API4INSPIRE - Inception Report

Version 1.2
2020-01-20

Contents

API4INSPIRE - Inception Report

This report gives an overview of the partners involved in the project, the projected timeline for completing the project, the approach the consortium plans on taking to complete the project and how external data providers and their data sets will be involved in the project.

Partner Overview

The evaluation of both the OGC API - Features as well as the SensorThings API pertaining to their applicability within the INSPIRE context requires a wide range of skills and expertise. This led us to the consortium approach between Fraunhofer IOSB, DataCove e.U. and GeoSolutions, allowing us to provide best of breed experience both for the APIs being evaluated, as well as pertaining to the fulfilment of INSPIRE requirements.
The following list contains the main contributors to the project, employed at the respective partner.

Fraunhofer IOSB

Fraunhofer IOSB has long experience pertaining to Sensor Web. They have developed the Open Source FROST-Server, the official reference implementation of the SensorThings API, and currently hold the co-chair of the OGC SensorThings SWG.

- Jürgen Moßgraber <juergen.mossgraber@iosb.fraunhofer.de>
- Hylke van der Schaaf <hylke.vanderschaaf@iosb.fraunhofer.de>
- Philipp Hertweck <philipp.hertweck@iosb.fraunhofer.de>

DataCove e.U.

Personnel from DataCove e.U. has been involved in the INSPIRE Implementation process since the onset. They facilitate the INSPIRE Forum pertaining to sensor data and currently hold the co-chair of the OGC O&M SWG.

- Katharina Schleidt <kathi@datacove.eu>
- Melinda Klein <Melinda.Christine@gmx.at>

GeoSolutions

GeoSolutions has long been the driving force behind GeoServer, that has become increasingly relevant for provision of INSPIRE WFS services. This implementation has now been extended to OGC API - Features.

- Andrea Aime <andrea.aime@geo-solutions.it>
- Nuno Oliveira <nuno.oliveira@geo-solutions.it>
- Simone Giannecchini <simone.giannecchini@geo-solutions.it>

Timeline

To allow for everyone to have a proper Christmas holiday, the date of M2 is moved two weeks back, and the date of M4 is moved one week back.

2019-12-17 (M1)	D0	Project inception report
2020-01-31 (M2)	D1	Methodology for evaluation of standard-based APIs TR Section 1 (providers) TR Section 2 (users) of the Technical Report (TR)
2020-03-24 (M4)	D2	Deployment strategies for standard-based APIs TR Section 3
2020-05-17 (M6)	D3	Technical documentation, source code and working prototype Information system and documentation
2020-06-17 (M7)	D4	Assessment of the impact of APIs TR Section 4
2020-09-17 (M10)	D5	Conclusions and recommendations for MS authorities TR Section 5 - Summary document and package of all training materials
2020-10-17 (M11)	TR	Final Technical Report

Approach for completing project tasks

This chapter lists, for each task, the initial ideas on how to approach the task. Since the project will use an agile approach, and since the details of the available data sets and deployment conditions are not known yet, deviations are likely to occur.
A GitHub[1] has been created for this project as a central access point for information for all involved parties. The GitHub Wiki will also serve as the core of supporting materials to be provided under D5.

D1. Methodology for evaluation of standard-based APIs

The foundation of the evaluation strategy will be laid down by the selected assessment criteria. For this purpose, we intend to utilize the core framework specified in the “Simple five level Open Data API evaluation model” proposed by Jarkko Moilanen. This model abstracts Tim Berners-Lee's 5-star deployment scheme for Open Data towards API evaluation as follows:

- **Level 1: All find**
- **Level 2: All use**
- **Level 3: All trust**
- **Level 4: All involved**
- **Level 5: All develop**

An interesting alternative approach would be based on the ISO 25010 Standard on Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. However, the SQuaRE methodology is tailored for the evaluation of the quality of the underlying software implementation, whereas the scope of this project pertains to the API specification itself. Thus the SQuaRE methodology will be analysed for concepts missing within the 5-Level API Evaluation structure, and this structure extended accordingly.

