geostandards-INDG20-60

Table of Contents

1. Summary . . . 2
2. Introduction . . . 2
   2.1. Context . . . 2
   2.2. Fundamental questions . . . 3
3. Methodology . . . 3
4. From OWS to web architecture . . . 6
   4.1. OWS Common . . . 6
   4.2. OGC API Common . . . 7
5. Data Encoding Standards . . . 8
6. Data Access Standards . . . 9
   6.1. Introduction . . . 9
   6.2. Web Feature Service (WFS) . . . 9
   6.3. OGC API - Features . . . 11
   6.4. Web Coverage Service (WCS) . . . 13
   6.5. OGC API Coverages . . . 14
   6.6. OGC API - Environmental Data Retrieval (EDR) . . . 16
   6.7. Sensor Observation Service (SOS) . . . 18
   6.8. OGC SensorThings API . . . 20
7. Processing Standards . . . 21
   7.1. Introduction . . . 21
   7.2. Web Processing Service (WPS) . . . 22
   7.3. OGC API - Processes . . . 23
8. Visualization standards . . . 25
   8.1. Web Map Service (WMS) . . . 25
   8.2. OGC API Maps . . . 26
   8.3. Web Map Tile Service (WMTS) . . . 27
   8.4. OGC API Tiles . . . 29
   8.5. Styled Layer Descriptor (SLD) . . . 30
   8.6. Symbology Encoding (SE) . . . 31
   8.7. OGC API Styles . . . 32
   8.8. OGC Symbology Conceptual Model . . . 34
9. Metadata & Catalogue Services Standards . . . 36
   9.1. Introduction . . . 36
   9.2. Catalog Services for the web (CSW) . . . 36
   9.3. Operations, Ressources . . . 36
   9.4. OGC API Records . . . 37

9.5. Operations, Ressources . . . 38
9.6. Spatio Temporal Asset Catalog . . . 38
9.7. Operations, Ressources . . . 39
10. Uses cases . . . 39
10.1. Sensor Data Standards . . . 39
10.2. From data discovery & access to portrayal . . . 42
10.3. Earth Observation data . . . 45
11. Conclusion . . . 46
12. Appendix A: bibliographical references . . . 47
13. Appendix B: conference papers . . . 47
14. Appendix C: follow up meetings . . . 47
14.1. 25-05-2022 . . . 47
14.2. 31-05-2022 . . . 48
14.3. 20-06-2022 . . . 49
14.4. 11-07-2022 . . . 51

1. Summary

2. Introduction

Recently, in 2022, the Open Geospatial Consortium (OGC) and the Open Source Geospatial Foundation (OSGeo) have renewed their Memorandum of Understanding. While the initial agreement dates from 2008, this renewal is especially motivated by the current focus on the OGC APIs and the will to empower developers (even non-geospatial) to leverage location in their development. Indeed, the OGC has initiated the creation of a new generation of standards based on the OpenAPI specifications to facilitate the integration of geospatial data in modern web applications and systems.

Underpinning the OGC Standards Roadmap cite:[ogc-roadmap], the development of all these standards represents a significant number of activities carried out by various OGC working groups, Testbeds and pilots from the OGC Innovation Program cite:[ogc-ip]. Some standards have been approved, many are still under development and it is therefore not always easy to follow their progress. Indeed, while some geodata infrastructures involving national entities as the(e.g. Meteorological Service of Canada cite:[eccc-msc] in Canada) are already deploying this new generation of standards, some initiatives run a phase of experimentation such as for instance the Geonovum OGC API Testbed Platform cite:[apitestbed].

2.1. Context

In Switzerland the exchange and publication of geographic information is governed by the Federal Act on Geoinformation (GeoIA) cite:[geoia]. The GeoIA is accompanied by two regulations:

- GeoIV/OGeo (Federal Council Ordinance) cite:[ogeo]

- GeoIV/OGeo-swisstopo (Ordinance specific for the Federal Office of Topography swisstopo) cite:[ogeo-swisstopo]

The GeoIV/OGeo contains the fundamental provisions, while the GeoIV/OGeo-swisstopo regulates the technical details, which can be modified by the competent federal office (swisstopo), with the participation of the cantons and after consultation with different partner organisations:

- eCH, a national association for the establishment of e-government standards
- SOGI/OSIG, a national association for geoinformation
- KGK-CGC, an organization of the Swiss Cantons in the field of geoinformation management

The GeoIV/OGeo-swisstopo specifies criteria such as the geodetic reference system, the description language for modeling geospatial data, metadata standards for geoinformation and minimal requirements for geodata.

In its 7th article: ”Minimal requirements for geoservices”, the GeoIV/OGeo-swisstopo refers to the e-Government standard eCH-0056 entitle Application Profile for Geoservices cite:[eCH-0056], which defines the implementation of basic geoservices by means of a set of additional guidelines and recommendations which make the services suitable for practical use. In addition, there are other regulations, such as metadata models cite:[gm03] and protocols for the exchange of base geodata under federal law.

As the eCH-0056 document currently refers to the WxS family of OGC specifications (WMS, WMTS, WFS, WCS, CSW, SE, SLD, SOS), our research aims to assess the maturity and the benefits of the new OGC APIs in the perspective of a revision of its legal framework.

2.2. Fundamental questions

From a practical perspective we can ask the question how organizations and institutions anticipate to leverage this new generation of standards in order to deploy a geospatial data infrastructure. This issue is at the core of this project that seeks to address it by running an OGC API Testbed platform with a special focus to the Swiss context. This project is embedded in the NGDI Resources program related to the Swiss Geoinformation Strategy cite:[indg] with the purpose to contribute to the upcoming revision of Swiss e-government standards regarding geoinformation such as the eCH-0056 cite:[eCH-0056] standard. This project is about a research carried out by academic partners (HEIG-VD, SUPSI, UNIGE) as well as the Swiss Office for Topography Swisstopo.

3. Methodology

Given the above general purpose, the intent is to develop knowledge and practice of these OGC API standards so as, for instance, to update guidelines of the instructions stated by eCH- 0056. More generally, how can the advances coming out from the OGC standardization process be integrated into the Swiss standardization process and even vice versa. The idea in this project is to setup a Testbed Platform to connect these two processes as illustrated by figure 1 by considering issues related to how requirements defined by stakeholders are handled by standardization works both sides. By running at each iteration a showcase on specific use cases with a set of standards to challenge, the Testbed Platform brings results as update guidelines (e.g. for eCH working groups in

the field of geoinformation) and feedbacks (e.g. OGC Change Requests). In addition, such a vision can provide an opportunity for dissemination and training material creation (e.g. tutorials, workshops).

Several activities are to be carried out to feed and run such a platform, among them: requirements analysis with stakeholders, identification of underlying standards to challenge, subscription/following in the relevant standards working groups, definition of experimental cases and evaluation criteria, deployment and maintenance of software to run showcases, documentation and dissemination.

For this project, to be seen as a first iteration, the expected outcomes include both quantitative and qualitative results that will be compiled into practical e-government guidelines for the implementation of standards from the OGC API family. The selected mainstream topic for the showcase and the underlying experimental cases is about climate change. While not yet connected in a complex pilot study, each case represents some of the required components from sensors to portrayal. The study is organized in three parts to challenge standards with software related to: (1) sensors data (2) data discovery, access and portrayal (3) earth observation data.

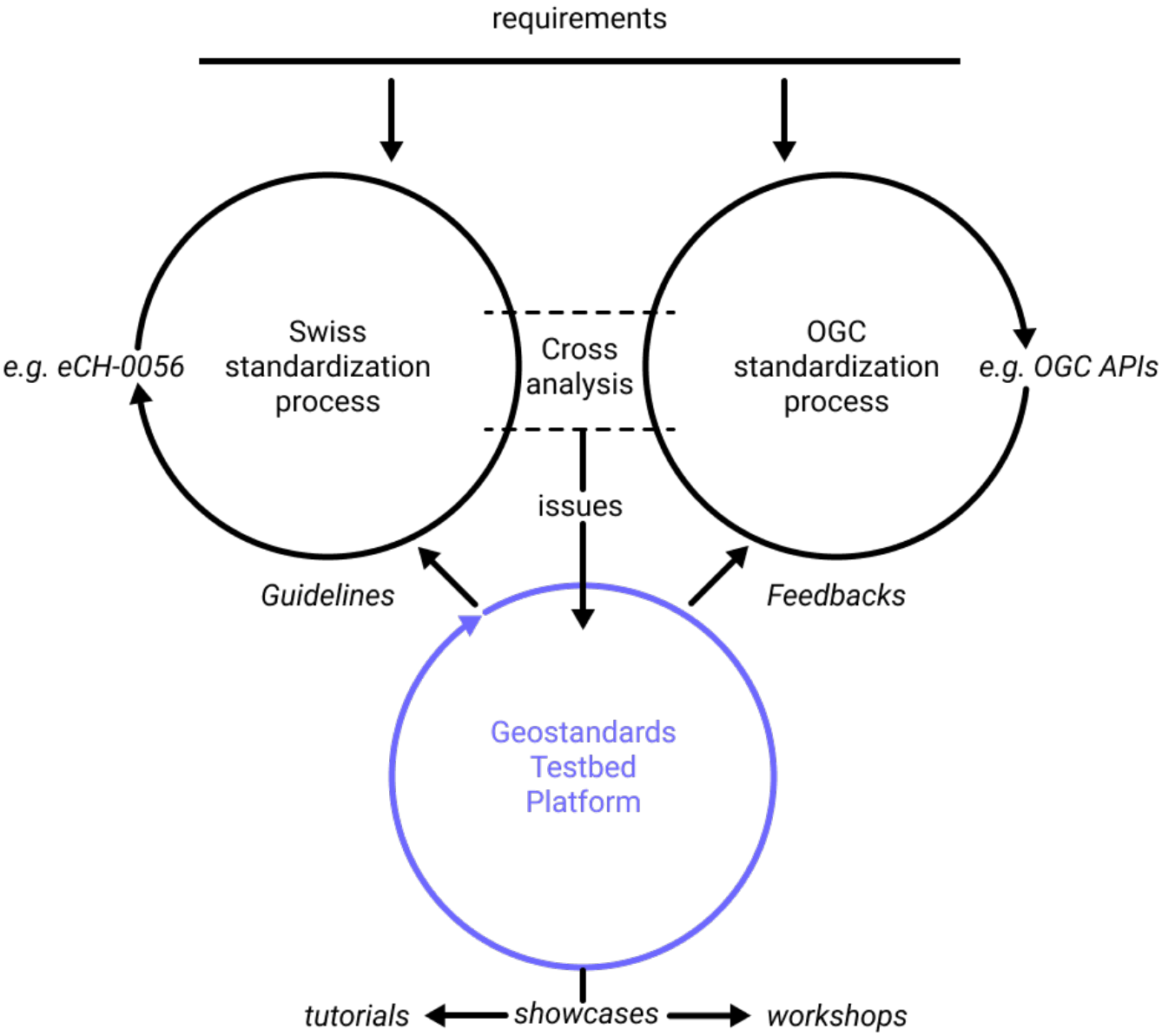

Figure 1. Proposed vision to connect Swiss and OGC standardization processes through a Testbed Platform

In term of software, the idea is to offer a unique project entry point for the discovery, experimentation and analysis of the new OGC APIs. To develop the knowledge and practice required, openness is not only related to the standards evaluated, but also required for the software aspects of the Testbed platform. That's why we identified the Geonovum OGC API Testbed tooling stack cite:[apitestbed] as a good base. It includes already open sourceopen-source packages and provides a detailed documentation available on GitHub. We deployed a project instance at https://ogc.heig-vd.ch/ (figure 2) which still needs to get populated with the underlying data and configured to serve everything required to run the experimental cases, from server to client sides. This solution will also be used to create tutorials in the style of the documentation proposed by Meteo Canada cite:[eccc-msc].

