring Facilities Theme. Both the water bodies as well as the facilities providing measurements on these water bodies have been made available via OAPIF, as this is the logical replacement for the WFS previously utilized for provision of such data. The diagram below illustrates the complex interlinkages between the three INSPIRE Themes utilized in the provision of this data.

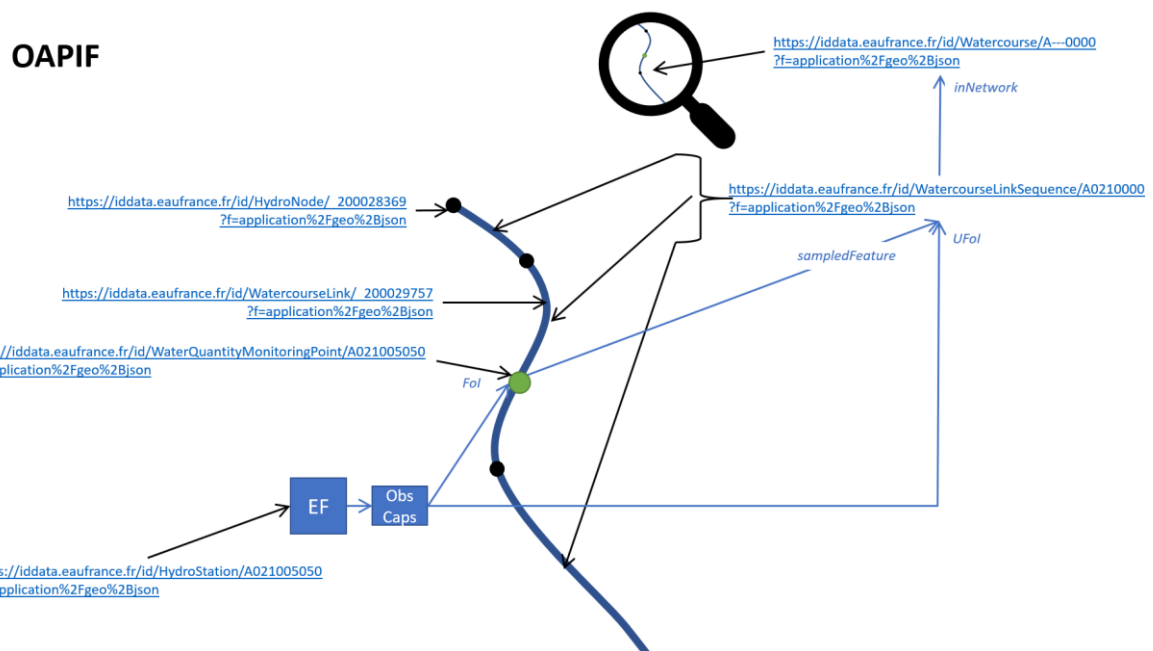


Figure 23: OAPIF River Network and Environmental Monitoring Facility

At the same time, the measurement data together with all relevant measurement meta-information has been exposed via a paired STA, as this API is far better suited for dealing with the complexities of real-time measurement data. In contrast to the OAPIF configuration described above, only minimal amounts of data pertaining to spatial features beyond the Environmental Monitoring Facility and the Feature of Interest are provided. In the configuration created within this project, only the wider Watercourse has been modelled as an additional Thing, whereby the link from the Thing representing the Environmental Monitoring Facility to the Thing representing the Watercourse has been realized utilizing the Custom Entity Linking³⁴ extension to STA. The STA based representation of the water monitoring stations is shown in the following diagram.

³⁴ <https://github.com/INSIDE-information-systems/SensorThingsAPI/blob/master/EntityLinking/Linking.md>

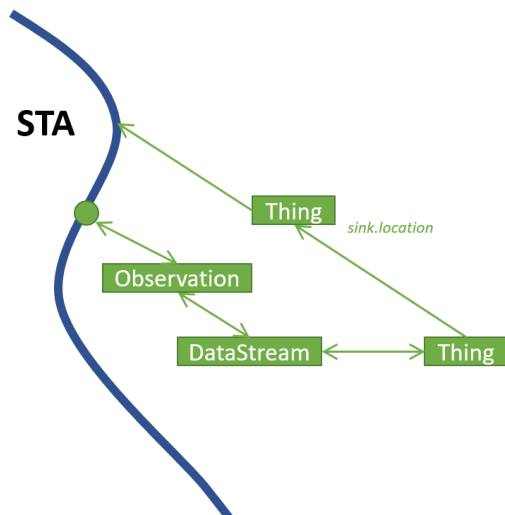


Figure 24: STA River Network and Environmental Monitoring Facility

The challenge lies in bringing these two representations of the same underlying data sources being provided by the two different APIs under investigation here together. In the diagram below, we show the relevant features pertaining to water monitoring stations as provided by both APIs, with the features stemming from OAPIF displayed in blue while the features being provided via STA are displayed in green. The brown arrows indicate equivalent features being provided by both APIs.

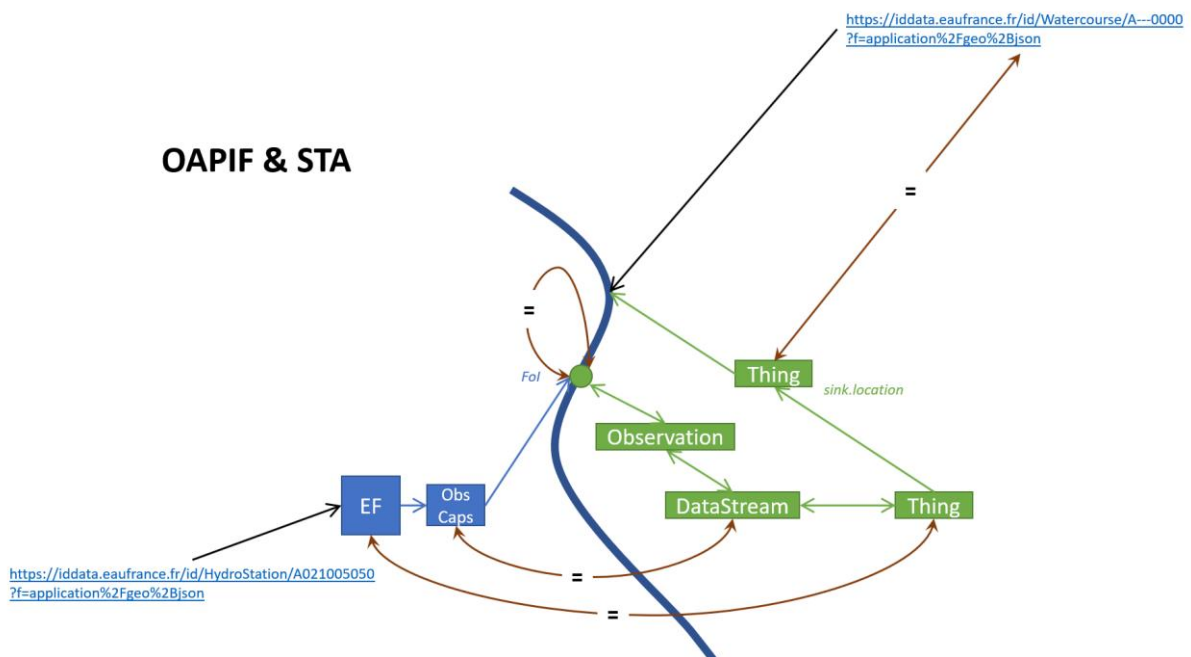


Figure 25: Combined OAPIF and STA River Network and Environmental Monitoring Facility

For prototyping purposes, these linkages have been provided in a simple ad-hoc manner by extending the data specifications by simple attributes providing the resolvable URL of the equivalent feature served by the other API. However, these two APIs exposing the same data source should be linked together in a standardized manner, ideally reusing existing mechanisms for indicating the existence and location of alternative profiles of the same data object. This topic of how to bring together the divergent OGC API approaches is currently under discussion within the OGC; further investigation is recommended in order to achieve a truly interoperable solution for such interoperability. This use case clearly emphasizes the need for clear resolvable identification of features as well as mechanisms for the provision of diverse representations (profiles) thereof, as both APIs are serving different representations of the same real world feature; while this issue was identified during the process of trying to cross-walk between the two APIs under investigation, the same principles apply to all cases

where different profiles of the same underlying data object must be provided to support the needs of different user communities.

6.2.3 Timelines and Transparency

While formally the OGC works in an open and transparent manner, OGC roadmaps are often hard to find, difficult to interpret and not complete. As OGC SWGs work on a voluntary basis, timelines are often shifted due to resource constraints among the members, with information on such delays only slowly propagated to official pages. In order to determine the actual status of current developments, one must peruse the details of the many documents and issues available from the GitHub repositories. Also, while the uptake of rapid prototyping and strong integration of hackathons and other developments sprints and events, it will be interesting if and when these many emerging API tools will eventually align with the finalized versions of the API Standards.

6.3 Recommendations for Further Work

During the course of the work on deploying and evaluating both OAPIF and STA, various issues came to our attention pertaining to the usability of both the individual API standards as well as the interplay between these APIs. While some of these issues are dependent on the extension or finalization of existing standards, others can be resolved through the provision of basic tooling as described in the sections below.

6.3.1 Error Handling

The error handling could be better in both APIs, as most errors are devolved to the basic HTTP error codes 200: success, 404: not found and 400: bad request. More detailed information on the nature of the bad request, e.g. how a service should respond to malformed user input such as a syntactically incorrect query is not specified, detailed responses are implementation dependent or not available. Extending the API specification to provide more detailed error responses on malformed or invalid requests, ideally including hints at potential resolution, often as simple as indicating that the request contains an odd number of brackets or quotes, would be a great help, especially to beginning users.

6.3.2 Data License Information

At present, data license information in INSPIRE is covered via the metadata available from a paired Catalogue Service (CSW), not from within the data models or download service endpoints. It is good practice to reference this metadata record via the gml:metadata element foreseen for all features falling under the general feature model. This method of data license information by reference to the CSW was well in line with the paradigms surrounding OWS, but may need to be reworked in light of recent API developments and philosophies such as a single point of access for all relevant information. Direct inclusion of data licensing information either within the data itself or via the APIs landing pages would make understanding if and how data can be utilized far easier for clients.