Further evaluation criteria will be gleaned from similar initiatives (OGC API and Environmental Data Retrieval API Hackathons) and the team’s experience; in addition, facets pertaining to usability as well as administrative and infrastructural aspects including those pertaining to leveraging existing investments in infrastructure and applications will be included in the final list of evaluation criteria. All criteria identified will be graded in terms of severity of potential impact and ramifications.

While the evaluation pertains to three different aspects pertaining to API development, deployment and use, we believe that the criteria identified can be reused across all of these aspects, whereby the different aspects may be slightly adjusted to better pertain to different facets of these criteria.

In order to ascertain how the usage of emerging APIs can be utilized to leverage existing data provider investments while reducing the resources required for both data provision and usage, cost-benefit analyses will be performed. These analyses will take slightly different forms dependent on the aspect

[1] https://github.com/DataCoveEU/API4INSPIRE

being analysed; separate analyses will be performed for development, deployment and use. In these variants, we will utilize the 5 Levels of our evaluation criteria as guidance; only be maintaining alignment between these analysis aspects will it be possible to understand how additional costs accrued in the development or deployment processes can be countermanded by benefits ensuing through simpler usage of provided data.

Whereas modern APIs greatly contribute to ease of uptake and use of available data, the effort required for the chore of providing existing data, be it through configuration of an existing server or the development of a dedicated application, tends to be directly proportional to the complexity of the desired resulting data model. In order to understand and quantify possible reductions of effort required for the provision of an API on existing datasets, we will explore reductions in scope of provided data in comparison to the full breadth required by INSPIRE by utilizing Simple Feature options proposed under the MIF action 2017.2 on alternative encodings. Benefits identified through such simplification options will be integrated into the cost-benefit analyses described above.

The evaluation process will be conducted through 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. Within the evaluation methodology description we will determine which evaluation methods are best suited to the appraisal of which of the evaluation criteria. Metrics will also be defined in order to aggregate the various responses to clear indicators pertaining to the individual criteria. All outputs of the evaluation process will be thoroughly analysed, insights gained and lessons learned documented, recommendations formulated.

In the matrix below, we provide a first overview of which evaluation methods we see as suitable for which evaluation criteria

Method:	API Development		API Deployment			API Use				
Criteria:	Heuristic Evaluation	Interview	Heuristic Evaluation	Questionnaire	Inter-view	Heuristic Evaluation	Peer Review	Hacka-thon	Question-naire	Inter-view
Level 1: All find										
Single Entry Point	X		X			X				
Documentation	X	X	X	X	X	X			X	X
Example Requests	X	X	X	X	X	X			X	X
Example Data	X	X	X	X	X	X			X	X
Level 2: All use										
JSON or XML	X		X			X				X
Data License	X	X	X	X		X	X			
Terms of Use	X	X	X	X		X				
Embedded Metadata	X		X	X		X	X		X	
Authentication	X		X	X	X	X	X		X	
API Standardization	X		X		X	X			X	
Level 3: All trust										
Analytics API	X		X			X	X		X	
Error Handling	X	X	X		X	X	X		X	
Queries and Cache			X	X		X	X		X	
Background Support	X	X	X			X			X	X
Availability			X	X		X			X	
API Validation	X	X	X	X		X			X	
Level 4: All involved										
SDK Availability	X					X			X	
Code Examples	X	X				X			X	X
Community	X	X	X	X	X	X			X	X
Playground	X	X	X	X	X	X			X	
Linked Documentation			X	X	X	X			X	
API Evolution	X		X		X	X			X	
Level 5: All develop										
Code Visible			X	X	X	X		X	X	X
Bug Tracker	X	X	X	X	X	X		X	X	
API License/reuse			X	X	X	X		X	X	X
Development Roadmap	X	X	X	X	X	X		X	X	
Test Framework available		X	X	X	X	X		X	X	X

D2. Deployment strategies for standard-based APIs

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
6. Execute and measure the specified queries
7. Optimize queries where necessary

We will build on this successful strategy, making minor modifications as required.