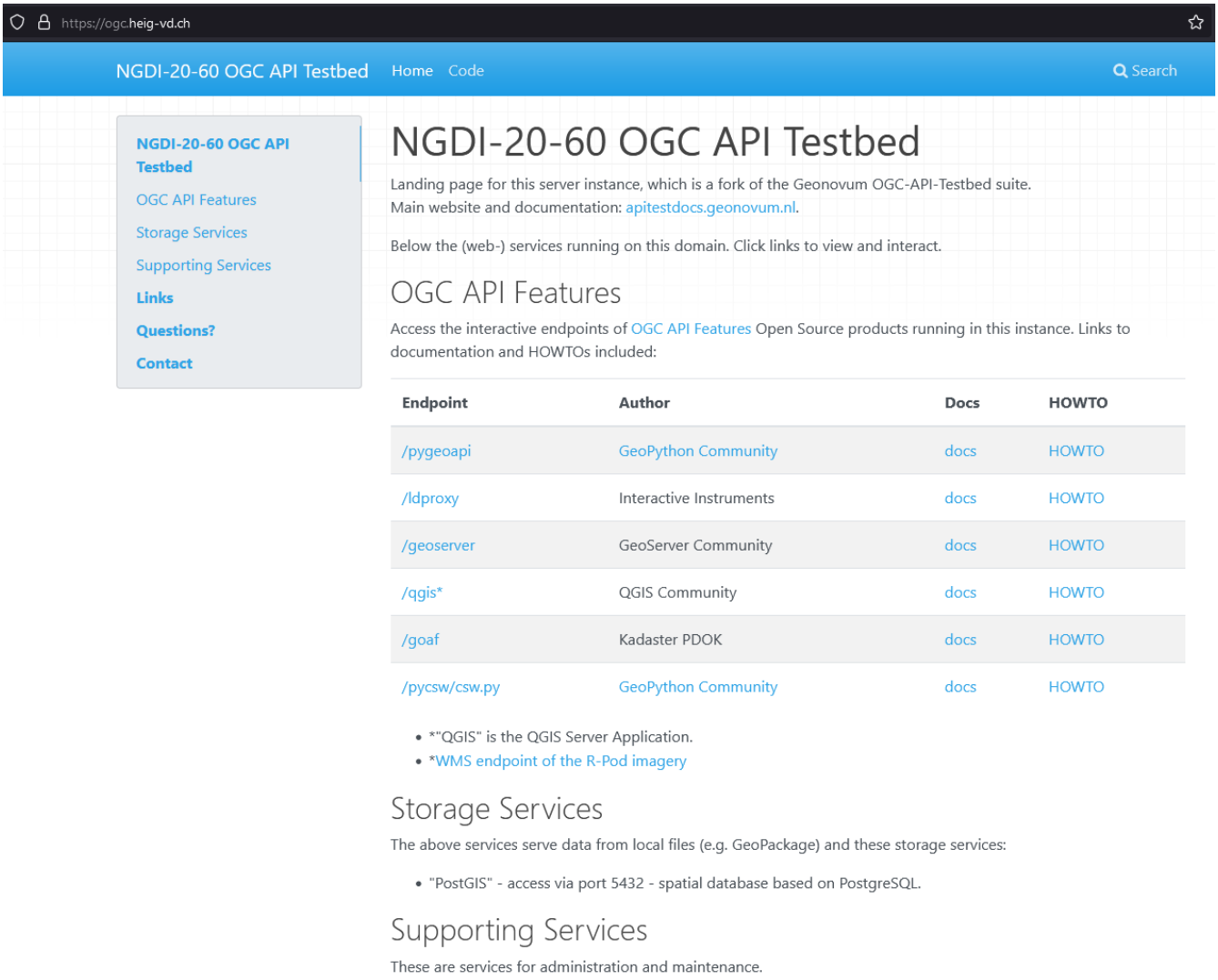

Figure 2. Landing page of the Testbed platform

This solution is expected to bring good visibility toward all actors concerned by the Swiss standardization process. It is to raise interest from government stakeholders, companies and universities, technical people or not, as a community in synergy when considering the advances at the OGC.

To test the experimental cases, FOSS4G implementations are deployed, especially server-side software with:

- FROST : focusing on sensor observation standards

- Geoserver : quite active in the implementation of a wide set of OGC API standards
- ldproxy : focusing on OGC API Features as an adapter sitting in front of existing WFS services
- pygeoapi : active in the implementation with the largest set of OGC API standards of the panel
- QGIS Server : quite active community in Switzerland, implementing also an OGC API Features client

The choice of these solutions was made to cover all the standards to be challenged in this project as well as with the variety of programming languages of the most common OSGeo software, the ease of deployment and the quality of their documentation. Table 1 has been built by going over the OGC API landing page of each, in conformance with the Landing Page Requirements Class of the specification OGC API - Common - Part 1: Core (OGC, 2021a).

Table 1. Implementations according to conformance declarations (status as of September 2022)

	FROST	Geoserver	ldproxy	pygeoapi	QGIS Server
SensorThingsAPI (OGC:18-062r2)	✓				
Features (OGC:17-069r3)		✓	✓	✓	✓
Maps (OGC:20-058)		✓			
Styles (OGC:20-009)		✓			
Tiles (OGC:20-057)		✓			
Coverages (OGC:19-087, draft)		✓		✓	
DGGS (draft)		✓			
EDR (OGC:19-086r4)				✓	
Processes (OGC:18-062r2)				✓	
Records (OGC:20-004)				✓	

4. From OWS to web architecture

4.1. OWS Common

The OGC Web Services Common (OWS Common) provides guidance to OGC members who are

developing OGC interface implementation standards. The purpose of OWS Common is to maintain consistency among OGC standards. OWS Common provides rules for specifying some of the parameters and data structures used in operation requests and responses. Each interface standard details additional aspects, including specifying all additional parameters and data structures needed in all operation requests and responses. The following is a list of some common aspects covered by OWS Common document:

- GetCapabilities operation (request, parameters, response)
- Exception reports
- Operations parameters:
  - Bounding box
  - Coordinate reference systems
  - Format
  - Data descriptions
  - Mulitlingual text encodings
- Operation request and response encoding (HTTP GET and HTTP POST)
- Guidance for OWS Implementation Specifications

4.2. OGC API Common

OGC API - Common is a common framework used in all OGC API's. OGC API - Common is build on the legacy of OWS Common and provides a common framework for all OGC APIs with the following code functionalities:

- based on OpenAPI 3.0
- HTML and JSON as the dominant encodings, alternative encodings are possible
- common and shared endpoints such as:
  - `/` (landing page)
  - `/conformance`
  - `/openapi`
  - `/collections`
  - `/collections/foo`
- common aspects such as pagination, links between resources, basic filtering, query parameters (`bbox`, `datetime`, etc.)

OGC API - Common allows for specification developers to focus on the key functionality of a given API (i.e. data access, etc.) while using common constructs. This harmonizes OGC API standards and enables deeper integration with less code. This also allows for OGC API client software to be more streamlined.

The `/conformance` endpoint indicates which standards and extensions are supported by a deployment of OGC API.

4.2.1. OGC API building blocks

The OGC API approach allows for modularity and "profiling" of APIs depending on your requirements. This means you can mix and match OGC APIs together.

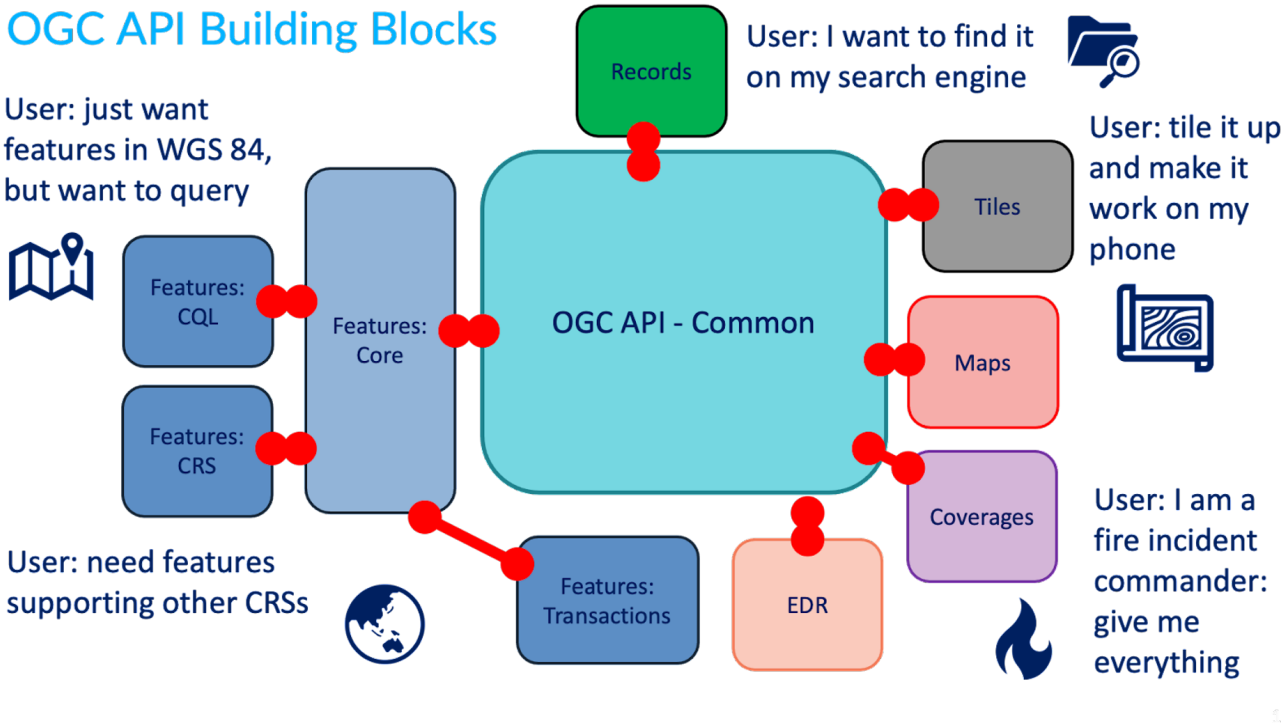

Figure 3. OGC API building blocks

You can read more about this topic in the building blocks website.

5. Data Encoding Standards

The datasets exchanged by geospatial software products have to be structured in a way that products by other vendors can read the data and interpret the information it contains. Without a well defined structure it would be almost impossible to enable an application to read and interpret data without losing or missing information. The need to define the structure and organisation of data formats and messaging protocols applies equally to binary formats such as GeoTIFF as it does to human-readable formats such as XML.

To support the ability of applications to exchange messaging and data that contains geospatial information, the OGC has developed a series of data model and encoding standards. These standards provide the rules for structuring and organising geospatial data for use in a given context. In some cases the standards present conceptual models, however in other cases the standards describe logical models that are intended to be implemented using one or more encoding formats. Many of the standards also provide rules on how to encode information according the logical models and in a particular format.

Let's take, for example, the Geography Markup Language (GML) standard. This standard describes a structure and rules for encoding geographic information in XML. The standard describes how instances of concepts such as features, geometry, coordinate reference systems, points, linestrings, polygons and several others should be written in XML. The rules are supported by an XML Schema

Definition (XSD) file to allow validation of datasets.

In storage systems with limited capacity and networks with limited bandwidth, an alternative encoding standard such as OGC GeoPackage tends to be preferred. The GeoPackage standard describes how to structure and organise of geospatial data when storing it in an SQLite database. SQLite is a popular embedded database that is often found on devices with Size, Weight and Power (SWaP) constraints, such as tablets, smartphones and Internet-Of-Things (IoT) microcomputers.

By clearly and formally specifying encoding rules, the GML, GeoPackage and other OGC encoding standards make it possible to establish tests for validating whether a dataset complies to the requirements stated in the standards.

More detailed information can be obtained via the following links:

- GeoPackage
- Geography Markup Language (GML)

Source: Open Geosptial Consortium

6. Data Access Standards

6.1. Introduction

This section describes the main standards for accessing data through web geoservices.

6.2. Web Feature Service (WFS)

6.2.1. Introduction

The OGC Web Feature Service (WFS) Interface Standard defines a set of interfaces for accessing geographic information at the feature and feature property level over the Internet. A feature is an abstraction of real world phenomena, that is it is a representation of anything that can be found in the world. The attributes or characteristics of a geographic feature are referred to as feature properties. WFS offer the means to retrieve or query geographic features in a manner independent of the underlying data stores they publish. Where a WFS is authorized to do so, the service can also update or delete geographic features. An instance of a WFS is also able to store queries in order to enable client applications to retrieve or execute the queries at a later point in time.