6.3.3 Validation and JSON Schema

For those familiar with OGC OWS, basic data validation is implicit due to the underlying XML encoding as well as the XML Schemas provided for the INSPIRE Themes, making data validation an integral part of a traditional OGC service landscape. However, the paradigm shift from OWS to OGC APIs has brought with it the shift from XML to JSON encoding; while JSON Schema is increasingly being utilized despite still being in draft status, it has not yet been integrated within either of the APIs.

At present, integration of JSON Schema is being investigated by the OGC for inclusion within OAPIF; once this has been integrated, clients will be able to not only validate the data, but also investigate the underlying structures of the features being provided, as presently possible through the XML Schema provided via OWS.

The STA specification explicitly declares the core data model for all classes, thus integrating the schema within the standard itself. However, this schema does not extend to contents of the properties element provided by all STA classes as of V1.1 (pertaining the Observation class, the parameters element). Allowing the API to provide additional information on the structure being utilized by a data provider for these elements would be good support for clients trying to understand the additional information provided within these standardized extension points. In addition to JSON Schema, JSON-LD is being investigated as a way of providing additional semantics both on the base data model as well as user defined extensions.

6.3.4 STA Query Builder

The advanced query capabilities available from STA endpoints allow for the creation of very complex and explicit requests. Data can be selected based on criteria pertaining to elements of multiple classes, e.g. a query can be formulated to return only those Observation result values where the value exceeds a threshold value provided within the properties of the associated Thing. Data from multiple classes can be retrieved via a single request by indicating the desired nesting, while individual attributes not required for a specific use case can in turn be filtered out, thus optimizing both the number of requests required to obtain the necessary data as well as the amount of data actually returned.

However, this power comes at a price, with beginning users often challenged in learning how to formulate queries that take advantage of the full capabilities of the API. This deficit could easily be overcome by simple tooling in the form of a STA Query Builder that helps end users navigate the core data model to interactively build queries with the required filter, select and expand functions. The resulting requests could then be easily transferred to other tools or modes of access.

6.3.5 Cross Linking between Representations

Already addressed above under the General Issues, further research is required into how to crosslink diverse electronic representations pertaining to the same real world objects, both between the different APIs evaluated here as well as between individual instances of these APIs provided by different data providers, at times in different countries. This issue also pertains to the provision of varying profiles of the same data resource, where it would often prove valuable to be able to discover alternative views on the same data source.

Within the OGC, the Interoperability Experiments (IE) ELFIE (Environmental Linked Features Interoperability Experiment) and SELFIE (Second ELFIE) have investigated options for such interlinkages, aligning and maintaining both identity and underlying semantics across representations through utilization of JSON-LD. While at the time these IEs were being carried out merging GeoJSON with JSON-LD was not possible due to underlying technical issues, thus requiring use of geosparql types for provision of spatial information, recent developments have now led to the necessary technical refinements and enabled formalization of GeoJSON-LD.

Topics that should be further investigated include definition and governance of common identifiers for real-world-objects, providing a common point of reference around which to aggregate available data resources from different data providers. Utilization of the “profile” element as defined in JSON Hypertext Application Language³⁵ is still under discussion within OGC APIs, as the requirement for provision of multiple profiles concurrently, well known to our data providers, is slowly gaining the attention of the wider standardization community.

This topic is gaining importance as ever more data becomes available online, with INSPIRE data being complemented with data from additional sources such as the Smart City Sensors we demonstrated together with the City of Hamburg. While this topic already existed in the “old” OGC web services, the new APIs bring new possibilities to the table that may help in solving this problem. The ultimate goal would be to not only be able to identify and link between entities across services, data models and INSPIRE Themes, but to also be able to search across models and INSPIRE Themes. This would lead to a truly unified geospatial data landscape.

6.3.6 Client Support

As already shown by experiences gained under the old OWS originally foreseen for provision of INSPIRE download services, the best service endpoints are mostly useless if there isn't corresponding client support available. Pertaining to the usage of OGC WFS and SOS, this issue has been slowly ameliorated through the implementation of the QGIS GMLAS toolbox plugin (https://plugins.qgis.org/plugins/gml_application_schema_toolbox/), enabling the consumption and use of GML complex features like INSPIRE harmonised data (vector), GeoSciML within QGIS, initially triggered by BRGM and co-financed by EEA/JRC. By providing interactive support not only for basic complex features within QGIS, but extending this functionality to natively support traversing the linkages between feature types, essential pertaining to SOS but also integral to many datasets provided via WFS, the QGIS GMLAS toolbox plugin greatly helped in easing the uptake of these INSPIRE services. Unfortunately, these developments didn't come until fairly late in the data provision process, causing undue friction pertaining to data provision as data providers were unwilling to provide data that they could not directly use.

³⁵ <https://tools.ietf.org/html/draft-kelly-json-hal-07#section-5.6>

In the context of the current shift to APIs, the continuation of such supporting client applications will be essential in assuring widespread uptake and use. While the superficial simplicity implied by the new access modalities and lightweight data encoding in JSON has engendered a great deal of excitement pertaining to these new data access modalities, one easily oversees the necessity of solid suite of tools underpinning these API endpoints, integrating the available data into existing workflows beyond direct browser access and simple scripts. With the recent endorsement of these new APIs by the INSPIRE MIG as INSPIRE Good Practices, it would be essential to act fast in transferring the current functionality of the QGIS GMLAS toolbox to the new OGC APIs. The open availability of such tools have been shown to greatly facilitate willingness of data provision, as financiers and decision makers can be directly convinced of the benefits of data and service interoperability when they have access to a client demonstrating the new functionality in contrast to just looking at data payloads via a browser.

6.3.7 Examples!

Finally, the one piece of feedback we received across all evaluation components - Examples! While concise standards and guidance documents are most valuable, true understanding often only comes with actually seeing and doing. All too often, the only available examples are trivialized, not reflecting the complexity encountered when faced with the task of providing real-world data. In other cases, the provided examples are inconsistent, further obfuscating the actual requirements. Ideally, examples are taken from operational datasets, assuring consistency with actual usage of the data and service models being explained. Examples from different usage areas serve to illustrate how underlying data models can be applied to different domains. Where possible, such examples should be complemented by sandbox systems that allow for interactive experimentation, letting users explore the application of these examples in an operational setting.

While more examples and interactive playgrounds are available pertaining to all aspects of STA development, deployment and usage due to its more mature nature, the complexity of the underlying data model and flexibility in applying this standardized data model to existing data sources would warrant additional more detailed examples describing such options. The tutorial pages created in the course of this project on our GitHub site together with the various demonstrators deployed in cooperation with our data providers are a good starting point for such work, whereby such a resource must be continuously extended as new usage areas for STA are explored. Fraunhofer IOSB will be integrating these materials into the already available STA resources available from the FROST-Server documentation site, thus assuring continuity and future maintenance. A simple yet valuable addition to the FROST STA implementation inspired by work on API4INSPIRE has been the provision of basic HTTP POST functionality to the underlying FROST Server directly from the landing page, bypassing the need for additional HTTP POST tools for experimentation, and thus providing built-in sandbox support.

Pertaining to OAPIF, in addition to examples illustrating the capabilities of the API, additional examples on how JSON encoding can be applied to various INSPIRE data models would be essential. The examples for Addresses and the Environmental Monitoring Facilities provided under the INSPIRE MIF Repository for action 2017.2³⁶ on alternative encodings provide a good starting point for such an example collection, once the open issues have been resolved; these could be complemented with examples from other INSPIRE Themes, ideally taken from existing INSPIRE compliant service endpoints providing additional OAPIF functionality (e.g. GeoServer endpoints, where the OAPIF configuration is derived from the existing WFS 2 configuration). Further pertaining to the INSPIRE data models, provision of JSON Schemas for these models would be a good support to MS. As the transactional aspects of OAPIF are still in the process of standardization, only read access to the API endpoints can be provided at present; thus interactive sandbox functionality is not yet an issue, simple endpoints suffice for experimentation purposes.

6.4 Recommendations for Member States

With the addition of OGC APIs to the rich set of options for the provision of INSPIRE Download Services, it becomes increasingly difficult for MS to determine which of these many options is best suited towards enabling access to the data they wish to provide while making best use of their existing infrastructure investments. In the following sections, the experiences and insights gained during this project and related work is reflected upon, in order to provide guidance for data providers both extending their existing systems as well as for those now enabling access to new datasets.

The initial question that must be posed by MS data providers pertains to the nature of the data being provided, as well as the INSPIRE Themes this data pertains to. The underlying datasets for many INSPIRE Themes consist

³⁶ <https://github.com/INSPIRE-MIF/2017.2>

solely of static spatial objects, modelled in INSPIRE as GML features, and to date usually provided via WFS2; for these cases, OAPIF is the logical choice when shifting to provision via API. For the provision of more dynamic data pertaining to spatial objects, e.g. values that change over time, the INSPIRE Data Specifications have integrated the OGC/ISO Observations and Measurements models, thus to date implying provision via OGC SOS; for these cases, STA is the logical choice when shifting to provision via API.

A further question that must be posed before deciding on provision modalities pertains to the requirements of the potential data users, the final rationale behind the entire data provision exercise. In cases where many users are limited to tools that can only consume WFS2 or SOS, more weight must be given to the provision of these OWS, whereas in cases where there is a strong interest from developer communities in accessing and utilizing the data, provision of APIs is recommended.

Finally, existing investments in service infrastructure must be taken into account. In dependence on the data provider's systems, shifting to the new APIs can be anywhere from exceedingly simple to simply impossible. For example, the way the OAPIF extension for GeoServer has been implemented, the entire existing server configuration as well as database mapping is retained when the OAPIF extension is deployed, providing instant access to all existing data originally made available via WFS2. In other cases where the existing SW does not provide such migration, at worst case the entire API configuration must be reimplemented.