Different strategies will also be 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) will be 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, we will also be leveraging data sources not yet available. 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.

For the steps #5 to #7, commonly utilized open source software will be 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.

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 Agency AFB, 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.
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 focus on these aspects as much as possible and label implementation specific aspects as such.

Deployment Overview

The goal of this exercise is to illustrate how the emerging APIs being evaluated in this project can be easily deployed across a wide array of different data sources and environments. In order to show diversity in deployment options, our deployment strategies cover the following constellations:

- Existing WFS2 deployments migration:
  - Utilizing ld-proxy for on-the-fly conversion
  - Deployments using GeoServer with the OGC API extension
- New deployments of OGC Features - API
  - Using a new GeoServer deployment
  - Deploying a standalone new development of the OGC API Features API
- New deployment of SensorThings API using FROST
  - Using the dedicated FROST DB
  - By accessing existing data sources via SQL Views

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

GeoServer Deployment

GeoServer is a free 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 the 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. This includes the most common data stores such as Shapefile, GeoPackage, PostgreSQL and GeoTIFF as well the most common used OGC services such as 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 directive specifications.

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. 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.

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.

OGC API – Features

OGC is defining a new set of services to replace the currently 20 years old OWS ones. 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.

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 correspond to compliance level SF-0 of OGC's Simple Features profile, while the latter correspond 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 our 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 require 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.

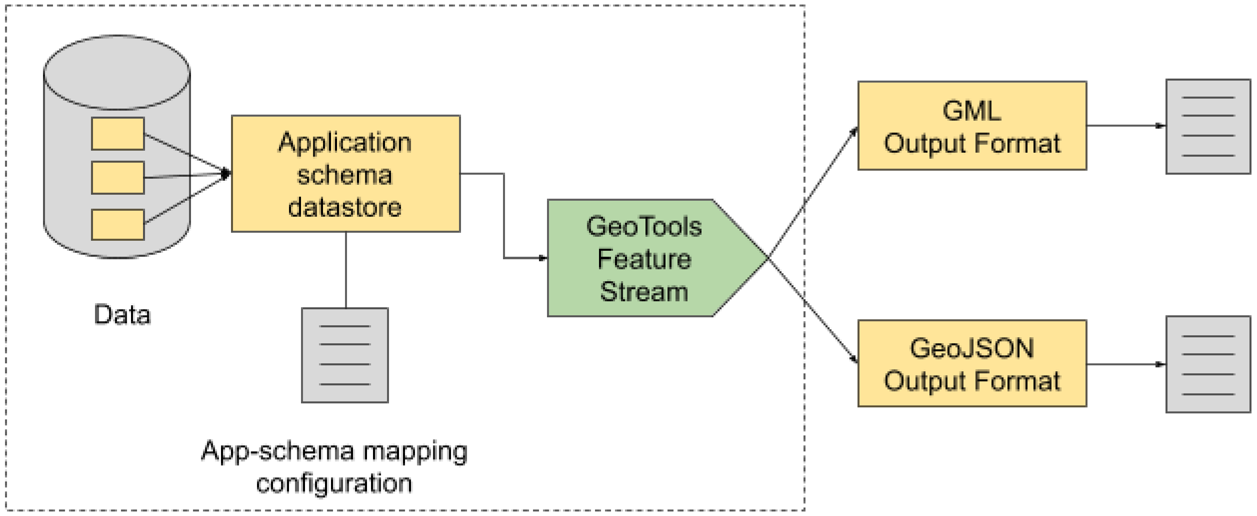

Figure 2: Overview of App-Schema architecture, the mappings between the relational database data model and the target GML will be used by App-Schema to produce complex features as well to interpret queries and translate them to SQL queries or in memory filters. The GML and GeoJSON encoders will consume the complex features stream a produce the respective outputs.

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.

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 modeling 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 this new guidelines without being tied to the complexity of old standards. 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.

OGC API – Features Development

In order to ascertain the complexity of developing an OGC API 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 vocational informatics school in Vienna. A team of interested students has been created, and implementation work initiated.