Source : Open Geospatial Consortium

6.2.2. Background

History

WFS version 1.0.0 in May 2002, followed by version 1.1.0 in May 2005,and version 2.0.0 in November 2010. Version 2.0.0 is the basis of ISO 19142. The OGC released WFS version 2.0.2 in July 2014.

Versions

Version 2.0.2 - 09-025r2 is the current latest.

Implementations

Implementations can be found on the OGC Compliance Database.

Endpoints on the NGDI-20-60 OGC API Testbed Platform

The Web Feature Service (WFS) standard has not been integrated into the Testbed platform.

6.2.3. Operations, Resources

Table 2. Overview of the WFS operations

Operation	Description
`GetCapabilities`	Retrieves metadata about the service, including supported operations and parameters, and a list of the available feature types.
`DescribeFeatureType`	Returns a description of the structure of feature types and feature properties offered or accepted by an instance of a WFS.
`GetFeature`	Returns a selection of feature instances from a data store published through the WFS.
`ListStoredQueries`	Returns a list of the queries that have been stored inside the WFS instance.
`DescribeStoredQueries`	Returns a description of the queries that have been stored inside the WFS instance.
`GetPropertyValue` (optional)	Retrieves the value of a feature property or part of the value of a complex feature property for a set of feature instances
`GetFeatureWithLock` (optional)	Serves a similar function to a GetFeature request but with the additional ability to lock a feature, presumably for subsequent updating or changes.
`LockFeature` (optional)	Locks a set of feature instances such that no other operations may modify the data while the lock is in place.
`Transaction` (optional)	Allows the feature instances and their properties to modified or deleted.
`CreateStoredQuery` (optional)	Creates and stores a query that can be rapidly and easily triggered by a client at a later point in time.

`DropStoredQuery` (optional)	Deletes a previously stored query from the server.

6.2.4. Relation to other OGC Standards

- OGC API - Features

6.2.5. Relation with eCH-0056

The WFS standard is currently mentioned in the following sections:

3.4, 5, 6.4, 6.6 (`QUAL-03`), 6.9, 6.9.2 (`WFS-01`,`WFS-02`,`WFS-03`,`WFS-04`,`WFS-05`,`WFS-06`,`WFS-07`,`WFS-08`,`WFS-09`,`WFS-10`,`WFS-11`,`WFS-12`), 6.11, appendix B (`Mapping: WFS metadata elements → GM03`, `Mapping: mandatory GM03 metadata elements → WMS, WMTS, WFS, WCS et CSW`) & E

of the eCH-0056 document.

6.2.6. Proposed modifications to eCH-0056

- Refer to version 2.0.2 of the WFS standard in the eCH-0056 document.

6.3. OGC API - Features

6.3.1. Introduction

OGC API - Features is a multi-part standard that offers the capability to create, modify, and query spatial data on the Web and specifies requirements and recommendations for APIs that want to follow a standard way of sharing feature data. The Core part of the standard is called **OGC API - Features - Part 1: Core**. The Core part of the specification describes the mandatory capabilities that every implementing service has to support and is restricted to read-access to spatial data. Additional capabilities that address specific needs will be specified in additional parts. Envisaged future capabilities include, for example, support for creating and modifying data, more complex data models, richer queries, and additional coordinate reference systems.

Source : Open Geospatial Consortium

6.3.2. Background

History

The OGC API Features no longer includes the notion of version. Standards are now named by their functionality.

OGC API - Features - Part 1: Core & OGC API - Features - Part 2: Coordinate Reference Systems by Reference were both released in April 2022.

2 other versions:

- OGC API - Features - Part 3: Filtering and the Common Query Language (CQL)

- OGC API - Features - Part 4: Create, Replace, Update and Delete

are currently in draft status.

Versions

OGC API - Features - Part 2: Coordinate Reference Systems by Reference - 18-058r1 is the current latest version

Implementations

Implementations can be found on the OGC Compliance Database.

Endpoints on the NGDI-20-60 OGC API Testbed Platform

Examples of implementations can be found on the NGDI-20-60 OGC API Testbed Platform

6.3.3. Operations, Resources

Table 3. Overview of OGC API Features resources, applicable HTTP methods and links to the OGC documentation

Resource	Path	HTTP method	Document reference
Landing page	`/`	GET	7.2 API landing page
Conformance declaration	`/conformance`	GET	7.4 Declaration of conformance classes
Feature collections	`/collections`	GET	7.13 Feature collections
Feature collection	`/collections/{collectionId}`	GET	7.14 Feature collection
Features	`/collections/{collectionId}/items`	GET	7.15 Features
Feature	`/collections/{collectionId}/items/{featureId}`	GET	7.16 Feature

6.3.4. Relation to other OGC Standards

- Web Feature Service (WFS)

6.3.5. Relation with eCH-0056

The OGC API - Features standard is not mentioned in the eCH-0056 document.

6.3.6. Proposed modifications to eCH-0056

- Integrate the OGC API Features in the same sections as the WFS standard in the eCH-0056 document according to its extensions.

- Add a new section to the eCH-0056 document that describes the OGC API Features standard.
- Update the structure of the eCH-0056 document according to the versions and functionalities of the OGC API Features.

6.4. Web Coverage Service (WCS)

6.4.1. Introduction

The OGC Web Coverage Service (WCS) supports electronic retrieval of geospatial data as “coverages.” Coverages are digital geospatial information representing space/time-varying phenomena, specifically spatio-temporal regular and irregular grids, point clouds, and general meshes. WCS offer the means to retrieve or query geographic coverages in a manner independent of the format in which the data is stored.

Source: Open Gepspatial Consortium

6.4.2. Background

History

- WCS 2.0.0 was approved in October 2010
- WCS 2.0.1 was approved in July 2012
- WCS 2.1 was approved in June 2018

Versions

2.1 is the current latest version

Implementations

Implementations can be found at the OGC database.

Endpoints on the NGDI-20-60 OGC API Testbed Platform

The Web Coverage Service (WCS) standard has not been integrated into the Testbed platform.

6.4.3. Operations, Resources

Table 4. Overview of the WCS Operations

Operation	Description
`GetCapabilities`	Retrieves metadata about the service, including supported operations and parameters, and a list of the available coverages.
`DescribeCoverage`	Returns a description of the coverage offered or accepted by an instance of a WCS.
`GetCoverage`	Returns a selection of coverage instances from a data store published through the WCS.

6.4.4. Relation to other OGC Standards

- [OGC API - Environmental Data Retrieval]

6.4.5. Relation with eCH-0056

The Web Coverage Service (WCS) standard is currently mentioned in the following sections:

5, 6.4, 6.9, 6.9.3 (`WCS-01`,`WCS-02`,`WCS-03`,`WCS-04`,`WCS-05`,`WCS-06`), appendix A (`Mapping: WCS metadata elements → GM03`, `Mapping: mandatory GM03 metadata elements → WMS, WMTS, WFS, WCS et CSW`) & E

of the eCH-0056 document.

6.4.6. Proposed modifications to eCH-0056

- Update the reference to the WCS standard in the eCH-0056 document according to its latest version.

6.5. OGC API Coverages

6.5.1. Introduction

The OGC API - Coverages draft specification defines a Web API for accessing coverages that are modeled according to the Coverage Implementation Schema (CIS) 1.1. Coverages are represented by some binary or ASCII serialization, specified by some data (encoding) format. Arguably the most popular type of coverage is that of a gridded coverage. Gridded coverages have a grid as their domain set describing the direct positions in multi-dimensional coordinate space, depending on the type of grid. Satellite imagery is typically modeled as a gridded coverage, for example.

6.5.2. Background

History

This standard is in a draft state.

Versions

Version 0.0.6 - 19-087 is in a draft state.

Implementations

Not applicable.

Endpoints on the NGDI-20-60 OGC API Testbed Platform

Examples of implementations can be found on the NGDI-20-60 OGC API Testbed Platform

6.5.3. Operations, Resources

Table 5. Coverage API Resources

Resource URI	Description
{datasetAPI}/	Landing page for this dataset distribution
{datasetAPI}/api	API description (e.g. OpenAPI)
{datasetAPI}/api	API documentation (optional, e.g. HTML)
{datasetAPI}/conformance	Conformance Classes
{datasetAPI}/collections	The list off all collections available, some or all of which may be accessible using this Coverage API. Each of these collection objects take the same form as that of the collection resource object described immediately below.
{datasetAPI}/collections/{collectionId}	resource corresponding to the collection with the unique identifier `{collectionId}`, which may be accessible as a coverage. The resource will describe key elements such as an `id`, `title`, `description`, available `crs` and `extent` (the coverage envelope) as well as links to resources pertaining to this collection. For coverages, it will either embed or link to a CIS JSON encoding of both the range type and the domain set. It is comparable to a WCS ***DescribeCoverage*** response, with the exception that the range type and domain set may have to be retrieved separately by following a link to accommodate the case where they may be considerably large, and the domain set may support query parameters to subset it.
Coverages	
{datasetAPI}/collections/{collectionId}/coverage	returns the coverage including all of its components (domain set, range type, range set and metadata), to the extent that it is supported by the selected representation (see format encoding for ways to retrieve in specific formats). It is comparable to a WCS ***GetCoverage*** response.
{datasetAPI}/collections/{collectionId}/coverage/rangeset	if supported by the service and by the selected representation, returns only the coverage's range set, i.e., the actual values in the selected representation without any accompanying description or extra information.
{datasetAPI}/collections/{collectionId}/coverage/rangetype	if available separately from the collection resource, returns the coverage's range type information, i.e., a description of the data semantics (their components and data type).
{datasetAPI}/collections/{collectionId}/coverage/domainset	if available separately from the collection resource, returns the coverage's domain set definition (the detailed n-dimensional space covered by the data).

Resource URI	Description
{datasetAPI}/collections/{collectionId}/coverage/metadata	if available, returns the associated coverage metadata as defined by the CIS model, which may be e.g. domain specific metadata.

6.5.4. Relation to other OGC Standards

- Web Coverage Service (WCS)

6.5.5. Relation with eCH-0056

The OGC API - Coverages standard is not mentioned in the eCH-0056 document.

6.5.6. Proposed modifications to eCH-0056

Not applicable as it is not yet a valid standard.

6.6. OGC API - Environmental Data Retrieval (EDR)

6.6.1. Introduction

OGC API - Environmental Data Retrieval is a standard that provides a family of lightweight interfaces to access Environmental Data resources. The standard, which is also called the Environmental Data Retrieval (EDR) API, addresses two fundamental operations; discovery and query. Discovery operations allow the API to be interrogated to determine its capabilities and retrieve information (metadata) about this distribution of a resource. This includes the API definition of the server as well as metadata about the Environmental Data resources provided by the server. Query operations allow Environmental Data resources to be retrieved from the underlying data store based upon simple selection criteria, defined by this standard and selected by the client.

Source: Open Gepspatial Consortium

6.6.2. Background

History

Version 1.0.0 was published in 2021.

Versions

Version 1.0.0 - 19-086r5 is the current latest version

Implementations

Implementations can be found on the OGC Product Database.

Endpoints on the NGDI-20-60 OGC API Testbed Platform

Examples of implementations can be found on the NGDI-20-60 OGC API Testbed Platform

6.6.3. Operations, Resources

Table 1 summarizes the access paths and relation types defined in this standard.

Table 6. Environmental Data Retrieval API Paths

Path Template	Relation	Resource
Common		
{root}/	none	Landing page
{root}/api	`service-desc` or `service-doc`	API Description (optional)
{root}/conformance	`conformance`	Conformance Classes
Collections		
{root}/collections	`data`	Metadata describing the collections of data available from this API.
{root}/collections/{collectionId}		Metadata describing the collection of data which has the unique identifier `{collectionId}`
Features		
{root}/collections/{collectionId}/items	`items`	Retrieve metadata about available items
Queries		
{root}/collections/{collectionId}/{queryType}		Retrieve data according to the query pattern
{root}/collections/{collectionId}/instances		Retrieve metadata about instances of a collection
{root}/collections/{collectionId}/instances/{instanceId}		Retrieve metadata from a specific instance of a collection which has the unique identifier `{instanceId}`

Where:

- `{root}` = Base URI for the API server
- `{collectionId}` = an identifier for a specific collection of data
- `{instanceId}` = an identifier for a specific version or instance of a collection of data
- `{queryType}` = an identifier for a specific query pattern to retrieve data from a specific collection of data

6.6.4. Relation to other OGC Standards

- Web Coverage Service (WCS)
- OGC API Coverages
- Sensor Observation Service (SOS)
- [OGC API SensorThings]

6.6.5. Relation with eCH-0056

The OGC API - Environmental Data Retrieval standard is not mentioned in the eCH-0056 document.