6.4.1 Considerations towards OAPIF

OAPIF as it stands today is still a very simple API standard, as only "Part 1: Core" and "Part 2: Coordinate Reference Systems by Reference" have been finalized to date. In this form, it provides basic access to feature data, allowing for the simple creation of lightweight applications by just "following the links" to related objects. Once "Part 3: Filtering and the Common Query Language (CQL)" and "Part 4: Simple Transactions" have been finalized, this API will provide a robust mode of interacting with the available data, as users will not only be able to define complex queries to retrieve only required data but also be able to use full Create, Read, Update and Delete (CRUD) functionality via OAPIF.

When considering upgrading existing infrastructure, as OGC API - Features is not yet ready to fully replace WFS2, the question is if adding early support for OAPIF is possible in a cost-effective way. If the used server software has support for the new API, it is likely that enabling the new API can be done without much effort. For example, when using GeoServer, one only needs to deploy the OAPIF extension; upon restart, all Features originally available via WFS will also be available via OAPIF. Doing so is highly recommendable, since users with relatively simple use-cases will greatly appreciate the new API. If the currently utilized server software does not support OAPIF, it should be investigated if OAPIF support is planned or not on the roadmap. If support is planned for the not too distant future, it is best to wait, and upgrade the existing software once the new version is available. If support for OAPIF is not planned, it would be wise to start investigating and experimenting with alternative solutions. Once OGC API - Features has the same functionality as WFS2, a full migration should be considered.

When setting up new infrastructure to publish feature data, data providers should definitely foresee provision of OAPIF in whatever system they choose for this purpose. While at present it is fairly easy to implement OAPIF directly on top of a spatial database as shown by the OGC API - Simple development done in the course of this project, the complexity will strongly increase once Parts 3 & 4 are finished, thus requiring a dedicated development team for implementation and maintenance. In most cases, it will prove far more efficient to utilize existing tools for data provision, whereby the ability to provide both WFS2 and OAPIF should be part of the selection criteria.

While APIs are far easier to use than OWS, the basic tenets of data provision still apply, existing data must be transformed and mapped to harmonized models before provision, an often painful process. In order to simplify the data provision process, this project has implemented two powerful extensions to GeoServer, providing a novel new method for interactively configuring complex features. The first extension allows for provision of data directly in the native database structure; similar to existing functionality exposing a single database table as a Simple Feature, the SmartDataLoader extends this functionality to Complex Features, allowing for the inclusion of data from associated tables, while letting the user prune unnecessary columns and associations. Once all necessary data has been exposed in the native database structure, the additional Templating component comes to play; in this approach, an example of the desired final output data serves as a template, and the user need only provide the path of the dynamic parts of the output data from within the database native endpoint configured via SmartDataLoader. In addition to making data provision far more intuitive, the Templating mechanism will also allow for the simple creation of diverse data profiles customized to the needs of individual communities. At present this functionality is available for both GeoJSON and JSON-LD response formats; work on extending this to both HTML and XML response formats is ongoing.

6.4.2 Considerations towards STA

While sensor data has often been seen as a slightly exotic side topic within INSPIRE, with the increasing dominance of the IoT and the emergence of Digital Twins, such highly dynamic data is becoming increasingly prevalent. While some of this data goes beyond the direct scope of the INSPIRE Themes, in many cases it is tightly linked to INSPIRE base data as shown in our Smart City Data Nest concerning the City of Hamburg, where real-time data on traffic and charging stations requires INSPIRE base layers such as the Road Transport Networks for context.

Existing infrastructure in the domain of observational time-series data usually means existing installations of the Sensor Observation Service (SOS). The SensorThings API can completely replace SOS, and has many usability advantages over SOS, so it is likely that users of the data will greatly appreciate access through the SensorThings API. However, simply replacing existing implementations with a new (different) API is usually not a good idea as it would break all workflows of users that depend on these legacy deployments. The easiest solution for adding STA compatibility to an existing deployment is to install a version of the server software that supports STA next to SOS. This has the advantage that data is not duplicated, there is no need to keep data synchronised between two servers and existing data-import workflows do not need to be changed. However, not all SOS servers support the SensorThings API. In this case a separate server instance will need to be deployed for STA support. Duplicating data from an SOS server to a STA server is quite simple due to the underlying use of the O&M Standard, as the ad-hoc air quality endpoint, directly harvested in real-time from operational INSPIRE SOS, has demonstrated.

For the publication of new observational data in situations where existing deployments are not relevant, using the OGC SensorThings API is highly recommended. The API is mature and supports a large variety of use-cases, from simple to very advanced. There is also a good number of implementations on both server- and client-side. Use of the SensorThings API is not limited to data that fits the traditional “observational data” description, as specified under various INSPIRE Themes, foremost Environmental Monitoring Facilities, but has also been shown to be very suitable for the provision of a wide range of topic areas, for example, statistical (demography) data, Covid-19 case data or vaccination status by Administrative Unit, or dynamic data in the area of Smart Cities, such as E-Car charging station availability. In reality, STA has very little to do with Sensors, the API is well suited for the provision of all dynamic data pertaining to features due to the integrated support for time-series and provision of measurement metadata. The API can even be used to publish data that would traditionally be seen as “feature” data, like for example rivers and water bodies, though this is not the strong-point of the SensorThings API, and should only be utilized when the focus is on the accompanying measurements.

List of abbreviations and definitions

Acronyms

| | |
|-------------|--|
| API | Application Programming Interface |
| CSV | Comma Separated Values |
| DB | Database |
| desktop GIS | Geographic Information System running on a desktop |
| DWBP | W3C Data on the Web Best Practices |
| ELISE | European Location Interoperability Solutions for e-Government |
| FROST | FRaunhofer Opensource SensorThings-Server |
| GeoJSON | An open standard format designed for representing geographical features. |
| GIS | Geographic Information System |
| HTML | HyperText Markup Language |
| INSPIRE | Infrastructure for Spatial Information in the European Community |
| INSPIRE MIF | INSPIRE maintenance and implementation framework |
| INSPIRE MIG | INSPIRE maintenance and implementation group |
| JSON | JavaScript Object Notation |
| MS | Member State |
| OGC | Open Geospatial Consortium |
| OGD | Open Government Data |
| OWS | OGC Web Services |
| REST | Representational state transfer |
| SF-X | Simple Features Level X |
| SOAP | Simple Object Access Protocol |
| SDK | Software Development Kit |
| SDWBP | Spatial Data on the Web Best Practices |
| STA | SensorThings API |
| SWG (OGC) | Standards Working Group |
| W3C | World Wide Web Consortium |
| WFS2 | Web Feature Service version 2 |
| XML | Extensible Markup Language |

Definitions

| | |
|------------------------------------|---|
| Administrative Stakeholders | Those responsible for providing resources for the provision and use of APIs |
| Data Providers | Organisations that make data available. |
| Data Users | People or organisations that use data made available by data providers. |
| Degrees of Freedom improvements | directions in which recommendations can lead to concrete impacts and |
| Evaluation Criteria | individual criteria derived from the Five Level Evaluation Model by Moilanen |
| Evaluation Types | methods of interrogating stakeholders: Heuristic Expert Evaluation, Peer Review, Interview, Questionnaire |
| Five Level Evaluation Model | API evaluation model by Jarkko Moilanen |

Peer Review

evaluation approach for API development evaluation defined by Farooq

Stakeholder Perspectives

different stakeholder roles: development, deployment, usage

Standards

The following standards are referenced throughout this document:

OGC® 07-036r1: OpenGIS® Geography Markup Language (GML) Encoding Standard

OGC® 09-025r2: OGC® Web Feature Service 2.0 Interface Standard – With Corrigendum

OGC® 10-100r3: Geography Markup Language (GML) simple features profile (with Corrigendum) (2.0) - profile

OGC® 12-006: OGC® Sensor Observation Service Interface Standard

OGC® 14-055r2: OGC OWS Context GeoJSON Encoding Standard

OGC® 15-078r6: OGC SensorThings API Part 1: Sensing v1.0

OGC® 16-032r2: OGC WaterML 2: Part 4 – GroundWaterML 2 (GWML2)

OGC® 17-069r3: OGC API - Features - Part 1: Core

OGC® 18-088r1: OGC SensorThings API Part 1: Sensing v1.1 (in public review)

Supporting Documents

Good Practice: INSPIRE download services based on OGC API - Features: <https://github.com/INSPIRE-MIF/gp-ogc-api-features>

OGC Web API Guidelines: <https://github.com/opengeospatial/OGC-Web-API-Guidelines>

OGC API Hackathon: <https://www.opengeospatial.org/projects/initiatives/oapihackathon19>

GDAL Support for OGC API - Features: <https://gdal.org/drivers/vector/oapif.html>

Simple Features GeoJSON Java: <https://github.com/ngageoint/simple-features-geojson-java>

W3C Data on the Web Best Practices: <https://www.w3.org/TR/dwbp/>

W3C Spatial Data on the Web Best Practices: <https://www.w3.org/TR/sdw-bp/>

W3C Semantic Sensor Network Ontology: <https://www.w3.org/TR/vocab-ssn/>

List of figures

Figure 1: Evaluation Process Overview..... 5

Figure 2: Overview of the evaluation process. The consultation step has been split by stakeholder perspective, as these different perspectives require quite different approaches to the evaluation process. All evaluation steps are described in greater detail in the sections below..... 31

Figure 3: Analysis overview of level of achievement 35

Figure 4: balance between deployment and use effort..... 35

Figure 5: Deployment Methodology 40

Figure 6: GeoServer New Smart App-Schema Data Source 45

Figure 7: OGC API - Simple Interface 47

Figure 8: LD-Proxy - add new service 48

Figure 9: LD-Proxy - show feature types 48

Figure 10: LD-Proxy - feature attributes..... 49

Figure 11: LD-Proxy - schema mapping 49

Figure 12: LD-Proxy - TN-A Data 50

Figure 13: OAPIF Development Perspective Evaluation Overview 60

Figure 14: OGC API Development Roadmap 61

Figure 15: OAPIF Deployment Perspective Evaluation Overview 64

Figure 16: User Feedback - Examples Required 65

Figure 17: STA Development Perspective Evaluation Overview 67

Figure 18: STA Deployment Perspective Evaluation Overview 69

Figure 19: STA User Perspective Evaluation Overview 71

Figure 20: Evaluation Overview 74

Figure 21: Evaluation Overview Numeric (1 Best, 3 Worst) 74

Figure 22: OAPIF River Network and Environmental Monitoring Facility 76

Figure 23: STA River Network and Environmental Monitoring Facility 77

Figure 24: Combined OAPIF and STA River Network and Environmental Monitoring Facility 77