This simple OGC API demonstrator will be created in an iterative manner. In a first 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 selected table. In further iterations, it will be possible to configure these names as required.

For encoding, this implementation will rely on simple feature encoding standards and only provide JSON encoded data. At present, this implementation will be limited to PostGIS databases, whereby we are considering providing additional support for SpatiaLite DBs. Austo 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.

LD-Proxy

In order to leverage existing WFS2 implementation not stemming from GeoServer deployments, or also 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.

FROST Deployment

For data storage, FROST uses the PostgreSQL database server with PostGIS geospatial extensions, 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"[2]. These extended properties sections are available in the 1.1 version of the STA standard, which FROST already implements.

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. 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.

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

D3. Technical documentation, source code and working prototype Information system and documentation

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. Expected documentation to come out of this project is a set of examples on how to map a custom, internal database to database views that match the FROST database tables.

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 necessaries 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. Is worth mentioning that independent of the selected output format, GML or GeoJSON, the App-Schema mappings will always require a target GML schema. Breaking that limitation would require a significant amount of work and is out of the scope of this project.

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.

API Deployments

As we have a wide spectrum of data providers involved in this work, different deployment strategies will be necessary dependant on the level of maturity of the individual partners. Data sources will be selected based on initial use cases defined to show the integration of data from multiple sources; these use cases will be selected to highlight cross theme and cross border challenges.

In the cases where there is already an existing INSPIRE compliant WFS2 deployment in operation, all harmonization and alignment work has already been performed, and the API deployment consists of merely adding the GeoSolutions OGC API extension to the GeoServer Deployment.

In most cases, INSPIRE compliant have not yet been deployed. Thus, before we can begin with the evaluation of the OGC API - Features and STA, we must first align the existing data sources with the INSPIRE Models and create the necessary configuration files in cooperation with the data providers.

All API deployments will be described and documented; in lieu of a fully populated metadata catalogue, all information together with the API endpoints will be made available via a GitHub repository. This portfolio of APIs will serve as the basis for the assessment of the APIs to be performed under D4.

D4. Assessment of the impact of APIs Section 4 of TR

The evaluation will be an ongoing iterative process within the project, as some evaluation methods can be done in the early stages, while for others such as the showcasing of these APIs within a hackathon setting, full deployment must first be achieved.

The metrics specified within the evaluation methodology will be applied to the following approaches:

1. Preliminary heuristic analysis: based on the project participants experience in the development of geospatial and environmental applications together with the evaluation criteria specified, project partners will critically analyse the APIs, highlighting initial issues and providing recommendations for amelioration.
2. Peer reviews: following the procedure laid down in D1, peer reviews will be held with experts from the project team, data provider staff as well as interested stakeholders. Based on an example, the reviewers will be guided through the usage of the API; in the subsequent mediated discussion, insights will be distilled. As the peer review methodology is exceptionally sensitive to design issues, evaluation criteria will be selected accordingly, and used to guide the individual reviews.
3. Interviews with data provider staff: In targeted interviews, we will engage data provider staff involved in the deployment and uptake of the new OGC APIs. This will provide insights into how to best leverage existing infrastructure investments, as well as documenting benefits and deficits of the APIs. These interviews will also allow us to tailor the guidance and training materials to the data providers requirements. This approach will also be utilized to glean information from the developers building the standalone OGC API Implementation.
4. Hackathon Interviews/Questionnaires: interviews will be conducted with selected hackathon participants, allowing for in-depth insights and feedback on their experience with the APIs; we will request those participants not being interviewed to provide feedback via questionnaire.

The output of these assessment methodologies will serve as input for the defined metrics, supporting the creation of guidance materials.

A workshop has been submitted to the 2020 INSPIRE Conference in Dubrovnik, whereby we will integrate Hackathon concepts into our APIthon. Additional evaluation events will be integrated within various related initiatives, be they national or thematic hackathons such as the regular Hackathons organized by the City of Hamburg or Munich, or other related events such as FOSS4G.

Further details on the assessment methodology and events will be provided within D1 Methodology for evaluation of standard-based APIs once these events have been confirmed.