6.6.6. Proposed modifications to eCH-0056

- Integrate the OGC API EDR in the same sections as the WCS standard in the eCH-0056 document according to its extensions.
- Add a new section to the eCH-0056 document that describes the OGC API EDR standard.
- Update the structure of the eCH-0056 document according to the versions and functionalities of the OGC API EDR.

6.7. Sensor Observation Service (SOS)

6.7.1. Introduction

The Sensor Observation Service (SOS) defines a web service interface that allows retrieval of observations, sensor metadata and representations of the features from which the observations are taken. The service interface also allows publishing of new observations, as well as registration and removal of sensors.

6.7.2. Background

History

SOS version 2.0 was released in April 2012 SOS version 1.0 was released in October 2007

Versions

Version 2.0 - 12-006 is the current latest.

Implementations

Implementations can be found at the OGC database.

Endpoints on the NGDI-20-60 OGC API Testbed Platform

The Sensor Observation Service (SOS) standard has not been integrated into the Testbed platform.

6.7.3. Operations, Resources

Table 7. Overview of the SOS operations

Operation	Description
`GetCapabilities`	Returns a document that describes the functionality and resources offered by the SOS service that is provided by the server.
`DescribeSensor`	Returns a description of the procedures or sensors associated with an SOS.
`GetObservation`	Returns observation data that has been collected by the procedure or sensor.
`GetFeatureOfInterest (Optional)`	Returns a description of the features of interest for which the SOS offers observations.
`GetObservationById (Optional)`	Allows the client application to retrieve an observation by passing a pointer to that observation.
`InsertSensor (Optional)`	Registers a new sensor system in the SOS.
`DeleteSensor (Optional)`	Deletes a new sensor system from the SOS.
`InsertObservation (Optional)`	Allows client applications to insert new observations for a registered sensor system.
`InsertResultTemplate (Optional)`	Allows client applications to upload a template for result values such that result values that conform to the template can be inserted into the SOS using subsequent calls of the InsertResult operation.
`InsertResult (Optional)`	Allows a client application to insert new observations for a sensor system by inserting only the results of the observations and reusing other metadata provided by a template.
`GetResultTemplate (Optional)`	Returns a result template that describes the exact structure used by a specific procedure or sensor to generate a new observation result.
`GetResult (Optional)`	Allows retrieving just the result values of observations without the entire metadata of the observation.

6.7.4. Relation to other OGC Standards

- [SensorThings API]

6.7.5. Relation with eCH-0056

The Sensor Observation Service (SOS) standard is currently mentioned in the following sections:

5, 6.14 and the appendix E

of the eCH-0056 document.

6.7.6. Proposed modifications to eCH-0056

Add a new section to the eCH-0056 document about the Sensor Observation Service (SOS) standard.

6.8. OGC SensorThings API

6.8.1. Introduction

The Internet of Things (IoT) is a global information infrastructure that enables advanced services by interconnecting both physical and virtual “things” based on existing and evolving interoperable information and communication technologies [ITU-T]. To facilitate geospatial interoperability between devices in the IoT, the OGC has published the OGC SensorThings API.

The OGC SensorThings API is a multi-part standard for an open and geospatial-enabled approach for interconnecting devices, data, and applications of the Internet of Things (IoT). The first part of the standard describes the interface for Sensing. The second part describes the interface for Tasking. The Sensing part standardizes the management and retrieval of observations and metadata from heterogeneous IoT sensor systems. The Tasking part, which is to be developed in the future, is expected to provide a standard way for parameterizing - also called tasking - of IoT devices that can be instructed to carry out observations or perform other functions.

Source: Open Geospatial Consortium

6.8.2. Background

History

SensorThings API Part 1: Sensing version 1.0 was released in 2016 followed by version 1.1 ion 2021.

Versions

Version 1.1 - 18-088 is the latest

Implementations

Implementations are listed on the OGC website.

Endpoints on the NGDI-20-60 OGC API Testbed Platform

Examples of implementations can be found on the NGDI-20-60 OGC API Testbed Platform

6.8.3. Operations, Resources

Table 8. Overview of the SensorThings API ressources

Operation	Description
`Thing`	The OGC SensorThings API follows the ITU-T definition, i.e., with regard to the Internet of Things, a thing is an object of the physical world (physical things) or the information world (virtual things) that is capable of being identified and integrated into communication networks [ITU-T].
`Location`	The Location entity locates the Thing or the Things it associated with. A Thing's Location entity is defined as the last known location of the Thing.
`HistoricalLocation`	A Thing's HistoricalLocation entity set provides the times of the current (i.e., last known) and previous locations of the Thing.
`Datastream`	A Datastream groups a collection of Observations measuring the same ObservedProperty and produced by the same Sensor.
`Sensor`	A Sensor is an instrument that observes a property or phenomenon with the goal of producing an estimate of the value of the property.
`ObservedProperty`	An ObservedProperty specifies the phenomenon of an Observation.
`Observation`	An Observation is the act of measuring or otherwise determining the value of a property.
`FeatureOfInterest`	The phenomenon against which an observation is made is a property of the feature of interest.

6.8.4. Relation to other OGC Standards

- Sensor Observation Service (SOS)

6.8.5. Relation with eCH-0056

The OGC SensorThings API standard is not mentioned in the eCH-0056 document.

6.8.6. Proposed modifications to eCH-0056

- Integrate the OGC SensorThings API in the same sections as the Sensor Observation Service (SOS) standard in the eCH-0056 document according to its extensions.
- Add a new section to the eCH-0056 document that describes the OGC SensorThings API.
- Update the structure of the eCH-0056 document according to the versions and functionalities of the OGC SensorThings API.

7. Processing Standards

7.1. Introduction

This section describes the main standards for accessing processes through web geoservices.

7.2. Web Processing Service (WPS)

7.2.1. Introduction

The OGC Web Processing Service (WPS) Interface Standard specifies a standard interface that provides access to pre-defined processes and provides job control operations that can instantiate, control and monitor processing jobs. In this context, the term process refers to any algorithm, calculation or model that either generates new data or transforms some input data into output data. A WPS enables the execution of computing processes that typically combine raster, vector, and/or coverage data to produce new raster, vector, and/or coverage data. The inputs, processes and outputs offered by a WPS can also be non-spatial.

Source : Open Geospatial Consortium

7.2.2. Background

History

WPS version 1.0.0 was released in June 2007. Version 2.0.1 was released in January 2015.

Versions

Version 2.0.1 - 14-065 is the current latest.

Implementations

Implementations can be found on the OGC Compliance Database.

Endpoints on the NGDI-20-60 OGC API Testbed Platform

The Web Processing Service (WPS) standard has not been integrated into the Testbed platform.

7.2.3. Operations

7.2.4. Relation to other OGC Standards

- OGC API - Processes

7.2.5. Relation with eCH-0056

The Web Processing Service (WPS) standard is not mentioned in the eCH-0056 document.

7.2.6. Proposed modifications to eCH-0056

If WPS should be included in the eCH-0056 document, it would be interesting to use the (more recent) OGC API Processes. The issue will be addressed in the workshops dedicated to the eCH-0056 revision.

7.3. OGC API - Processes

7.3.1. Introduction

The OGC API — Processes standard supports the wrapping of computational tasks into executable processes that can be offered by a server through a Web API and be invoked by a client application. The standard specifies a processing interface to communicate over a RESTful protocol using JavaScript Object Notation (JSON) encodings. The standard leverages concepts from the OGC Web Processing Service (WPS) 2.0 Interface Standard but does not require implementation of a WPS. The Core part of the standard is called OGC API - Processes - Part 1: Core. The Core part of the standard supports the wrapping of computational tasks into executable processes that can be offered by a server through a Web API and be invoked by a client application either synchronously or asynchronously. Examples of computational processes that can be supported by implementations of this specification include raster algebra, geometry buffering, constructive area geometry, routing, imagery analysis and several others.

Source : Open Geospatial Consortium

7.3.2. Background

History

Several of the concepts specified in OGC API - Processes originated in work specifying a RESTful interface for WPS 2.0. From February 2019 onwards, all work relating to a RESTful interface for the WPS 2.0 was changed to focus on OGC API - Processes.

Versions

OGC API - Processes - Part 1: Core - 18-062r2 is the current latest.

Implementations

Implementations can be found here.

Endpoints on the NGDI-20-60 OGC API Testbed Platform

Examples of implementations can be found on the NGDI-20-60 OGC API Testbed Platform

7.3.3. Operations

Table 9. Overview of the OGC API Processes resources

Resource	Path	Purpose
Landing page	/	This is the top-level resource, which serves as an entry point.

Resource	Path	Purpose
Conformance declaration	`/conformance`	This resource presents information about the functionality that is implemented by the server.
API Definition	`/api`	This resource provides metadata about the API itself. Note use of **/api** on the server is optional and the API definition may be hosted on completely separate server.
Process list	`/processes`	Process identifiers, links to process descriptions.
Process description	`/processes/{processID}`	Retrieves a process description.
Process execution	`/processes/{processID}/execution`	Creates and executes a job.
Job status info	`/jobs/{jobID}`	Retrieves information about the status of a job.
Job results	`/jobs/{jobID}/results`	Retrieves the result(s) of a job.
Job list	`/jobs`	Retrieves the list of jobs.
Job Deletion	`/jobs/{jobID}`	Cancels and deletes a job.

7.3.4. Relation to other OGC Standards

- Web Processing Service (WPS)

7.3.5. Relation with eCH-0056

The OGC API - Processes standard is not mentioned in the eCH-0056 document.

7.3.6. Proposed modifications to eCH-0056

- Add a new section to the eCH-0056 document that describes the OGC API - Processes.
- Update the structure of the eCH-0056 document according to the versions and functionalities of the OGC API - Processes.

8. Visualization standards

This section describes the main standards for visualising geodata through web geoservices.

8.1. Web Map Service (WMS)

8.1.1. Introduction

The OGC Web Map Service Interface Standard (WMS) defines a set of interfaces for requesting map images over the Internet. WMS makes it easy for a client to request images on demand changing parameters such as size and coordinate reference systems. A WMS server (i.e. a service that implements the WMS standard) provides information about what maps the service provides, and it produces a map and answers queries about content the content of a map.

Source : Open Geospatial Consortium

8.1.2. Background

History

The version 1.0.0 was released in April 2000, followed by version 1.1.0 in June 2001,and version 1.1.1 in January 2002. The OGC released WMS version 1.3.0 in January 2004.

Versions

Version 1.3 - 06-042 is the current latest.

Implementations

Implementations can be found on the OGC Compliance Database.

Endpoints on the NGDI-20-60 OGC API Testbed Platform

The Web Map Service (WMS) standard has not been integrated into the Testbed platform.

8.1.3. Operations

Table 10. Overview of the WMS operations

Operation	Description

GetCapabilities	Retrieves metadata about the service, including supported operations and parameters, and a list of the available layers.
GetMap	Retrieves a map image for a specified area and content.
GetFeatureInfo (optional)	Retrieves the underlying data, including geometry and attribute values, for a pixel location on a map.
DescribeLayer (optional)	Indicates the WFS or WCS to retrieve additional information about the layer.
GetLegendGraphic (optional)	Retrieves a legend for a map.

8.1.4. Relation to other OGC Standards

- OGC API Maps

8.1.5. Relation with eCH-0056

The Web Map Service (WMS) standard is currently mentioned in the following sections:

3.4, 5, 3.7, 6.3, 6.6 (QUAL-02), 6.7 (WMS-01,WMS-02,WMS-03,WMS-04,WMS-05,WMS-06,WMS-07,WMS-08,WMS-09,WMS-10), 6.11, 6.12, appendix B (Mapping: WMS metadata elements → GM03, Mapping: mandatory GM03 metadata elements → WMS, WMTS, WFS, WCS et CSW) & E

of the eCH-0056 document.