List of tables

| | |
|--|-----|
| Table 1: Stakeholder Overview | 9 |
| Table 2: Overview of Evaluation Criteria..... | 14 |
| Table 3: Evaluation types for the stakeholder perspectives develop and deploy | 27 |
| Table 4: Evaluation types for the stakeholder perspective use | 29 |
| Table 5: Overview Data Providers and Datasets | 56 |
| Table 6: shows how the criteria in the Five Level evaluation model relate to the characteristics defined by ISO 25010. A comparison shows that by adding the “Suitability”-criteria, all aspects of the ISO standard are covered by our Five Level model as well. | 100 |
| Table 7: STA Datastream Mapping Austrian Air Quality | 101 |
| Table 8: STA Sensor Mapping Austrian Air Quality | 102 |
| Table 9: STA FoI Mapping Austrian Air Quality | 103 |
| Table 10: STA Thing/Location Mapping Austrian Air Quality..... | 104 |
| Table 11: STA Observation Mapping Austrian Air Quality | 104 |
| Table 12: STA Observation Mapping EEA Air Quality..... | 105 |
| Table 13: STA Observation Metadata Mapping EEA Air Quality | 106 |

Annexes

Annex A - Extended evaluation criteria

This annex lists the evaluation criteria, with questions illustrating how the criteria apply to the three different stakeholder perspectives.

Level 1: All find

This level corresponds to very basic non-standardized data provision, data is available together with simple examples. This level focuses on in-house use of APIs. Requires minimal effort in provision but is difficult to impossible to use.

Relevant questions: Is the API mainly designed for in-house use? Did the API happen by accident? Did the API evolve without relevant oversight or adherence to standards? Was the API designed based on a large set of realistic use-cases?

Single Entry Point

Is all information available from a single source (the portal), either directly or through links?

Finding all data one needs is a key requirement for any task. Ideally, there is a single place (the portal) through which all required information can be reached. For development this pertains to the API description, for deployment the data model definition, and for API use the data exposed through the API and the metadata required for understanding the data.

Effort required: 1.

Benefit provided: 4

Dev: Does the API documentation contain all required information for implementing the API? Does it contain clear references to all other APIs or standards that are used as part of the API?

Deploy: Does the API documentation contain all required information for modelling the use-case data? Can metadata be published through the API or does it have to be supplied through other means? Does the API have a single entry point, or do different end-points need to be documented elsewhere?

Use: Is all the data provided by the API available through a single entry point (e.g. starting page?). Is all metadata available through the API, or does this have to be downloaded from a separate location?

Documentation

Is updated documentation available?

Documentation is the main source to get information about the API. It should answer all the questions arising for all interested in the API, be it the software developer implementing the server, the publisher mapping his data to the data model, or the user requesting data. The documentation should be clear and precise, so that no "trial and error" is needed.

Effort required: 2.

Benefit provided: 5

Dev: Is the API Standard clearly documented (specified), to allow an unambiguous implementation?

Deploy: Does the documentation clearly explain how to map use case data to the data model of the API?

Use: Is the API clearly documented and understandable? Does the documentation describe how the API can be used? Does the documentation offer the information needed to realize the intended use case? Are the usage constraints clearly described?

Example Requests

Are there examples of API requests as part of the documentation?

The interaction point with an API are requests, created by a client-(application), sent to the API implementing server, processed and the result is sent back. The request contains all information, which describes the users information needs. Example requests help with understanding the documentation from all perspectives.

Effort required: 2; Community input possible.

Benefit provided: 5

Dev: Are there example requests available which show a potential usage of the API? Is it clear what incoming requests to expect, based on the examples?

Deploy: Do the example requests illustrate how the data modelling influences the use of the data? Are there example requests available which show a potential usage of the API? Are all aspects covered by the sample requests?

Use: Are example requests available demonstrating the functionality of the API? Are different use cases covered by the sample requests? Can the samples be used as a basis for one's own use-case specific requests?

Example Data

Are there examples of the data returned by API requests?

After processing the received request, the server returns the requested data. It's advantageous if the example data matches the example requests, and is described in detail.

Effort required: 2; Community input possible.

Benefit provided: 5

Dev: Is there example data available? Is it possible to use the example data to test the API implementation? Is the example data available in a format that is expected as response from the server?

Deploy: Is there example data available which can be loaded into the implementation? Is this example data covering all aspects of the use-case? Is there an easy alignment of the example data and the data that should be exposed by the API? Are there example responses available which can be used as a reference to validate the results provided by the deployed server?

Use: Are there example responses available that show possible results from the server? Is example data available which can be easily imported into an existing implementation, to test the use of the API?

Discoverability

Is it possible to discover deployed instances of the API based on the resources provided?

What means are available for discovering deployed instances of the API? Are dedicated services such as the OGC Catalogue Service Web (CSW) required for discovery? Can deployed instances of the API be discovered via normal search engines following W3C Data on the Web Best Practices (DWBP)³⁷ and Spatial Data on the Web Best Practices (SDWBP)³⁸?

Note: this criterion was added to reflect the organizational and standardization background against which this evaluation is being performed.

Effort required: 2.

Benefit provided: 5.

Dev: Are mechanisms for making deployed instances of the API discoverable available? Does making the API discoverable by search engines greatly increase the implementation effort?

Deploy: Must discovery criteria be explicitly specified, or is it automatically extracted from the data resources? Must additional discovery services be deployed, or is discovery via search engines integral to the API?

Use: Is it easy to discover deployed instances of the APIs providing relevant data? Can deployed instances of the API be discovered via search engines or must dedicated services/APIs be utilized for discovery?

Level 2: All use

This level corresponds to basic standardized data provision. Data is available in a standardized format, basic standards based access functionality is provided. This level focuses on providing data to external users. Medium effort in provision, use is possible, but not optimal.

³⁷ <https://www.w3.org/TR/dwbp/>

³⁸ <https://www.w3.org/TR/sdw-bp/>

Relevant questions: Has the API been reviewed by external users? Has the API been designed for providing data to external users?

JSON or XML

Does the API support the use of JSON and/or XML?

Data needs to be represented in a serialized form to be transmitted. For web-based APIs, JSON and XML are established as de-facto-standard, since they're human readable and many implementations exists for various programming languages, which allows easy integration and reuse. These aspects are relevant for server development and API use, but not for the deployment.

Effort required: 1.

Benefit provided: 4

Dev: Should the response of the API provide data in JSON or XML? Does the incoming request containing JSON/XML data? Is the JSON/XML-representation specified in a way that allows the easy implementation?

Deploy: Not relevant.

Use: Is the provided data available in JSON/XML? Is JSON/XML used to create requests? Are both representations available? Is it easy to reuse existing tools/libraries to parse the representation?

Data License

Are data license details given through the API?

The main value of an API is provided through the data that is published through the API. Therefore, it's important that the license of the data is also available through the API itself. The data license is usually a legal document, and in most cases this document will be linked to from within certain API responses. Ideally, the data license is based on a standard licensing scheme such as CC BY.

Effort required: 1.

Benefit provided: 3

Dev: Is it clear how data license information should be made available through the API?

Deploy: Is there a way to make the data license available through the API? Is it possible to add all required licensing information?

Use: Is the data license available through the API? Is it clear where to find the data license in the API? Is the data license easy to find and understand? Is a standard license (such as CC BY) being used?

Terms of Use

Are Terms of Use clear and easily accessible?

In addition to data license, the terms of use should be available. A data provider should be able to specify how and with which constraints an API service can be used, and it should be clear to users of the API where to find this information. The terms of use are usually described in a legal document, and in most cases this document will be linked to from within certain API responses.

Effort required: 1.

Benefit provided: 3

Dev: Is it clear how the terms of use should be made available through the API? Does the API describe technical mechanisms to enforce the terms of use?

Deploy: Can the Terms of Use be configured in the server? Is there a technical mechanism to enforce the Terms of Use (e.g. rate limiting)? Is it possible to enforce the terms of use with external mechanisms not defined by the API?

Use: Are the terms of use accessible through the API? Is it clear under which constraints the API can be used? Is it clear when technical mechanisms that enforce the terms of use take effect?

Embedded Metadata

Does returning data include metadata?

Getting data is often not sufficient. Additional metadata is needed to use and interpret the data correctly. E.g. what is represented by the data? What are the units? The API must allow the developer to enable the provider to provide the metadata required by the user to interpret the data.

Effort required: 3 – 4.

Benefit provided: 4

Dev: Is it clear how metadata should be included in the API? How flexible is the mechanism for providing metadata?

Deploy: Is it possible to integrate all relevant metadata into the API and its data model? Is it clear how this should be integrated?

Use: Is the relevant metadata available or is some information missing? Is it possible to automatically consume/process the provided metadata?

Authentication

Does the API support authentication/authorization?

Not all data should be available publically. To limit access to authorized users, authentication/authorization (auth*) mechanisms exist. To ease integration, the auth* methods should be based on existing, well-known protocols (e.g. OAuth³⁹ or OpenID-Connect⁴⁰). After authenticating, the authorization mechanisms define exactly which data the user is allowed to see.

Effort required: 2 – 4.

Benefit provided: 3

Dev: Does the API specify how to deal with authentication / authorization? Does the API specify security mechanisms? Does the API limit the used security mechanisms to the specified ones? Is it clear how authorization should be applied to the data model?

Deploy: Is it possible to enable/configure an auth* method on the server? Is it possible to configure multiple authentication methods? Is it possible to integrate with existing identity management systems? Can all security aspects be handled by the auth* method? Is it possible to configure authorization methods? Is it clear how authorization methods interact with the data model?