D5. Conclusions and recommendations for MS authorities

Based on the outcomes of this evaluation, such additional support materials as has been deemed relevant shall be created. These materials will take the form of 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. During the project, we will be collecting and structuring all materials identified as relevant for the development, deployment and use of OGC API on the project GitHub; whereby the content can later be transferred to the platform of the customer’s choice. 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.

Involvement of the data providers

The data providers will be connected to the project using an agile approach during all phases of the project. During the early phases of the project, use cases will be defined based on the available data sets and experience the data providers have with their current data customers. Priority will be given to those use cases and data sets that allow both APIs to be used in the same use case, where possible from different providers.
An example of such overlap between data providers is data for the Rhine valley in the border region between France and Baden Württemberg (Germany). This area is covered by the data sets from BRGM and AFB from France, and by the data sets from the LUBW in Germany.
Another important condition in the selection of data sets and use cases is that for each data provider at least one data set is chosen for each of the APIs, so that both APIs are deployed at each data provider, unless the data provider has no data sets that can be exposed using an API.
After the use cases and data sets have been chosen, the data will need to be mapped to the relevant API, taking into account the INSPIRE requirements. The data providers will be involved to ensure a correct understanding of the data required to make this mapping.
Once the mapping in complete, the deployment of the services on the hosting infrastructure of the data provider needs to take place. Naturally this requires extensive involvement of the data provider.
In the next phase of the project the impact of the APIs needs to be evaluated. This will include interviews with staff of the data providers and hackathons involving end-users of the data providers.
Finally the findings of the project need to be documented and the results distributed. The target audience of the documentation includes the data providers and thus the data providers will be involved in the creation of the documentation to ensure it meets their needs. The data providers will also take part in distributing the project results through their existing communication channels.

Initial list of Data Sets

The following list gives an overview of the different data sets that the data providers can supply.
Note: in addition to the software systems listed in the table below, the simple OGC API implementation being developed will be tested on various of these datasets.

BRGM

Data set	**API**	**SW**	**Type**
Geology	OGC API - Features	GeoServer	Static
Mineral Resources	OGC API - Features	GeoServer	Static
Soil	OGC API - Features	GeoServer	Static
Hydrography	OGC API - Features	GeoServer	Static
Environmental Monitoring Facilities	STA	FROST	Dynamic
Area Management, … Reporting	OGC API - Features	GeoServer	Static

Zones			
Natural Risk Zones	OGC API - Features	GeoServer	Static

AFB

Data Set	**API**	**SW**	**Type**
Geographical Grid Systems	OGC API – Features	GeoServer	Static
Hydrography	OGC API - Features	GeoServer	Static
Transport networks	OGC API – Features	GeoServer	Static
Land Cover	OGC API – Features	GeoServer	Static
Area Management, … Reporting Zones	OGC API – Features	GeoServer	Static
Environmental monitoring Facilities	STA	FROST	Dynamic
Oceanographic geographical features	STA	FROST	Dynamic
Sea Regions	OGC API – Features	GeoServer	Static
Species Distribution	OGC API - Features	GeoServer	Static

ZAMG:

Data Set	**API**	**SW**	**Type**
Meteorological features (STA, dynamic)	STA	FROST	Dynamic
Existing INSPIRE EF	WFS2 → OGC API – Features	GeoServer	Static

Austro Control:

Data set	**API**	**SW**	**Type**
Air Transport Networks	WFS2 → OGC API – Features	GeoServer	Static
Air Transport Networks (simple features)	OGC API – Features	Standalone Dev	Static

Hamburg:

Data Set	**API**	**SW**	**Type**
(Electro) Mobility	STA	FROST	Dynamic
Traffic control	STA	FROST	Dynamic
Environmental data	STA	FROST	Dynamic
Urban data	OGC API – Features	GeoServer	Static

Landesanstalt für Umwelt Baden-Württemberg:

Data Set	**API**	**SW**	**Type**

Ground water	STA	FROST	dynamic, very low frequency
Surface water	STA	FROST	dynamic, very low frequency