8.1.6. Proposed modifications to eCH-0056

Nothing to mention.

8.2. OGC API Maps

8.2.1. Introduction

The OGC API Maps specification defines a set of interfaces for requesting map images over the Internet. OGC API Maps makes it easy for a client to request images on demand changing parameters such as size and coordinate reference systems. A OGC API Maps server (i.e. a service that implements the OGC API Maps standard) provides information about what maps the service provides, and it produces a map and answers queries about content the content of a map.

8.2.2. Background

History

The OGC API Maps no longer includes the notion of version. Standards are now named by their functionality.

Versions

[OGC API Maps Part 1 Core](https://docs.ogc.org/DRAFTS/20-058.html) is currently in draft status.

Implementations

The implementation of the OGC API Maps is not part of the OGC Compliance Database yet.

Endpoints on the NGDI-20-60 OGC API Testbed Platform

Examples of implementations can be found on the NGDI-20-60 OGC API Testbed Platform

8.2.3. Operations

Table 11. Overview of OGC API Records resources, applicable HTTP methods

Resource	Path	HTTP method
Landing page	`/`	GET
Conformance declaration	`/conformance`	GET
Maps collections	`/collections`	GET
Maps collection	`/collections/{collectionId}`	GET
Maps	`/collections/{collectionId}/items`	GET

8.2.4. Relation to other OGC Standards

- Web Map Service (WMS)

8.2.5. Relation with eCH-0056

The OGC API Maps standard is not mentioned in the eCH-0056 document.

8.2.6. Proposed modifications to eCH-0056

The OGC API Maps standard has not yet been validated by the OGC. It is therefore difficult to take a position on its integration into the eCH-0056 document.

8.3. Web Map Tile Service (WMTS)

8.3.1. Introduction

The OGC Web Map Tile Service Implementation Standard (WMTS) defines a set of interfaces for making web-based requests of map tiles of spatially referenced data using tile images with predefined content, extent, and resolution. The standard includes the WMTS Specification (“WMTS Spec”) 07-057r7 OpenGIS Web Map Tile Service Implementation Standard along with collateral documentation such as profiles and XML documents.

8.3.2. Background

History

The version 1.0.0 of the WMTS Specification (“WMTS Spec”) 07-057r7 OpenGIS Web Map Tile Service Implementation Standard was published in 2010, and the Web Map Tile Service Simple Profile was published in 2016.

Versions

Version 1.0.0 - 07-057r7 is the current latest.

Implementations

Implementations can be found on the OGC Compliance Database.

Endpoints on the NGDI-20-60 OGC API Testbed Platform

The Web Map Tile Service (WMTS) standard has not been integrated into the Testbed platform.

8.3.3. Operations

Table 12. Overview of the WMTS operations

Operation	Description
`GetCapabilities`	Provides a “ServiceMetadata” document, which describes how to identify WMTS resources or generate WMTS request operations.
`GetTile`	Allows a client to request a tile from a WMTS server.
`GetFeatureInfo (Optional)`	Provides information about the features at or near a particular pixel location
`ServiceMetadata resource (Optional)`	Describes the abilities and information holdings of the specific server implementation.
`Tile resource (Optional)`	Provides a fragment of a map representation of a layer.
`FeatureInfo resource (Optional)`	Provides information about the features located at a particular pixel of a tile map. It does this in a manner similar to the WMS GetFeatureInfo operation by providing, for example, thematic attribute name and value pairs in textual form.

8.3.4. Relation to other OGC Standards

- OGC API Tiles

8.3.5. Relation with eCH-0056

The Web Map Tile Service (WMTS) standard is currently mentioned in the following sections:

2, 3.4, 6.4, 6.8 (`WMTS-01`,`WMTS-02`,`WMTS-03`,`WMTS-04`,`WMTS-05`,`WMTS-06`,`WMTS-07`,`WMTS-08`) appendix B (`Mapping: WMTS metadata elements → GM03`, `Mapping: mandatory GM03 metadata elements → WMS, WMTS, WFS, WCS et CSW`) & E of the eCH-0056 document.

8.3.6. Proposed modifications to eCH-0056

Add joint references to OGC API Tiles in eCH-0056 document.

8.4. OGC API Tiles

8.4.1. Introduction

The OGC API Tiles specification defines a set of interfaces for requesting map tiles over the Internet. OGC API Tiles makes it easy for a client to request tiles on demand changing parameters such as size and coordinate reference systems. A OGC API Tiles server (i.e. a service that implements the OGC API Tiles standard) provides information about what tiles the service provides, and it produces a tile and answers queries about content the content of a tile. The OGC API Tiles allows to access the same data as the Web Map Tile Service (WMTS) standard, but with a different API and could includes both vector and raster data.

8.4.2. Background

History

The OGC API Tiles standard no longer includes the notion of version. Standards are now named by their functionality. Version 1.0 was released in the end of 2022.

Versions

[OGC API Tiles Part 1 Core - 20-057](https://docs.ogc.org/is/20-057/20-057.html) is the latest version.

Implementations

Implementations can be found on the OGC Compliance Database.

Endpoints on the NGDI-20-60 OGC API Testbed Platform

Examples of implementations can be found on the NGDI-20-60 OGC API Testbed Platform

8.4.3. Operations

Table 13. Overview of OGC API Tiles resources, applicable HTTP methods

Resource	Path	Landing page
/	Conformance declaration	/conformance
Tileset list	···/tiles	Tileset
···/tiles/{tileMatrixSetId}	Tile	··· /tiles/{tileMatrixSetId}

8.4.4. Relation to other OGC Standards

- Web Map Tile Service (WMTS)

The OGC API Tiles is intended to be used in conjunction with the following OGC standards:

- OGC API Maps
- OGC API Styles
- OGC API Coverages
- [OGC API Processes]

8.4.5. Relation with eCH-0056

The OGC API Tiles standard is not part of the current version of the eCH-0056 document.

8.4.6. Proposed modifications to eCH-0056

The OGC API Tiles standard should be introduced in the eCH-0056 document alongside with the WMTS specification. Recommendations on its combination with other standards should also be proposed once these have been validated.

8.5. Styled Layer Descriptor (SLD)

8.5.1. Introduction

Geospatial data (vector and raster) have no intrinsic visual component. In order to see data, it must be styled. Styling specifies color, thickness, and other visible attributes used to render data on a map. A WMS provides a set of style options for each data set; however these are preconfigured by the server, and the user cannot create, inspect, modify a style. The Styled Layer Descriptor (SLD) is a standard that enables an application to configure in an XML document how to properly portray layers and legends in a WMS. It uses Symbology Ending (SE) to specify styling of features and coverages. The SLD Profile of WMS enhances a WMS with additional operations to support styling of features from WFS and coverages from WCS.

Source : Open Geospatial Consortium

8.5.2. Background

History

The version 1.0 was released in 2002 followed by the version 1.1 in 2007.

Versions

Versions 1.1 - 05-078r4 is the current latest.

Implementations

Implementations can be found on the OGC Compliance Database.

Endpoints on the NGDI-20-60 OGC API Testbed Platform

The Styled Layer Descriptor (SLD) standard is not part of OGC Compliance Database.

8.5.3. Operations

Not applicable.

8.5.4. Relation to other OGC Standards

- OGC API Styles
- [OGC Web Map Service (WMS)]

8.5.5. Relation with eCH-0056

The Styled Layer Descriptor (SLD) standard is currently mentioned in the following sections: 6.7 (`WMS-09`), 6.12 (`SLD-01`) & appendix E of the eCH-056 document.

8.5.6. Proposed modifications to eCH-0056

As an encoding, SLD should not be integrated in a separate section but combined with other standards (e.g. WMS).

8.6. Symbology Encoding (SE)

8.6.1. Introduction

Geospatial data (vector and raster) have no intrinsic visual component. In order to see data, it must be styled. Styling specifies color, thickness, and other visible attributes used to render data on a map. The Symbology Encoding (SE) standard defines the language to formally encode the rules of how to portray features and coverages.

Source : Open Geospatial Consortium

8.6.2. Background

History

The version 1.1.0 was approved as a standard in July 2006. Previous use of symbology encoding was through version 1.0.0 of the Styled Layer Descriptor (SLD) standard. To allow parts that are not specific to SLD and Web Map Services (WMS) to be reused, SLD 1.0.0 was split up into the separate standards of SE 1.1.0 and SLD 1.1.0.

Source : Open Geospatial Consortium

Versions

Version 1.1.0 - 05-077r4 is the current latest.

Implementations

The Symbology Encoding standard is not part of OGC Compliance Database.

Endpoints on the NGDI-20-60 OGC API Testbed Platform

Not applicable

8.6.3. Operations

8.6.4. Relation to other OGC Standards

- [OGC Symbology Conceptual Model: Core Part]
- Styled Layer Descriptor (SLD)
- OGC API Styles
- [OGC Web Map Service (WMS)]

8.6.5. Relation with eCH-0056

The Symbology Encoding (SE) standard is currently mentioned in the sections 6.11 the eCH-056 document.

8.6.6. Proposed modifications to eCH-0056

The Symbology Encoding (SE) standard should be kept in the eCH-056 document alongside with the OGC Symbology Conceptual Model standard.

8.7. OGC API Styles

8.7.1. Introduction

The OGC API - Styles draft specification defines a Web API that enables map servers, clients as well as visual style editors, to manage and fetch styles that consist of symbolizing instructions that can

be applied by a rendering engine on features and/or coverages. The API implements the conceptual model for style encodings and style metadata.

8.7.2. Background

History

OGC API - Styles - Part 1: Core standard is currently in draft status.

Versions

OGC API - Styles - Part 1: Core standard is currently in draft status.

Implementations

The OGC API Styles is not part of the OGC Compliance Database.

Endpoints on the NGDI-20-60 OGC API Testbed Platform

Examples of implementations can be found on the NGDI-20-60 OGC API Testbed Platform

8.7.3. Operations, ressources

The API building blocks support the resources and operations listed in the table below with the associated conformance class and the link to the document section that specifies the requirements.

The `baseResource` is a path template for any API resource with which styles can be associated.

Table 14. Overview of OGC API Stlyes resources and applicable HTTP methods

Resource	Path	HTTP method	Conformance class	Document reference
Base resource	`{baseResource}`	GET	core	[base-resource]
Conformance declaration	`/conformance`	GET	core	[conformance_declaration]
Styles	`{baseResource}/styles`	GET	core	[get_styles]
		POST	manage-styles	[create_style]
			style-validation	[style_validate]
Style	`{baseResource}/styles/{styleId}`	GET	core	[get_style]
		PUT	manage-styles	[replace_style]
			style-validation	[style_validate]
		DELETE	manage-styles	[delete_style]

Resource	Path	HTTP method	Conformance class	Document reference
Style metadata	`{baseResource}/styles/{styleId}/metadata`	GET	core	[get_style_metadata]
		PUT	manage-styles	[replace_style_metadata]
		PATCH	manage-styles	[update_style_metadata]

8.7.4. Relation to other OGC Standards

- Styled Layer Descriptor (SLD)
- Symbology Encoding (SE)
- OGC Symbology Conceptual Model

8.7.5. Relation with eCH-0056

The OGC API Styles standard is not mentioned in the eCH-0056 document.

8.7.6. Proposed modifications to eCH-0056

The OGC API Styles standard has not yet been validated by OGC. It is therefore difficult to take a position on its integration into the eCH-0056 document.

8.8. OGC Symbology Conceptual Model

8.8.1. Introduction

8.8.2. Background

The Symbology Conceptual Model is a new approach:

- to provide the felxibility required to achieve adequate cartographic styling and fill the needs of a variety of information communities
- to achieve high level styling interoperability without encoding dependencies.

History

Versions

Version 1.0 was released in 2020.

Implementations

The Symbology Encoding standard is not part of OGC Compliance Database.

Endpoints on the NGDI-20-60 OGC API Testbed Platform

Not applicable

8.8.3. Operations

Although not applicable here, it is important to mention that the standard is divided into different classes such as:

- Class Style
- Class Rule
- Class Symbolizer
- Class ParameterValue
- Class Literal
- Class UOM Codelist
- Class Color
- Class Fill
- Class Stroke
- Class Graphic
- Class GraphicSize
- Class Label
- Class Font

8.8.4. Relation to other OGC Standards

- Symbology Encoding (SE)
- Styled Layer Descriptor (SLD)
- OGC API Styles

8.8.5. Relation with eCH-0056

The OGC Symbology Conceptual Model: Core Part standard is not mentioned in the eCH-0056 document.