Use: If using the API needs some kind of authentication, is it based on existing methods that can be reused? Is there a clear way to discover which authentication method should be used? Is it possible to seamlessly manage/use the provided authentication method while accessing the API?

API Standardization

Is the API itself standardised, and is this specification openly available?

Note: this criterion was added to reflect the organizational and standardization background against which this evaluation is being performed.

For an API to become widespread, it is helpful if the API is formally adopted as a standard by a well-respected, international standards body. Furthermore, an API standard rarely stands alone. Most standards refer to other standards for specific aspect. For example, in most standards, whenever a date or time is used, the encoding is done according the ISO 8601 standard. Referencing existing standards usually reduces the effort required for implementing the standard and makes it easier for clients to use the standard, since they can use common libraries that implement these referenced standards. It also reduces the effort of mapping data models, since standardised building blocks are likely to already be in use.

Effort required: 5; Community input required.

Benefit provided: 5

Dev: Is the API itself standardised? Does the API re-use existing standards that are implemented/supported by mature libraries? Are the standards that the API references readily available?

³⁹ <https://en.wikipedia.org/wiki/OAuth>

⁴⁰ https://en.wikipedia.org/wiki/OpenID_Connect

Deploy: Is the API itself standardised? Does the API data model re-use existing data model standards? Are the standards that the API references readily available? Is it possible to map the existing data model to the API? Is this easily possible?

Usage: Is the API itself standardised? Does the provided API correspond to the API standard and the defined data model? Does the API re-use existing standards that are implemented/supported by mature libraries? Are the standards that the API references readily available?

Suitability

Is the API suitable for the intended use?

Note: this criterion was added based on the criteria available from within the ISO 25101 Standard.

For an API to be deployed and used, it has to be suitable for the intended use case. From the deployment perspective, this means the API has to be implemented in server software that fits in the deployment landscape of the data provider and that can be connected to, or loaded with, the data that the data provider intends to publish. From the user perspective, this means that the user must be able to request, in a suitably efficient manner, the data that he needs.

Effort required: 3.

Benefit provided: 4

Dev: Not relevant.

Deploy: Is it possible to deploy the implementation to the existing infrastructure? Can the existing infrastructure be reused? Is it possible to use existing data sources and expose them, using the API implementation?

Use: Having a given use-case is it possible to realise all involved aspects with the given API? Is the overhead involved with the use of the API acceptable?

Level 3: All trust

This level corresponds to mature standardized data provision. Encompassing data is available in complex standardized format, powerful and mature standards based access functionality is provided. Fairly large effort in provision, use is well supported.

Relevant questions: Has the API been designed for use in a diverse set of use cases, in a diverse set of environments? Can the API support the complexity of data models required for real-world use cases? Has the API been designed for use with large data sets?

Query and Analytics API

Does the API include Querying and Analytics?

Note: this criterion was modified from the original to include query functionality together with analytics

APIs are often used to get data from a service. Usually only a subset of the available information is of interest. Therefore, the API should contain querying and analytic capabilities. First, this includes a suitably powerful filter mechanism to limit the response to those parts of the data that the user is interested in. Second, this includes a mechanism in the API to do basic analytical calculations, e.g. some aggregation functions.

Effort required: 4.

Benefit provided: 5

Dev: Is the filter and analytic mechanism of the API clearly defined and easy understandable? Is it easy to implement this functionality? Are libraries available for development of these filters?

Deploy: Is the available data source able to handle the filter/analytic requests coming from the API server implementation? Or is there a need to fully import and transform the data into the server to have the full analytics API available?

Usage: Does the API offer a possibility to filter for the needed data? Is it possible to map the use-case specific request to the supported filter mechanism? Does the API offer the needed aggregation functionality or is some post-processing needed? Are those mechanisms easy to understand?

Error Handling

Is Error Handling in place and documented?

While using an API errors might occur. Either there's an issue with the server itself, or the data sent by the user isn't correct or doesn't match the available data. To allow a user to handle these errors, the API should specify how errors are reported back to the user. At the same time, the error message should not expose sensitive internal information about the server deployment.

Effort required: 2.

Benefit provided: 4

Dev: Are there standard error codes and error responses defined? Is standard behaviour defined for when errors are encountered? Is the error behaviour aligned to existing best practices?

Deploy: Is it possible to create an inconsistent data model that will cause errors for the client? Are there good ways to find errors in the data modelling? Is it clear which error messages may expose internal information?

Use: Is there a well defined error behaviour? Are error messages and error codes defined in the API? Is it clear how to handle specific errors? If there is an error in the usage, is some information provided, how to fix adapt and fix the usage?

Performance and Cache

Can the API offer sufficient performance and does it support caching?

Note: this criterion was modified from the original to include performance functionality together with caching

APIs may have inherent performance bottlenecks that become apparent when deploying and using the API with large data sets or a large number of users. To increase performance and efficiency, caching mechanisms may be used.

Effort required: 2 – 4.

Benefit provided: 3

Dev: Is caching of queries part of the API standard? Is the caching mechanism clearly specified? Is the caching mechanism defined in a way that existing implementations (or even existing infrastructure) can be reused? Are there inherent performance bottlenecks in the API? Is it possible to create a server that scales horizontally (by adding more back-end servers)?

Deploy: Is there a way to use existing caching infrastructure, external to the server software? Are there limits to the amount of data served by a single server instance? Can horizontal scaling be used to increase the number of clients served?

Use: Is the caching mechanism clearly described? Is it clearly defined if and when a user might get outdated data? Does the standard offer the possibility to establish a client-side caching?

Background Support

Is the development and maintenance of the API supported by a big, stable entity or company?

Deciding to use a specific API requires investments on all sides (dev, deploy, use). Thus, for the future development of the API itself and the implementation is important, to be sure that the API will still be relevant in the future. A big entity or company in the background increases this probability.

Effort required: 1 – 5; Community input possible.

Benefit provided: 4

Dev: Is the API standard supported by a big entity or company?

Deploy: Is the entity developing the API well-known, and is it thus likely that there are multiple server implementations of the API? Are there multiple, independent server implementations?

Use: Is there big entity or a company in the background which supports the standard?

Availability

Is it easy to integrate into existing workflows and toolsets?

Many users and domain experts have well established workflows that use existing tools and (desktop) software packages. Switching to a different data source or API may break these workflows, which would greatly reduce the incentive for users to switch to the new API. These tools can include those used by providers to import data

from external or primary sources into the local data infrastructure, or those used by data users to visualise or otherwise use the data.

Note: this criterion was modified from the original to include availability of relevant tooling.

Effort required: 4; Community input required.

Benefit provided: 4

Dev: Not relevant.

Deploy: Are there tools available that help with the data mapping? Are there tools available that help with the data import into the server?

Use: Are there plugins available for the client software that is popular in the domain of the API.

API Data Validation

Can the data returned by the API be validated?

For interoperability it is important that the data returned by an API conforms to the data model defined by the API. It is beneficial if there is a way to automatically check this, for example by checking the JSON against a JSON Schema, or XML against an XML Schema Definition.

Note: this criterion was added to reflect the organizational and standardization background against which this evaluation is being performed.

Effort required: 3; Community input possible.

Benefit provided: 4

Dev: Can a developer verify that the implementation returns correctly modelled data?

Deploy: Is there a way to verify that the data mapping is done correctly by validating the returned data?

Use: Can a user easily verify that the data being provided by the API conform to a given definition?

Level 4: All involved

This level pertains more to the maturity of the wider ecosystem built up around the standardized specifications. The aspects described go beyond the control of the individual data providers and thus must be supported through diverse stakeholder communities. Achieving this level facilitates both the provision and usage perspectives.

Relevant questions: Can support for the API be seen as a community effort?

SDK Availability

Are API SDK's available for one or more environments?

Software Development Kits (SDKs) simplify the interaction with the API on the development and client side. They contain libraries that reduce the amount of code that needs to be created to interact with the API and help reuse existing work and integrating the API in new contexts.

Effort required: 3; Community input possible.

Benefit provided: 2

Dev: Is there an SDK available which implements the API? Are all aspects covered (e.g. parsing the request, parsing filters, ...)? Is the SDK usable regarding license and technical implementation?

Deploy: Not relevant.

Use: Is there and SDK available for using the API? Are all aspects covered (e.g. parsing the request, parsing filters, ...)? Is the SDK usable regarding license and technical implementation?

Code Examples

Are there examples of code in one or more commonly used programming languages?

Like SDKs, code examples simplify and clarify the interaction with an API. They show in detail how certain aspects of the API should be used and can often directly be executed to see the covered features live in action.

Effort required: 3; Community input possible.

Benefit provided: 4

Dev: Are there code example available, showing how to implement specific aspects of the API? Are those examples in one or more programming languages? Is it possible to transfer those examples to other programming languages?

Deploy: Not relevant.

Use: Are there code example available, showing how to use specific aspects of the API? Are those examples in one or more programming languages? Is it possible to transfer those examples to other programming languages?

Community

Is there a growing community to consult if needed?

A big, active and friendly community can be, in addition to the documentation, an additional source of information. Questions, best-practices and issues can be discussed within a community to spread available knowledge and provide support.

Effort required: -; Community input required.

Benefit provided: 4

Dev: Is there a community available which can help solving API specific issues while implementing the standard? Are the provided communication channels adequate to consult the community? Is the community friendly and open for new members?

Deploy: Is there a community available which can help solve issues with the data modelling and mapping? Are the provided communication channels adequate to consult the community? Is the community friendly and open for new members?

Use: Is there a community available which can help solving API specific issues while using the standard? Are the provided communication channels adequate to consult the community? Is the community friendly and open for new members?

Playground

Is there an API Playground for testing and getting familiar with the API?

Learning by doing and practically testing the usage of the API helps getting more insights. A playground helps with this. Such a playground can range from a public server that anyone can access, to a Docker image that can be quickly deployed with standard settings, to a one-click-install installation package that can run on a desktop PC.

Effort required: 2; Community input possible.

Benefit provided: 3

Dev: Is there a playground available which can serve as an example/reference implementation which can be used to match the own implementation?

Deploy: Is there a playground available showing a sample implementation of the API? Is it possible to simply adapt the content of the playground to your use-case?