8.8.6. Proposed modifications to eCH-0056

- Integrate the OGC Symbology Conceptual Model in the same sections as the Symbology Encoding (SE) standard in the eCH-0056 document according to its extensions.
- Add a new section to the eCH-0056 document that describes the OGC Symbology Conceptual Model.
- Update the structure of the eCH-0056 document according to the versions and functionalities of the OGC Symbology Conceptual Model.

9. Metadata & Catalogue Services Standards

9.1. Introduction

This section describes the main standards for accessing metadata through web geoservices.

9.2. Catalog Services for the web (CSW)

9.2.1. Introduction

Catalogue services support the ability to publish and search collections of descriptive information (metadata) for data, services, and related information objects. Metadata in catalogues represent can be queried and presented for evaluation and further processing by both humans and software. Catalogue services are required to support the discovery and binding to registered information resources within an information community.

Source : Open Geospatial Consortium

9.2.2. Background

History

The version 2.0.2 was released in 2007 followed by the version 3.0 in 2016 which adds open search support.

Versions

Version 3.0 - 12-168r6 is the current latest. It adds open search support.

Implementations

Implementations can be found on the OGC Compliance Database.

Endpoints on the NGDI-20-60 OGC API Testbed Platform

The Catalogue Services for the Web (CSW) standard has not been integrated into the Testbed platform.

9.3. Operations, Ressources

Table 15. Overview of the CSW operations

Operation	Description
`GetCapabilities`	Returns information about the server instance.
`DescribeRecord`	Returns the information models used by the server to return the metadata records.

`GetRecordById`	Retrieves the default representation of catalogue records using an identifier.
`GetRecords`	Searches for records given a set of criteria.
`GetDomain (Optional)`	Used to obtain the range of values of a metadata property.
`Harvest (Optional)`	Create/update metadata by asking the server to `pull` metadata from somewhere.

9.3.1. Relation to other OGC Standards

- OGC API Records
- Spatio Temporal Asset Catalog

9.3.2. Relation with eCH-0056

The CSW standard is currently mentioned in the following sections:

3.4, 6.10 (`CSW-01`, `CSW-02`, `CSW-03`), appendix A, & appendix B

of the eCH-0056 document.

9.3.3. Proposed modifications to eCH-0056

Refer to version 3.0 of the specification.

9.4. OGC API Records

9.4.1. Introduction

OGC API - Records is a new OGC API standard that provides a simple and consistent way to publish and access descriptive information about geospatial resources. It is based on the OGC API - Features standard and uses the same core concepts of resources, collections, and items. The OGC API - Records standard defines a core set of functionality that can be extended by additional functionality defined in separate standards.

9.4.2. Background

History

The OGC API - Records standard is currently in draft status.

Versions

OGC API Records Part 1 Core - 20-004 is currently in draft status.

Implementations

The implementation of the OGC API Records is not part of the OGC Compliance Database yet.

Endpoints on the NGDI-20-60 OGC API Testbed Platform

Examples of implementations can be found on the NGDI-20-60 OGC API Testbed Platform

9.5. Operations, Ressources

Table 16. Overview of OGC API Records resources, applicable HTTP methods

Resource	Path	HTTP method
Landing page	`/`	GET
Conformance declaration	`/conformance`	GET
Records collections	`/collections`	GET
Records collection	`/collections/{collectionId}`	GET
Records	`/collections/{collectionId}/items`	GET

9.5.1. Relation to other OGC Standards

- Catalog Services for the web (CSW)
- Spatio Temporal Asset Catalog

9.5.2. Relation with eCH-0056

The OGC API - Records standard is not mentioned in the eCH-0056 document.

9.5.3. Proposed modifications to eCH-0056

As this is not yet an OGC standard, we cannot make any recommendations for the eCH-0056 document.

9.6. Spatio Temporal Asset Catalog

9.6.1. Introduction

The Spatio Temporal Asset Catalog (STAC) specification is a common language to describe geospatial information, so it can more easily be worked with, indexed, and discovered. At its core, the SpatioTemporal Asset Catalog (STAC) specification provides a common structure for describing and cataloging spatiotemporal assets.

Implementations

The implementation of STAC is not part of the OGC Compliance Database yet.

Endpoints on the NGDI-20-60 OGC API Testbed Platform

Examples of implementations can be found on the NGDI-20-60 OGC API Testbed Platform

9.7. Operations, Ressources

The STAC Specification consists of 4 semi-independent specifications. Each can be used alone, but they work best in concert with one another.

- **STAC Item** is the core atomic unit, representing a single spatiotemporal asset as a GeoJSON feature plus datetime and links.
- **STAC Catalog** is a simple, flexible JSON file of links that provides a structure to organize and browse STAC Items. A series of best practices helps make recommendations for creating real world STAC Catalogs.
- **STAC Collection** is an extension of the STAC Catalog with additional information such as the extents, license, keywords, providers, etc that describe STAC Items that fall within the Collection.
- **STAC API** provides a RESTful endpoint that enables search of STAC Items, specified in OpenAPI, following OGC's WFS 3.

Source : https://stacspec.org/en

9.7.1. Relation to other OGC Standards

- Catalog Services for the web (CSW)
- [OGC API - Records]

9.7.2. Relation with eCH-0056

STAC is not mentioned in the eCH-0056 document.

9.7.3. Proposed modifications to eCH-0056

It is suggested to integrate STAC into the eCH-0056 document to replace AtomFeeds.

10. Uses cases

10.1. Sensor Data Standards

In Switzerland there's no official standard defined to share with geospatial interoperability located environmental observations. In fact, the eCH-0056 at section 6.14 *Services de mesure et d'exploitation* indicates the OGC SOS v2.0 :cite[ogc-sos] and the Sensor Planning Service (SPS) v2.0 :cite[ingo2011ogc] as reference standard but indicates that currently no directive or recommendation is in place. The SOS standard was initially released in 2012. It follows the classical OGC WxS services. It defines interfaces toward sensors (data producers) and users (data consumers) based on the Simple Object Access Protocol (SOAP). Data are encoded in Extensible Language Markup (XML) and are based on the OGC Observations and Measurements (O\&M)

:cite[cox2011observations] to represent data and ONT the OGC SensorML :cite[bott2007opengis],:cite[robin2014ogc] to represent sensor description. The standard exposes two main requests for data producers: (1) *RegisterSensor* to add a procedure to the service by means of a SensorML description and (2) *InsertObservation* to inject a new observation using the O\&M. To interact with the users SOS offers three main requests: (1) *GetCapabilities* to conform with OGC commons and access metadata about the server, including how to generate requests and what parameters can be used; (2) *DescribeSensor* to access the information in SensorML of a specific procedure that generates the data; and (3) *GetObservation* to download data in O\&M format applying filters on sensors, location, time, observed properties and feature of interest. Due to the extra effort of Web Interfaces to parse and handle XMLs some SOS software like istSOS :cite[cannata2019performance] and 52North-SOS implementations started to develop their own JSON based API. To cope with this problem, in 2015 the OGC developed the SensorThings API (STA) version 1.0 :cite[ogc-sensorthings], which is not actually part of the OGC API but share most of the approaches, which are based on the use of RESTful services and JSON format. We can consider this standard as the evolution of the SOS toward the implementation of ready-to-consume services for Web user interfaces. The main difference in the data model (see figure below) is the conceptualization of *Datastream* which groups observations measuring the same observed property and produced by the same sensor and of *Things* which is a physical element that is integrated in the communication network (similarly to a Wireless Sensor Network node). STA offers a Representational state transfer (RESTful) API that permits to create, read, update, delete (CRUD) elements using the HTTPS verbs (POST, GET, PATCH, DELETE). Entities are accessed by IDs and URLs. URLs can be extended to interrelated elements and defined query parameters can be set.

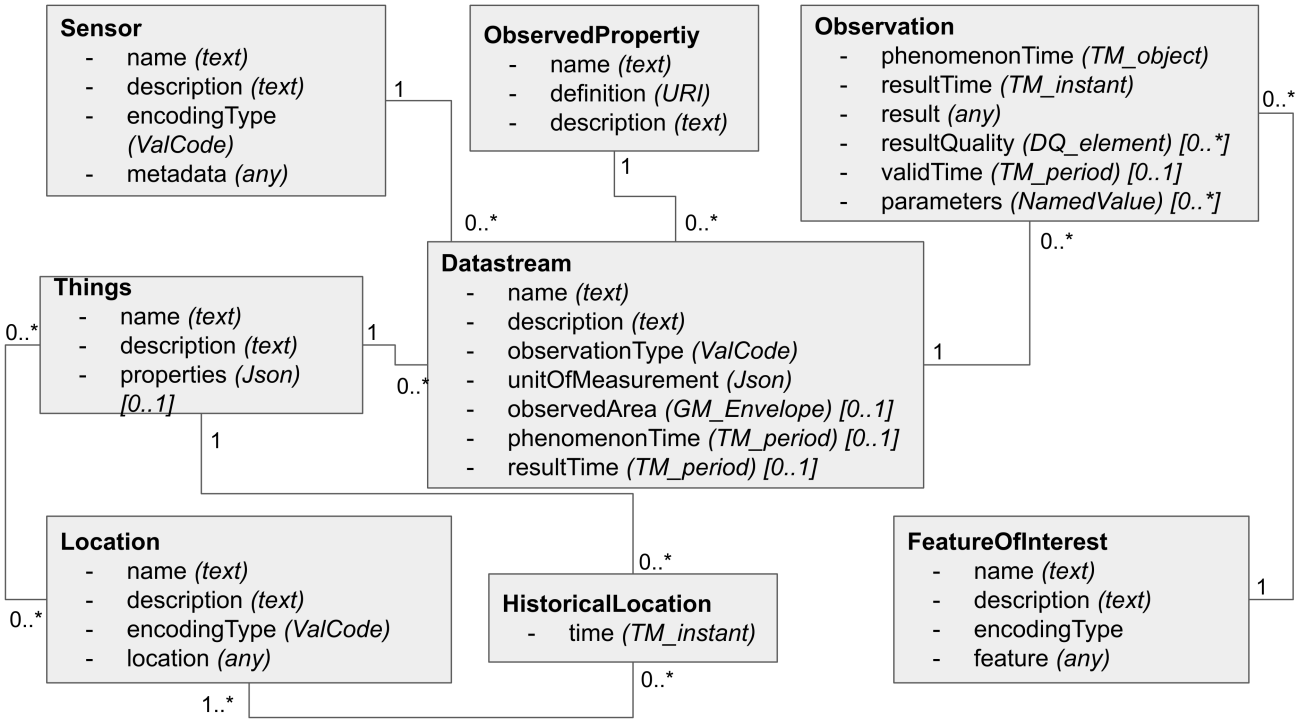

Figure 4. SensorThings API data model OGC SensorThings API v1.0

10.1.1. Environmental Observations and Consuming Applications in Switzerland

Major Swiss authoritative national offices that manage monitoring network for environmental data are: (1) MeteoSwiss, the national meteorological office that collects weather information from the

SwissMetNet that comprises about 160 automatic monitoring stations observing weather and climate variables and about 100 automatic precipitation stations :cite[suter2006swissmetnet]; (2) Federal Office for the Environment (FOEN) that use the hydrometric monitoring network composed by about 260 stations observing surface water levels and discharges :cite[schwanbeck2018reti]; and (3) Swiss Federal Institute for Forest, Snow and Landscape Research (SLF) that manage the *IMIS (Intercantonal Measuring and Information System)* network comprising 186 stations measuring snow, wind and other avalanche specific parameters. At the best knowledge of the authors, none of these offices uses any sensor related OGC standards and their Web application for data access are based on specific own non-standard solutions. In most of the cases, the Web application consists in a map with base layers served by OGC WxS services and a static vector layer of the stations localization (GeoJSON or KML) with owner defined metadata. Once a location is selected, the application, using the metadata, compose the URL that points to the observations stored in a static file (JSON, CSV or even PDF). For example, the FOEN exposes on the Web the location of monitored water temperatures as a static GeoJson (https://data.geo.admin.ch/ch.bafu.hydroweb-messstationen_temperatur/ch.bafu.hydroweb-messstationen_temperatur_de.json) with an id attribute, used to later access another static file in CSV (https://www.hydrodaten.admin.ch/lhg/az/dwh/csv/ BAFU_2167_Wassertemperatur1.csv) format and containing a series of Time-Value Pairs (TVP). Other similar examples can be found at https://meteolakes.ch and at https://meteoswiss.ch.

10.1.2. Testbed actions

The hydro-meteorological monitoring network of the Canton Ticino is currently managed using the Sensor Observation Service (SOS) standard :cite[ogc-sos]. It has been selected as representative of a practical implementation of basic data required for the climate change impact assessment pipeline. The network, which has a 40 years long time-series, is currently composed of 60 stations and 140 sensors observing precipitation, air temperature and humidity, water temperature, river height. Collected information is operationally used by the local administration to design and actuate water resources protection and allocation to guarantee a sustainable management of the resource and the natural environment while protecting from the impacts of extreme events like floods and droughts. The Sensor Things API operational applicability is evaluated by testing this standard to fulfil all the major in place daily practical operations like for example data quality management, data sharing with third parties, data collection from vendor specific sensors and data analyses and visualization. At this stage of the research, the FROST implementation of STA has been set up and the data migration scripts has been prepared and are processing the data migration that is not yet completed. Nevertheless, some preliminary considerations can be derived.

The first tested step is the migration of the SOS service to the STA service. To perform this operation a number of mapping and assumption has to be done and consequently a script has been implemented to automatically migrate data. The equivalent of registering a sensor in SOS is the creation of a *Datastream* (POST request) that includes connection with (1) a sensor, (2) an observed property and (3) a things with possibly the location. To do so you therefore need to either have the IDs of the three related elements to be used as a reference, creating them in advance if they do not exists, or include directly the elements in the payload. It is worth to be noted that in FROST, any included elements in the request, if not indicated as a reference, is going to be created regardless the existence of a perfectly equal element. This potentially lead to duplicated elements: think of a set of 10 self registering sensors that measure precipitation, at each registration they will create a new *ObservedProperty* resulting in 10 elements with the same name, definition and description, but with different ID. For this reason the script, register only once the different elements keeping track

of the IDs and finally create the *Thing*. After that, the script can start collecting observations from SOS and injecting them on the STA using a POST request of *Observations*. In istSOS we can register multiple observations at once providing a *swe:DataRecord* and in FROST we can uses the CreateObservations using the DataArrays format. Nevertheless to stress the system observations are going to be inserted one by one. This operation make the data migration a slow process, so that the data migration rate is of 1,88 observations/second. For a 20 years long series of 10 minutes data that therefore has 1,051,200 observations this result in a migration time of 22,87 days. It worth to be noted that this rate is not affected by the data retrieval request to SOS since observations are retrieved in chunks of 7 days and only when in memory sequentially injected in a loop of POST requests. Another aspect to consider is that in istSOS you can register observations of multiple observed properties making use of the *swe:DataArray* and similarly in FROST using the *Multidatastream* extension that represents a complex observation type. While in istSOS you can retrieve the observations of one of the *observedProperties* in FROST you can retrieve them only as a complex observation. Finally, in general in STA the three elements have a very minimal set of required information, and in this sense remove part of the complexity of SOS. Nevertheless to cope with compatibility it allows to extend metadata with generic fields to be used discretionally by the user to store "text-like" objects (e.g.: JSON, XML). For example USGS :cite[usgsSTA] in the \textit{Property} field of the *Things* inserted specific information like *monitoringLocationType* or *hydrologicUnit* that are then used to access data. This makes the solution compliant with STA but this lead to loosing practical interoperability since each agency would use it with non defined metadata (what an *hydrologicUnit* means? where is its definition?). Future analyses will investigate the performance of Message Queuing Telemetry Transport (https://mqtt.org/) interface for data migration, the compatibility with data validation procedures and possible implications derived by its adoption.

10.2. From data discovery & access to portrayal

10.2.1. Background

In the context of data discovery, access and portrayal, the well-known OGC WxS standards WFS :cite[ogc-wfs], WMS :cite[], WMTS :cite[ogc-wmts] have been used for more than ten years and still widely in use. In association with these standards, styling aspects are defined by the standards SLD :cite[ogc-sld] \& SE :cite[ogc-se]. These are typically referenced by the eCH-0056 Geoservices Application Profile: WMS 1.3.0 (section 6.7), WMTS 1.0.0 (section 6.8), WFS 2.0 (section 6.9.2), WCS 2.0.1 (section 6.9.3), CSW 2.0.2 (section 6.10), SE 1.1.0 (section 6.11) and SLD 1.1.0 (section 6.12).

For this project part, we focus on standardisation work at the OGC related to discovery, data access to visualisation, as made available at https://ogcapi.ogc.org/ and according to their versioning mentioned by the table below. Indeed, the table describes the relationship between the considered OGC APIs and their current equivalents in the context of raster and vector related standards.

Table 17. From WxS family to OGC APIs

OGC API	Version	WxS fashioned
OGC API Features	1.0	WFS
OGC API Maps	0.0.1	WMS
OGC API Styles	1.0.0	SLD

OGC SymCore	1.0	SE
OGC API Tiles	0.0.4	WMTS
OGC API Records	1.0.0	CSW

To test and analyze these standards and specifications, two experimental cases are setup:

- with the use of Geoclimate :cite[Bocher2021], an open source geospatial toolbox that computes a set of urban climate parameters based on OpenStreetMap data. The intent is to publish these parameters with metadata, data and maps using the new OGC APIs :cite[ogc-api].
- with the Swiss National geodata models that have been published by the Swiss Government as Minimal Geodata Models (MGM) :cite[mgm] using the Swiss INTERLIS modeling language. It is also mandatory for these models to provide styling and symbology instructions according to a spreadsheet-based model which can be obtained here for the following example.

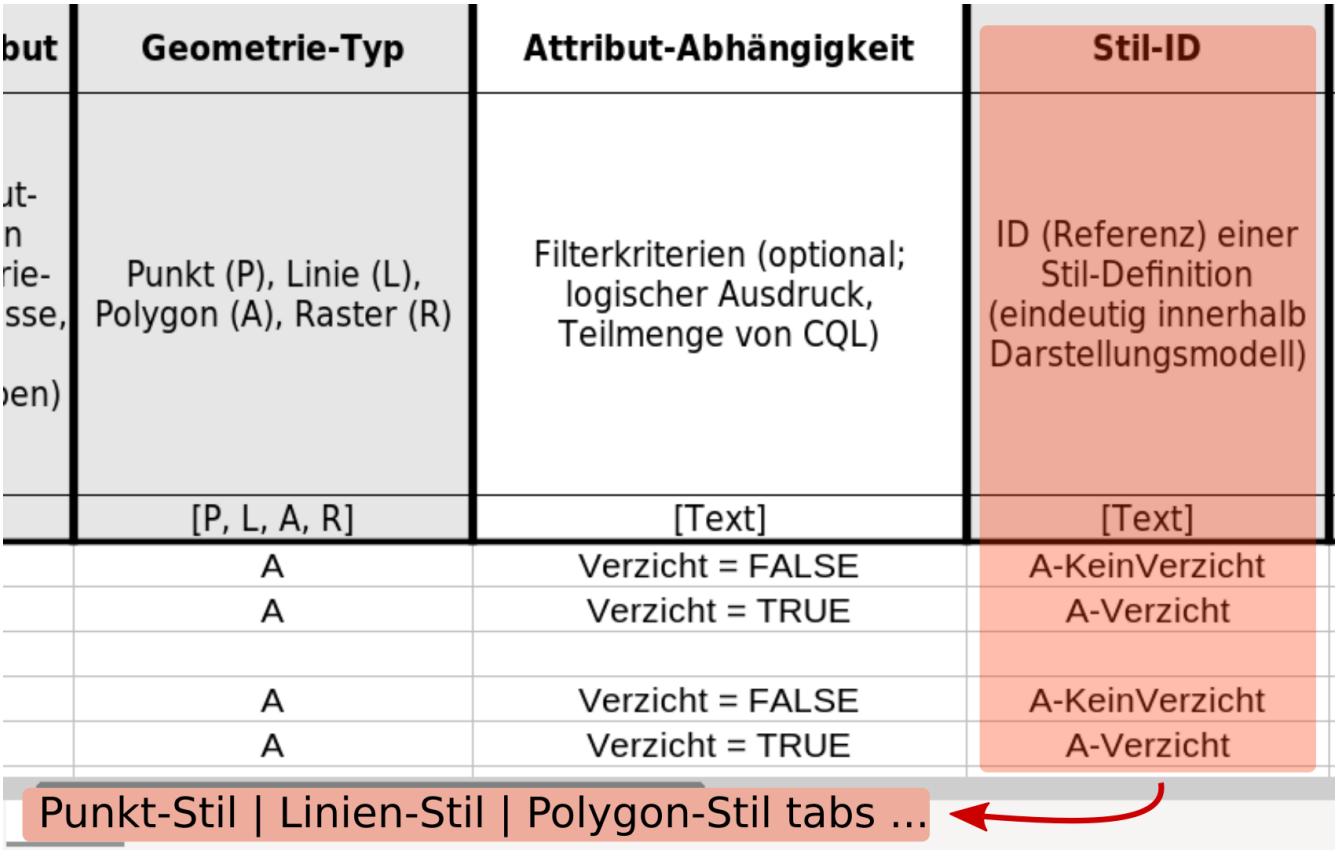

Figure 5. Styling and symbology instructions according to a spreadsheet-based model (Area reserved for water MGM)

Such styling and symbology instructions described in spread- sheet may then be formatted according to an encoding in con- formance with SymCore extensions and encodings :cite[symcore]. The encoding example below uses GeoCSS :

```
/* @title Espace réservé aux eaux (ERE)
 * @abstract Modèle de représentation pour
    l'espace réservé aux eaux de surface,
    cours d'eau latéraux et plans d'eau */
 * {
    /* @title ERE */
```

```
        [obligation = 1] {
            fill: #ddebf7;
            stroke: #ffcc00;
            stroke-width: 6px;
        }
        ;
        /* @title Renonciation */
        [obligation = 0 ] {
            stroke:#ffcc00;
            stroke-width: 4px;
            stroke-dasharray: 4 4;
        }
        ;
    }
```

Which allows to produce the following map :

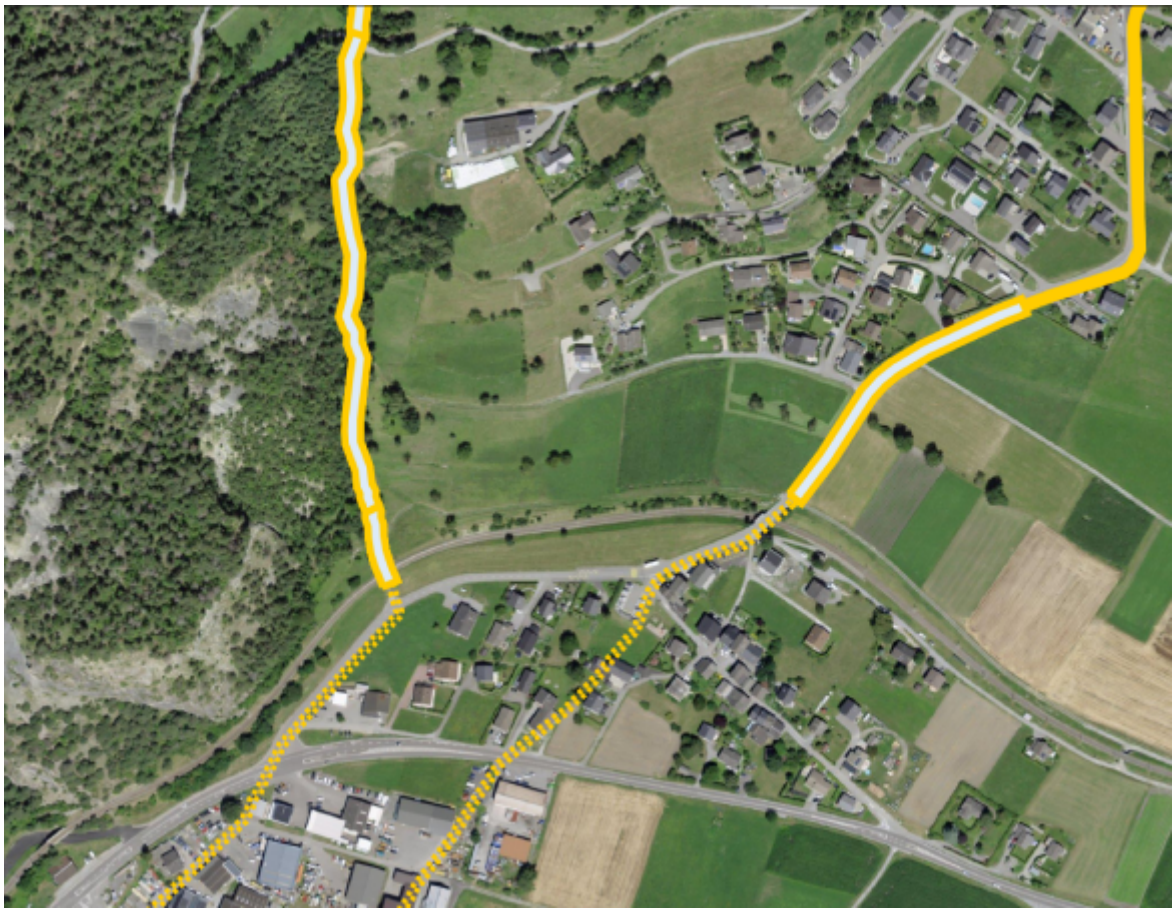

Figure 6. Overview of the Area reserved for water portrayal using the above GeoCSS encoding for GeoServer