Use: Is there a playground service, implementing the API standard available? Does this playground contain sample data, allowing to test all aspects of the API? Is it possible to simply adapt the content of the playground to your use-case?

Linked Documentation

Is documentation linked to code examples and back again?

A documentation that links to code examples, example data and request (and back) helps the reader to better understand the API.

Effort required: 2; Community input possible.

Benefit provided: 3

Dev: Does the documentation link to the examples? Is there also a backward link from the examples to the documentation?

Deploy: Does the API documentation link to data modelling examples, and do those examples link back to the API documentation?

Use: Is there a documentation available which links the documentation to the samples? Is there also a backward link from the examples to the documentation?

API Evolution

Can a developer, data provider or user provide feedback on issues with the data model or API functionality and are custom extensions to the API foreseen?

Note: this criterion was added to reflect the organizational and standardization background against which this evaluation is being performed.

APIs are rarely perfect and finished in their first incarnation. They also always have to strike a balance between the diverse requirements of many different use-cases on the one hand, and complexity on the other. Because of this, it is important that developers, providers and users have a way to provide feedback on issues, deficiencies and unclarities in the API. It is important that these issues are addressed in future versions of the API. Similarly, it is very helpful if the API has clear extension points so that the API can be extended for certain use cases that require functionality that is not in the API.

Effort required: 2; Community input required.

Benefit provided: 2

Dev: Can issues with the API be reported? Are these issues taken into account in the development of the next version of the API? Are standard extension points foreseen by the data model and the API?

Deploy: Can issues with the data model of the API be reported? Can the data model be extended to better fit a use case?

Use: Can issues with the API be reported? Is it clear which API and data model extensions are active on a given server instance?

Level 5: All develop

This level describes a well developed and mature ecosystem built up around the standardized specifications. The aspects described go far beyond the control of the individual data providers and thus must be supported through diverse stakeholder communities. Achieving this level facilitates both the provision and usage perspectives.

Relevant questions: Can (further) development of the API be seen as a community effort?

Code Visible

Is code visible/can be cloned?

Having access to the source code of a reference implementation allows a deeper understanding of the implementation. It offers the possibility to check if specific behaviour was intended or is a bug. Available open-source code with a suitable license offers the possibility to add own changes, so that there is no dependency on a third-party.

Effort required: 5; Community input required.

Benefit provided: 3

Dev: Is the source code of a reference implementation of the API available? Is the source code well structured, so that issues that arise during the implementation can be checked against this reference implementation?

Deploy: Are there open-source server implementations of the API? Is the source code well structured and aligned to best practices in software development?

Use: Are there open-source client libraries available? Is the source code well structured, so that issues that arise during the usage can be checked in other implementations?

Bug Tracker

Can bugs, issues and suggestions be reported in a public place and is this dialogue public?

Though an API is not software and can thus not have "bugs" in the traditional sense, an API can still have inconsistencies, errors and unclear definitions. Often these issues are not noticed until the API is applied in

specific use cases, or implemented by multiple people. Having a Bug Tracker publically available offers a place, where to report issues and to track discussions and solutions. Having such a system openly available allows input from a wider community, helping the system to evolve to support a wider user community. It also makes it easier to collaborate on extensions.

Effort required: 1; Community input possible.

Benefit provided: 4

Dev: Is there a Bug Tracker available to report inconsistencies or errors in the API definition and discuss and design future extensions?

Deploy: Is there a Bug Tracker available to report inconsistencies or omissions in the data model that the API defines? Is there a place to discuss data model extensions?

Use: Is there a Bug Tracker available to report inconsistencies or errors in the API definition? Can feature request and extension proposals be discussed there?

API License/reuse

Is the API's license known, are parts of the API covered by patent claims, and does it allow further development and re-use?

Parts of APIs can be covered by patent claims, making it impossible to implement the API without paying royalties. The license of the API is important and might be a blocker, if the API needs to be re-used or if further development of the API is required. Ideally, if a license is required this should be based on a standard licensing scheme such as CC BY.

Effort required: 2; Community input possible.

Benefit provided: 3

Dev: Can the API be implemented without using patented technologies? Is the license of the API known? Is the license of the API based on a standard and open licensing scheme such as CC BY? Is it possible/allowed to re-use the API? Is it permissible to further develop the API?

Deploy: Is the license of the API known? Is it possible/allowed to re-use the API? Is it possible/allowed to further develop the API?

Use: Is the license of the API known? Is it possible/allowed to re-use the API? Is it possible/allowed to further develop the API?

Development Roadmap

Is the API's development roadmap known and is it visible for all?

A roadmap can help to understand the further development direction of the API and to know what to expect in the (near) future.

Effort required: 3; Community input possible.

Benefit provided: 2

Dev: Is there a development roadmap of the API available? Is it clear what to expect in the following months/years?

Deploy: Is there a development roadmap for the server implementation available? Is there a clear direction for the future?

Use: Is there a development roadmap of the API available? Is there a development/deployment roadmap of the data provider available?

Linked Data Ready

Is the data provided structured in a Linked Data ready manner, i.e. JSON-LD?

Note: this criterion was added to reflect emerging technological advances to be expected within the stakeholder community.

Linked Data is a technology that looks very promising and that has been on the horizon for some time now, with parts and concepts of it finding their way into APIs and data models. While it is not yet practical to have

an API that fully employs all Linked Data principles, it is possible to design the API and data models in a way to allows Linked Data adoption in the future.

Effort required: 3; Community input required.

Benefit provided: 3

Dev: Can Linked-Data principles be applied to the API responses without violating the API specification?

Deploy: Can Linked-Data principles be used in the data model?

Use: Not relevant.

Test Framework available

Can conformance to the API be formally tested?

Note: this criterion was added to reflect the organizational and standardization background against which this evaluation is being performed.

Test Frameworks can be used to verify the implementation and deployment of the API. This helps interoperability by insuring the deployed services implement the API correctly.

Effort required: 4; Community input required.

Benefit provided: 3

Dev: Is there a test framework available to automatically verify the compliance of a server implementation with the API? Is the test framework publicly available so that the test can be run easily?

Deploy: Is there a test framework available to automatically verify that the server is deployed compliant with the API? Is the test framework publicly available so that it can be run easily?

Use: Is there a test framework available to test if the usage is compliant with the API (e.g. by providing a validator for requests)? Alternatively, is there some sort of formal certification mechanism for showing compliance being met as an outcome of a validation performed by the data provider.

Annex B - SQuaRE: ISO 25010 Model

For evaluating the quality of a software product the ISO 25010 Standard on Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) is well established. The quality is defined as “the degree to which the system satisfies the stated and implied needs of its various stakeholders, and thus provides value.” (from ISO 25010). However this intends to be applied to software components, it can be reused to evaluate APIs as well. SQuaRE does not define specific properties, concrete concepts or implementations. It defines a set of high level characteristics software systems (and also APIs) should fulfil. We'll use these characteristics and set them in the context of the API evaluation while our main evaluation metric is the “Five Level Open Data API evaluation model”. In contrast to the abstract ISO standard, the five level model provides specific properties against which the APIs can be evaluated. Since SQuaRE defines high level characteristics, without specific implementations, this offers the possibility to take those characteristics and map those to the properties, defined in the Five Level model. On the one hand this offers to understand the bigger context of the property. On the other hand we can prove that the list of properties covers all quality characteristics.

The ISO 25010 splits the quality characteristics into two groups, which we apply to our three different perspectives (API development, deployment, use). Those two groups are:

— **Quality in use model:** These characteristics relate to the interaction with a system.

We evaluate the interaction with the API in the *API use* perspective.

— **Product quality model:** These characteristics relate to the impact the system/software has on stakeholders.

The product quality model relates to the implementation of the API standard and is considered in the *API deploy* perspective.

Classifying the *API development* perspective in context is not obvious. It clear that the result of the API development should keep the product quality model in mind. Anyhow, the characteristics of the quality in use model can be applied, since similar there are aspects on the API provider (development) and consumer (use) side.

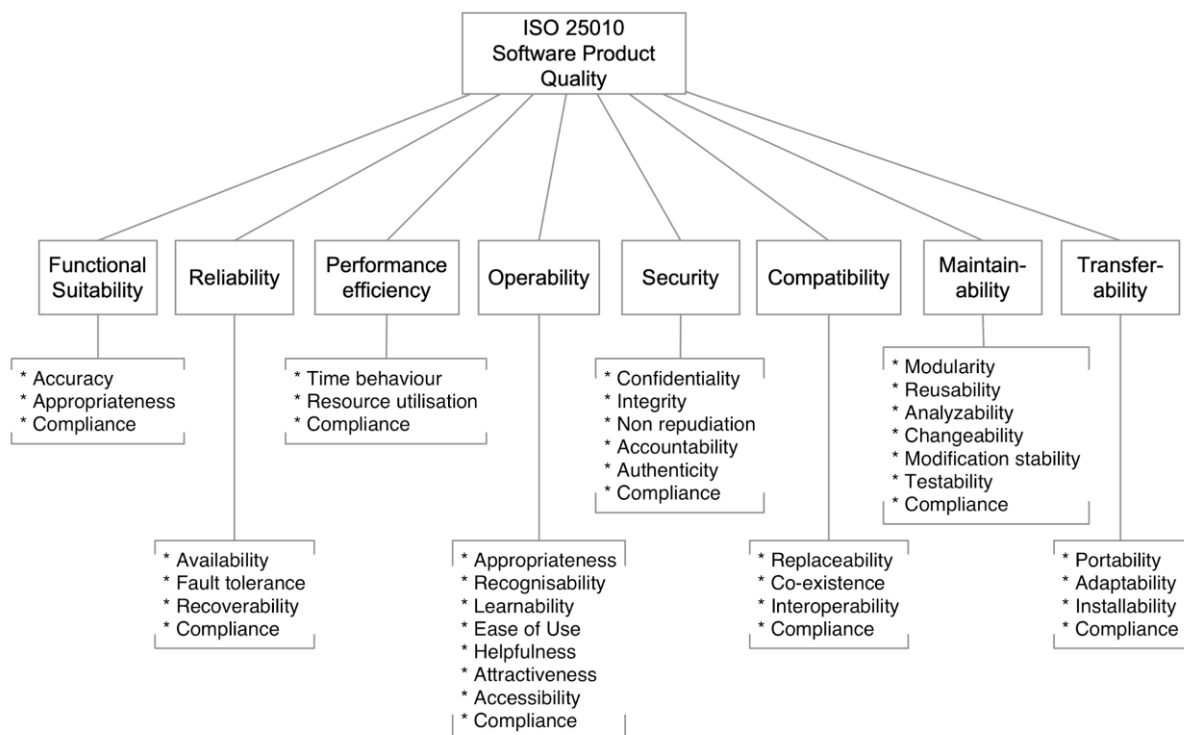
In this report we omit a detailed explanation of the characteristics. Our focus is to set the SQuaRE characteristics in the context of the five level model, to prove the completeness of our evaluation criteria. For a complete overview and definition of the characteristics, we refer to the corresponding ISO document.

The **quality in use model** is split into five characteristics, with partly additional sub-characteristics:

1. Effectiveness
2. Efficiency
3. Satisfaction
 - (a) Usefulness
 - (b) Trust
 - (c) Pleasure
 - (d) Comport
4. Freedom from risk
 - (a) Economic risk mitigation
 - (b) Health and safety risk mitigation
 - (c) Environmental risk mitigation
5. Context coverage
 - (a) Context completeness
 - (b) Flexibility

Some of those characteristics (Effectiveness, Efficiency, Satisfaction) can be directly applied to APIs, whereas it's not that obvious for *Freedom from risk* and *Context coverage*. Anyhow *Freedom from risk* is important when implementing/using an API in a slightly altered version: "Is there a legal risk in using the API? Is there a technical risk in using the API, e.g. the used concepts are outdated? Is there a risk that further developments are discontinued?". The context coverage aims at the targeted use case: "Can the use-case be implemented using the API? Is the API flexible enough?"

The **product quality model** is summarized in the following picture:



Taken from: <https://jaxenter.de/req4arcs-qualitaet-faellt-nicht-vom-himmel-86493>

A crucial point in the API Deployment perspective is the implementation serving the API. Since this is a software application, the product quality model can be directly applied.

| Criteria | ISO | |
|---------------------------|--|----------------------------|
| | Product quality model (API Deployment) | Quality in Use (API Usage) |
| Level 1: All find | | |
| Single Entry Point | Usability | Effectiveness, Efficiency |
| Documentation | Usability | Effectiveness, Efficiency |
| Example Requests | Usability | Effectiveness, Efficiency |
| Example Data | Usability | Effectiveness, Efficiency |
| Level 2: All use | | |
| JSON or XML | Usability, Maintainability, Portability | Efficiency |
| Data License | Compatibility (legal) | Freedom from risk |
| Terms of Use | Compatibility (legal) | Freedom from risk |
| Embedded Metadata | Usability, Maintainability, Portability | Effectiveness, Efficiency |
| Authentication | Security | Freedom from risk |
| * API Standardization | Usability, Maintainability, Portability | Freedom from risk |
| * Suitability | Compatibility (*technical) | Context coverage |
| Level 3: All trust | | |
| * Query and Analytics API | Functional Suitability, Performance efficiency | Effectiveness, Efficiency |
| Error Handling | Usability, Reliability | Freedom from risk |
| * Performance and Cache | Performance efficiency | Effectiveness, Efficiency |
| Background Support | Reliability | Freedom from risk |
| * Availability | Reliability | Freedom from risk |
| * API Validation | Reliability | Freedom from risk |
| Level 4: All involved | | |
| SDK Availability | Functional Suitability, Usability | Satisfaction, Efficiency |
| Code Examples | Usability | Satisfaction, Efficiency |
| Community | Usability | Satisfaction, Efficiency |

| | | |
|-----------------------------|---|--------------------------|
| Playground | Usability | Satisfaction, Efficiency |
| Linked Documentation | Usability | Satisfaction, Efficiency |
| * API Evolution | Usability | Freedom from risk |
| Level 5: All develop | | |
| Code Visible, Portability | Reliability, Maintainability | Freedom from risk |
| Bug Tracker | Usability, Reliability | Freedom from risk |
| API License/reuse | Compatibility (legal), Reliability | Freedom from risk |
| Development Roadmap | Reliability, Portability | Freedom from risk |
| * Linked Data Ready | Usability, Maintainability, Portability | Freedom from risk |
| * Test Framework available | Usability, Reliability, Maintainability | Freedom from risk |

Table 6: shows how the criteria in the Five Level evaluation model relate to the characteristics defined by ISO 25010. A comparison shows that by adding the "Suitability"-criteria, all aspects of the ISO standard are covered by our Five Level model as well.

Annex C Mapping INSPIRE AQD to SensorThings API

Austrian Inspire Air Quality Data

The alignment between the data being harvested from the Austrian INSPIRE air quality services and SensorThings API was quite simple due to the similarities in the underlying data models, as is shown in the following tables. In addition, we made use of the mapping between INSPIRE and SensorThings available from the publication [Extending INSPIRE to the Internet of Things through SensorThings API - doi:10.3390/geosciences8060221](https://doi.org/10.3390/geosciences8060221)

The necessary information for the description of the observed properties was taken directly from the corresponding EEA Vocabulary.

| STA Class | STA Attribute | AQD Field |
|--------------------|------------------------|---|
| DATASTREAMS | | |
| DATASTREAMS | ID | /wfs:FeatureCollection/wfs:member[1]/aqd:AQD_SamplingPoint/@gml:id |
| DATASTREAMS | PHENOMENON_TIME_START | /wfs:FeatureCollection/wfs:member[91]/aqd:AQD_SamplingPoint/ef:operationalActivityPeriod[3]/ef:OperationalActivityPeriod/ef:activityTime/gml:TimePeriod/gml:beginPosition |
| DATASTREAMS | PHENOMENON_TIME_END | /wfs:FeatureCollection/wfs:member[91]/aqd:AQD_SamplingPoint/ef:operationalActivityPeriod[3]/ef:OperationalActivityPeriod/ef:activityTime/gml:TimePeriod/gml:endPosition |
| DATASTREAMS | PROPERTIES/processType | "http://inspire.ec.europa.eu/codeList/ProcessTypeValue/process" |

| STA Class | STA Attribute | AQD Field |
|---------------------|-------------------------|---|
| DATASTREAMS | PROPERTIES/resultNature | "http://inspire.ec.europa.eu/codeList/ResultNatureValue/primary" |
| OBSERVATIONS | FEATURE_ID | /wfs:FeatureCollection/wfs:member[1]/aqd:AQD_SamplingPoint/ef:observingCapability/ef:ObservingCapability/ef:featureOfInterest/@xlink:href |
| DATASTREAMS | OBS_PROPERTY_ID | /wfs:FeatureCollection/wfs:member[1]/aqd:AQD_SamplingPoint/ef:observingCapability/ef:ObservingCapability/ef:observedProperty/@xlink:href |
| DATASTREAMS | SENSOR_ID | /wfs:FeatureCollection/wfs:member[1]/aqd:AQD_SamplingPoint/ef:observingCapability/ef:ObservingCapability/ef:procedure/@xlink:href |
| DATASTREAMS | PROPERTIES/metadata | "http://luft.umweltbundesamt.at/inspire/wfs?service=WFS&version=2.0.0&request=GetFeature&typeName=aqd:AQD_SamplingPoint" |
| DATASTREAMS | THING_ID | /wfs:FeatureCollection/wfs:member/aqd:AQD_SamplingPoint/ef:broader/@xlink:href |

Table 7: STA Datastream Mapping Austrian Air Quality

| STA-Class | STA-Attribute | AQD Field |
|-----------|-----------------------------|---|
| SENSORS | ID | /wfs:FeatureCollection/wfs:member[1]/aqd:AQD_SamplingPointProcess/@gml:id |
| SENSORS | PROPERTIES/namespace | /wfs:FeatureCollection/wfs:member[1]/aqd:AQD_SamplingPointProcess/ompr:inspireId/base:Identifier/base:namespace |
| SENSORS | NAME | /wfs:FeatureCollection/wfs:member[1]/aqd:AQD_SamplingPointProcess/aqd:measurementEquipment/aqd:MeasurementEquipment/aqd:equipment/@xlink:href |
| SENSORS | PROPERTIES/measurementtype | /wfs:FeatureCollection/wfs:member[1]/aqd:AQD_SamplingPointProcess/aqd:measurementType/@xlink:href |
| SENSORS | PROPERTIES/method | /wfs:FeatureCollection/wfs:member[1]/aqd:AQD_SamplingPointProcess/aqd:measurementMethod/aqd:MeasurementMethod/aqd:measurementMethod/@xlink:href |
| SENSORS | DESCRIPTION | /wfs:FeatureCollection/wfs:member[1]/aqd:AQD_SamplingPointProcess/aqd:equivalenceDemonstration/aqd:EquivalenceDemonstration/aqd:demonstrationReport |
| SENSORS | PROPERTIES/responsibleParty | |