Regarding the publication of vector data using the OGC API Features standard, we can state that all software packages already support this standard :cite[ogc-api-feature-implementations]. Regarding the tiling of data sets, for a long time the existing WMTS standard has been largely used, but a standard for vector tiling has never been established up to now. A possible explanation for this lack of standardization is on the one hand the complexity of vector tiling (e.g. regarding the handling of attributes or projections), but on the other hand the success of the Mapbox Vector tiles specification :cite[mvt-spec] that have been widely adopted. The OGC API Tiles specification is on a conceptual level similar to the WMTS standard and defines the addressing and tiling of the data. One

difference is that the OGC API Tiles specification allows for several formats (both vector and raster) to be computed. This way of defining tiles assures on the one hand the compatibility with existing WMTS services (i.e. allowing applications to easily integrate both existing WMTS layers with tiled vector layers), but also with the Mapbox Vector tiles specification. On the software side GeoServer already supports the OGC-API tiles specification rendering the formats jpg, png, GeoJSON, topojson and mapbox-vector-tile.

Concerning portrayal, we may notice two related aspects: about OGC API Styles, about OGC SymCore. Firstly, OGC API Styles is inline with the conceptual model for styles, style encodings and style metadata as documented in chapter 6 of the *OGC Testbed-15: Encoding and Metadata Conceptual Model for Styles Engineering Report*. Especially it states that a style may be made available in one or more so-called stylesheets. Moreover style metadata are made available through the API with general descriptive information about the style, structural information (e.g., layers and attributes), and so forth to allow users to discover and select existing styles for their data. Having several stylesheets available does not guarantee the same visualization of the cartographic result for the final user, because each stylesheet may be based on different models and encodings (e.g. SLD, Mapbox style, GeoCSS, etc). Nonetheless, it opens the possibility to make full use of the cartographic capabilities and richness of the various underlying symbology models.

Secondly, OGC SymCore pushes forward portrayal interoperability with the idea to standardize also the symbology part. The approach is so-called *one conceptual model, many encodings*, which means that many flavors of encodings are possible but each in conformance with a common conceptual rendering behavior of cartographic capabilities. The intent is that finally, independently of the compliant encoding used, the cartographic result will be the same for the final user.

10.3. Earth Observation data

Regarding Earth Observation data acquired by satellites, there are some interesting new emerging standards in the OGC API family that are currently being developed. Among the selected standards to be tested, we have considered: Coverages; Environmental Data Retrieval (EDR); Records; Processes; and the Discrete Global Grid System (DGGS). To test these new standards, we have decided to set up a pygeoapi instance interfaced with the Swiss Data Cube (Analysis Ready Data archive of satellite imagery :cite[sdc]. As of May 2022, we have developed/tested the following scenarios to use the various APIs mentioned previously using as a source a Normalized Difference Water Index (NDWI) time-series generated with the Swiss Data Cube :cite[sdc-ndwi]:

- Single geotiff and NetCDF (multidimensional) files published as Coverages (coverage API)
- item Series of geotiff files published with the SpatioTemporal Asset Catalog (STAC)
- NetCDF file exposed with Environmental Data Retrieval (EDR API) to extract time-series of pixel values
- Metadata (from the SDC GeoNetwork catalog) imported and published using the (Records API)
- Create a zonal stat process to analyze data by canton (Process API)
- Test different plugins in QGIS and R to query the tested APIs
- item Explore the Discrete Global Grid System (DGGS)

The first test showed that the publication is smooth and somehow easier than with OGC WxS

fashioned services making simple the publication of complex and large raster layers. The first tangible result is the release (in April 2022) in production mode of the STAC API to expose the entire content of the Swiss Data Cube: 38 years (1984-2022) of satellite imagery on Switzerland (Landsat5-7-8-9; Sentinel-1-2) + other national datasets (e.g., Land Cover, Digital Elevation Model). The API is available at: https://explorer.swissdatacube.org/stac allowing to query and access Analysis Ready Data served by the Swiss Data Cube directly in a client application (i.e. QGIS) via a JSON format (see figure below). Once all the scenarios have been completed, a demonstration instance will be made publicly available to access the different tested API on the Testbed platform.


```
{
    "stac_version": "1.0.0", "id": "odc-explorer",
    "title": "Default ODC Explorer instance",
    "type": "Catalog",
    "links": [
    {
      "title": "Collections",
      "description": "All product collections",
      "rel": "children",
      "type": "application/json",
      "href": "http://explorer.swissdatacube.org
                  :5001/stac/collections"
    },
    {
      "title": "Arrivals",
      "description": "Most recently added items",
      "rel": "child",
      "type": "application/json",
      "href": "http://explorer.swissdatacube.org
                  :5001/stac/arrivals"
    },
    { ... },
    {
      "title": "combiprecip_scene",
      "description": "Hourly Precipitation
          Estimation through Raingauge-Radar
          (by GRID-Geneva)",
      "rel": "child",
      "href": "http://explorer.swissdatacube.org
                  :5001/stac/collections
                       /combiprecip_scene"
    },
    { ... }
}
```


11. Conclusion

The standardization process is still in heavy activity to progress on the OGC API standards, with pieces approved step by step. Also we see many software implementing these pieces. Sometimes it

is even by supporting the standardization work at the heart as a kind of continuous proof of concept. The OSGeo communiy is particularly active in this field, being it in C/C++, Java or Python.

Considering the utilization of the well-established OGC WxS-standard series, the new OGC standards represent a major step towards interoperability. An example of applications that clearly benefits from this process are hybrid applications that utilize both non-spatial data and spatial data: if yesterday developers had to implement specific service interfaces for spatial data and for non-spatial data for one application, this is more and more simplified thanks to a common architectural style and geostandards. For the latter, we may also notice the general will to keep specifications modular and extensible including a conceptual level and accepting various flavors of formats.

In the past, Switzerland already played an important role in the establishment of geospatial standards due to the federal organization of the country. In the early 1990 the INTERLIS :cite[eCH-0031] standard has been created, the use of which is today required by the law for specific fields (e.g. cadastral data and minimal geodata models that define how different entities exchange data for specific themes). The Swiss Government has invested many resources to build interfaces that allow for a compatibility between the national standards and the international standards (e.g. the eCH-0018 standard :cite[ech-0118] which specifies an interface between the INTERLIS standard and the OGC Geographic Markup Language standard). Due to the regular establishment of new international standards such as the OGC API family, these interfaces need to be adapted.

Therefore and given the geostandards.ch strategy :cite[geostandards] - *sustainable and benefit-oriented standardisation in the field of geoinformation in Switzerland as well as the effective steering of the development of solutions and software tools in the environment of GeoIG and the National Spatial Data Infrastructure (NSDI)* - the Testbed Platform described in this article should be helpful, from the tactical level to the operational level. As such, the Testbed approach is intended to live on through successive iterations.

12. Appendix A: bibliographical references

bibliography::[]

13. Appendix B: conference papers

- Spiergarten Treffen 2022
- FOSS4G 2022 & ISPRS

14. Appendix C: follow up meetings

14.1. 25-05-2022

14.1.1. Date & Time

- 25.05.2022 09:00-10:00

14.1.2. Attendees

- Frank
- Jens
- Maxime

14.1.3. Agenda

Who	What	Comments
Frank	Plan MS-Teams meetings every 3 weeks	Done
Frank	Synchronisation of meeting notes on GitHub	Done
Frank	Valuation of presentation materials, publication etc. on GitHub	Done
Frank	Add a link to the TB platform on the project documentation an ad hoc section in the documentation	Done

14.1.4. Related documents and resources

- Project Documentation
- NGDI-20-60 OGC API Testbed Platform

14.1.5. Next meeting

31.05.2022

14.1.6. Miscellaneous

Nothing to mention

14.2. 31-05-2022

14.2.1. Date & Time

31.05.2022 09:00-12:00

14.2.2. Attendees

- Frank
- Maxime

14.2.3. Agenda

Who	What	Comments
Frank	Prepare templates for meeting notes on GitHub	Done
Frank	Feed the documentation and the testbed platform (Howto's, WP's Results)	To be carried out throughout the project
Frank	Depict the CH Standards Landscape - presentation of the situation as it is	Done
Frank	Make a decision matrix for the choice of standards (OGC, eCH). Typical questions & critical factors are: • When is a standard interesting / ready to be standardized in Switzerland as well? • Groundtruthing/verification (Interviews with e-CH / users etc.)	To be carried out throughout the project

14.2.4. Related documents and resources

- Nothing to mention

14.2.5. Next meeting

20.06.2022

14.2.6. Miscellaneous

Nothing to mention

14.3. 20-06-2022

14.3.1. Date & Time

20.06.2022 09:00-10:00

14.3.2. Attendees

- Frank

- Jens
- Maxime
- Olivier

14.3.3. Agenda

Who	What	Comments
Olivier	Presentation of the FOSS4G publication	Done
Jens	Discusses that the new OGC standards are shifting focus towards web-applications. This trend potentially creates problems regarding bridges between INTERLIS and new OGC standards.	-
Jens	Proposes the build the output in conformity with eCH-structure (e.g. https://www.ech.ch/it/standards/60396)	This idea is approved and has to be added to the project final restitution.
Jens	Proposes to implement APIs in relation to MeteoSwiss data (following Massimiliano's idea)	This is a great idea for a future iteration.
Frank	State of play of financial resources	Complete the financial report for the 24.06.2022
Frank	Dissemination of the work in the context of workshops	Frank will take care of the exchange with swisstopo regarding the organisation of the Kolloquium

14.3.4. Related documents and resources

- FOSS4G article
- 20220620-NGDI-20-60 follow up meeting notes

14.3.5. Next meeting

11.07.2022

14.3.6. Miscellaneous

Nothing to mention

14.4. 11-07-2022

14.4.1. Date & Time

11.07.2022 09:00-09:30

14.4.2. Attendees

- Frank
- Maxi
- Maxime

14.4.3. Agenda

Who	What	Comments
Frank	eCH-0056 kickoff meeting in august	
Maxime	integrate the ech-0056 structure into the report	
Frank	will provide a proposition of reporting structure by the end of the week (15/07/2022)	
Maxi	SOS mentioned in ecH-0056 but no recommendation made	
Maxi	MeteoSwiss not really interested in new OGC APIs but other data provider could be identified	
Frank	Quick How to’s to play with the new OGC APIs (e.g. ArcGIS, QGIS, web client, etc.)	

14.4.4. Related documents and resources

14.4.5. Next meeting

06.09.2022

14.4.6. Miscellaneous

Nothing to mention