| STA-Class | STA-Attribute | AQD Field |
|-----------|---|--|
| SENSORS | PROPERTIES/responsibleParty/individualName | /wfs:FeatureCollection/wfs:member[1]/aqd:AQD_SamplingPointProcess/ompr:responsibleParty/base2:RelatedParty/base2:individualName/gmd:LocalisedCharacterString |
| SENSORS | PROPERTIES/responsibleParty/organisationName | /wfs:FeatureCollection/wfs:member[1]/aqd:AQD_SamplingPointProcess/ompr:responsibleParty/base2:RelatedParty/base2:organisationName/gmd:LocalisedCharacterString |
| SENSORS | PROPERTIES/responsibleParty/adminUnit | /wfs:FeatureCollection/wfs:member[1]/aqd:AQD_SamplingPointProcess/ompr:responsibleParty/base2:RelatedParty/base2:contact/base2:Contact/base2:address/ad:AddressRepresentation/ad:adminUnit/gn:GeographicalName/gn:spelling/gn:SpellingOfName/gn:text |
| SENSORS | PROPERTIES/responsibleParty/locatorDesignator | /wfs:FeatureCollection/wfs:member[1]/aqd:AQD_SamplingPointProcess/ompr:responsibleParty/base2:RelatedParty/base2:contact/base2:Contact/base2:address/ad:AddressRepresentation/ad:locatorDesignator |
| SENSORS | PROPERTIES/responsibleParty/postCode | /wfs:FeatureCollection/wfs:member[1]/aqd:AQD_SamplingPointProcess/ompr:responsibleParty/base2:RelatedParty/base2:contact/base2:Contact/base2:address/ad:AddressRepresentation/ad:postCode |
| SENSORS | PROPERTIES/responsibleParty/electronicMailAddress | /wfs:FeatureCollection/wfs:member[1]/aqd:AQD_SamplingPointProcess/ompr:responsibleParty/base2:RelatedParty/base2:contact/base2:Contact/base2:electronicMailAddress |
| SENSORS | PROPERTIES/responsibleParty/telephoneVoice | /wfs:FeatureCollection/wfs:member[1]/aqd:AQD_SamplingPointProcess/ompr:responsibleParty/base2:RelatedParty/base2:contact/base2:Contact/base2:telephoneVoice |
| SENSORS | PROPERTIES/responsibleParty/website | /wfs:FeatureCollection/wfs:member[1]/aqd:AQD_SamplingPointProcess/ompr:responsibleParty/base2:RelatedParty/base2:contact/base2:Contact/base2:website |
| SENSORS | PROPERTIES/metadata | "http://luft.umweltbundesamt.at/inspire/wfs?service=WFS&version=2.0.0&request=GetFeature&typeName=aqd:AQD_SamplingPointProcess" |

Table 8: STA Sensor Mapping Austrian Air Quality

| STA-Class | STA-Attribute | AQD Field |
|--------------|-------------------------|---|
| FEATUR ES | ID | /wfs:FeatureCollection/wfs:member[1]/aqd:AQD_Sample/@gml:id |
| FEATUR ES | FEATURE | |
| FEATUR ES | GEOM | |
| FEATUR ES | GEOM | /wfs:FeatureCollection/wfs:member[1]/aqd:AQD_Sample/sams:shape/gml:Point/@sr sName |
| FEATUR ES | GEOM | /wfs:FeatureCollection/wfs:member[1]/aqd:AQD_Sample/sams:shape/gml:Point/gml: pos |
| FEATUR ES | PROPERTIES/m etadata | "http://luft.umweltbundesamt.at/inspire/wfs?service=WFS&version=2.0.0&request= GetFeature&typeName=aqd:AQD_Sample" |

Table 9: STA FoI Mapping Austrian Air Quality

| STA-Class | STA-Attribute | AQD Field |
|------------|----------------------------------|---|
| THIN GS | ID | /wfs:FeatureCollection/wfs:member[1]/aqd:AQD_Station/@gml:id |
| THIN GS | PROPERTIES/nam espace | /wfs:FeatureCollection/wfs:member[1]/aqd:AQD_Station/ef:inspireId/base:Identifier /base:namespace |
| THIN GS | NAME | /wfs:FeatureCollection/wfs:member[1]/aqd:AQD_Station/ef:name |
| THIN GS | PROPERTIES/med iaMonitored | "http://inspire.ec.europa.eu/codelist/MediaValue/air" |
| THIN GS | PROPERTIES/mea surementRegime | "http://inspire.ec.europa.eu/codelist/MeasurementRegimeValue/continuousDataColl ection" |
| THIN GS | PROPERTIES/mob ile | "FALSE" |
| THIN GS | PROPERTIES/begi nTime | /wfs:FeatureCollection/wfs:member[1]/aqd:AQD_Station/ef:operationalActivityPeri od/ef:OperationalActivityPeriod/ef:activityTime/gml:TimePeriod/gml:beginPosition |
| THIN GS | PROPERTIES/end Time | /wfs:FeatureCollection/wfs:member[1]/aqd:AQD_Station/ef:operationalActivityPeri od/ef:OperationalActivityPeriod/ef:activityTime/gml:TimePeriod/gml:endPosition |
| THIN GS | DESCRIPTION | |

| STA-Class | STA-Attribute | AQD Field |
|-----------|---------------------|--|
| THINGS | PROPERTIES/metadata | http://luft.umweltbundesamt.at/inspire/wfs?service=WFS&version=2.0.0&request=GetFeature&typeName=aqd:AQD_Station |
| LOCATIONS | GEOM | /wfs:FeatureCollection/wfs:member[1]/aqd:AQD_Station/ef:geometry/gml:Point/@srsDimension |
| LOCATIONS | GEOM | /wfs:FeatureCollection/wfs:member[1]/aqd:AQD_Station/ef:geometry/gml:Point/@srsName |
| LOCATIONS | GEOM | /wfs:FeatureCollection/wfs:member[1]/aqd:AQD_Station/ef:geometry/gml:Point/gml:pos |

Table 10: STA Thing/Location Mapping Austrian Air Quality

| STA-Class | STA-Attribute | AQD Field |
|--------------|-----------------|--|
| DATASTREAMS | SENSOR_ID | /sos:GetObservationResponse/sos:observationData/om:OM_Observation/om:procedure/@xlink:href |
| DATASTREAMS | OBS_PROPERTY_ID | /sos:GetObservationResponse/sos:observationData/om:OM_Observation/om:observedProperty/@xlink:href |
| OBSERVATIONS | FEATURE_ID | /sos:GetObservationResponse/sos:observationData/om:OM_Observation/om:featureOfInterest/@xlink:href |
| OBSERVATIONS | RESULT_NUMBER | /sos:GetObservationResponse/sos:observationData/om:OM_Observation/om:result/swe:DataArray |
| OBSERVATIONS | DATASTREAM | /sos:GetObservationResponse/sos:observationData/om:OM_Observation/om:parameter[2]/om:NamedValue/om:value |

Table 11: STA Observation Mapping Austrian Air Quality

EEA Air Quality Data

The EEA makes CSV files with the air quality measurements available. To find the CSV files, one has to make a GET request to the following URL:

https://fme.discomap.eea.europa.eu/fmedatastreaming/AirQualityDownload/AQData_Extract.fmw?CountryCode=<CountryCode>&Pollutant=<PollutantCode>&Year_from=<StartYear>&Year_to=<EndYear>&Source=All&Output=TEXT&TimeCoverage=Year

The result of this request is a text file with one URL per line. These URLs point to the CSV files containing the actual observations, and are of the following scheme.

Source URL:
<https://ereporting.blob.core.windows.net/downloadservice/<CountyCode> <PollutantCode> <stationId> <Year> > timeseries.csv>

| Column: | Mapped to: |
|-------------------|--|
| Concentration | Observation/result |
| DatetimeBegin | Observation/phenomenonTime (start of interval) |
| DatetimeEnd | Observation/phenomenonTime (end of interval) |
| AirQualityStation | Thing/name |
| AirPollutant | ObservedProperty/name |

Table 12: STA Observation Mapping EEA Air Quality

To find the correct Datastream, the following query is sent to the SensorThings service:

`v1.1/Datastreams?$filter=Thing/properties/localId eq '<AirQualityStation>' and ObservedProperty/name eq '<AirPollutant>'`

If this query does not return a Datastream, a new Datastream is created, and possibly a new Thing, Location, Sensor and ObservedProperty. The data required for these entities is read from the station metadata file.

The station metadata can be downloaded for all stations in Europe in one file, containing 58842 records.

Source URL: http://discomap.eea.europa.eu/map/fme/metadata/PanEuropean_metadata.csv

| Column: | Mapped to: |
|-----------------------|---|
| AirQualityStation | Thing, Location name properties/localId |
| Countrycode | Thing, Location, Datastream, Sensor properties/countryCode |
| Namespace | Thing, Location, Datastream, Sensor properties/namespace |
| AirQualityStationArea | Thing, Location properties/mediaMonitored |
| ObservationDateBegin | Thing properties/beginTime |
| ObservationDateEnd | Thing properties/endTime |
| SamplingProces | Sensor name properties/localId |

| Column: | Mapped to: | |
|---|-------------------------------------|----------------------------------|
| MeasurementEquipment | Sensor | metadata |
| AirPollutantCode | ObservedProperty | definition properties/localId |
| SamplingPoint | Datastream | name properties/localId |
| <Source URL> | Thing, Location, Datastream, Sensor | properties/metadata |
| http://dd.eionet.europa.eu | Thing, Location, Datastream, Sensor | properties/owner |
| http://inspire.ec.europa.eu/codelist/MediaValue/air | Thing | properties/mediaMonitored |

Table 13: STA Observation Metadata Mapping EEA Air Quality

GETTING IN TOUCH WITH THE EU

In person

All over the European Union there are hundreds of Europe Direct information centres. You can find the address of the centre nearest you at: https://europa.eu/european-union/contact_en

On the phone or by email

Europe Direct is a service that answers your questions about the European Union. You can contact this service:

- by freephone: 00 800 6 7 8 9 10 11 (certain operators may charge for these calls),
- at the following standard number: +32 22999696, or
- by electronic mail via: https://europa.eu/european-union/contact_en

FINDING INFORMATION ABOUT THE EU

Online

Information about the European Union in all the official languages of the EU is available on the Europa website at: https://europa.eu/european-union/index_en

EU publications

You can download or order free and priced EU publications from EU Bookshop at: <https://publications.europa.eu/en/publications>. Multiple copies of free publications may be obtained by contacting Europe Direct or your local information centre (see https://europa.eu/european-union/contact_en).

The European Commission's science and knowledge service

Joint Research Centre

JRC Mission

As the science and knowledge service of the European Commission, the Joint Research Centre's mission is to support EU policies with independent evidence throughout the whole policy cycle.



EU Science Hub

ec.europa.eu/jrc



@EU_ScienceHub



EU Science Hub - Joint Research Centre



EU Science, Research and Innovation



EU Science Hub



Publications Office
of the European Union

doi:10.2760/00811

ISBN 978-92-76-49644-1