OGC® DOCUMENT: 20-071

External identifier of this OGC® document: http://www.opengis.net/doc/UG/ogcapi-common/1.0

OGC®
Making location count.

OGC API - COMMON - USERS GUIDE

USER GUIDE

PUBLISHED

Version: 1.0.0
Submission Date: 2022-11-09
Approval Date: 2022-02-02
Publication Date: YYYY-MM-DD
Editor: Charles Heazel

OGC®
Making location count.

License Agreement

Copyright notice

Note

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CONTENTS

I ABSTRACT

The OGC API — Common Standard is a multi-part Standard that specifies reusable building-blocks that can be used in the construction of OGC API Standards. The OGC API — Common — Users Guide presents information useful to developers or users of implementations of the OGC API — Common Standard. The information in the Users Guide is not normative. That is, it is not mandatory. However, it may prove essential to fully understand the normative text in the OGC API — Common Standard. The Users Guide is therefore intended to serve as an aid to developers and users.

II KEYWORDS

The following are keywords to be used by search engines and document catalogues.

OGC document, geographic information, spatial data, API, json, html, OpenAPI, REST, Common

III PREFACE

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the Standard set forth in this document, and to provide supporting documentation.

IV SECURITY CONSIDERATIONS

No security considerations have been made for this document.

V SUBMITTING ORGANIZATIONS

The following organizations submitted this Document to the Open Geospatial Consortium (OGC):

- Ecere Corporation
- Heazeltech LLC
- Universitat Autònoma de Barcelona (CREAF)

VI SUBMITTERS

All questions regarding this submission should be directed to the editors or the submitters:

Table — Submitters

NAME	AFFILIATION
Charles Heazel *(editor)*	Heazeltech
Jérôme Jacovella-St-Louis	Ecere Corporation
Joan Masó	UAB-CREAF

1

SCOPE

1 SCOPE

The OGC API — Common Standard is a multi-part Standard that specifies reusable building-blocks that can be used in the construction of OGC API Standards. The purpose of the OGC API — Common Standard is to define those fundamental building blocks and requirements which are applicable to all, if not most, OGC Web API Standards.

The **OGC API — Common — Users Guide** presents information useful to developers or users of implementations of OGC API — Common. The Users Guide focuses on OGC API — Common but is applicable to other OGC API Standards, specifically those OGC API Standards that implement requirements classes from OGC API — Common. The Users Guide discusses the architecture adopted by OGC API — Common, use of identifiers, link relations, OGC API Collections, and the OpenAPI specification.

2

TERMS AND DEFINITIONS

2 TERMS AND DEFINITIONS

This document uses the terms defined in OGC Policy Directive 49, which is based on the ISO/IEC Directives, Part 2, Rules for the structure and drafting of International Standards. In particular, the word “shall” (not “must”) is the verb form used to indicate a requirement to be strictly followed to conform to this document and OGC documents do not use the equivalent phrases in the ISO/IEC Directives, Part 2.

This document also uses terms defined in the OGC Standard for Modular specifications (OGC 08-131r3), also known as the ‘ModSpec’. The definitions of terms such as standard, specification, requirement, and conformance test are provided in the ModSpec.

For the purposes of this document, the following additional terms and definitions apply.

2.1. Uniform Resource Identifier (URI)

an identifier consisting of a sequence of characters matching the syntax rule named “<URI>”. (IETF RFC 3986)

2.2. Uniform Resource Locator (URL)

the subset of URIs that, in addition to identifying a resource, provide a means of locating the resource by describing its primary access mechanism (e.g., its network “location”). (IETF RFC 3986)

2.3. Web API

API using an architectural style that is founded on the technologies of the Web. (W3C Data on the Web Best Practices)

2.4. **Web Resource**

a resource that is identified by a URI.

3

ABBREVIATED TERMS

3 ABBREVIATED TERMS

API	Application Programming Interface
CORS	Cross-Origin Resource Sharing
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IANA	Internet Assigned Numbers Authority
OGC	Open Geospatial Consortium
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
YAML	YAML Ain't Markup Language

4

EVOLUTION FROM OGC WEB SERVICES

4 EVOLUTION FROM OGC WEB SERVICES

OGC Web Service (OWS) Standards implement a Remote-Procedure-Call-over-HTTP architectural style using XML for payloads [7]. This was the state-of-the-art when OWS Standards were originally designed in the late 1990s and early 2000s. However, technology has evolved. New Resource-Oriented APIs provide an alternative to the Service-Oriented Approach. New OGC Web API Standards are under development to provide API alternatives to the OWS Standards.

The OGC API — Common Standard specifies common modules for defining OGC Web API Standards that follow the current Web architecture. In particular, the recommendations as defined in the W3C/OGC best practices for sharing Spatial Data on the Web as well as the W3C best practices for sharing Data on the Web [11] [10].

5

ARCHITECTURE

5 ARCHITECTURE

The following “architecture” captures the core concepts which inform all OGC API Standards.

5.1. Datasets

Web APIs implement a Resource Oriented approach to Web-based distributed computing. A coherent set of APIs must be based on a common understanding of the resources to be shared. For OGC API Standards, those resources would be identified using an abstract concept of a geospatial dataset. Such a dataset would have the following characteristics.

- A dataset can be a vector feature collection, a coverage/imagery, or a collection of datasets (therefore potentially a mix of any of these things).
- A dataset has associated metadata, including some essential information:
    - Information about what type of dataset it is (vector features (and what type of vector features if limited to one type, e.g., polygons,lines, or points), coverages (values) / imagery (pixels), or sub-datasets— more than one of those things);
    - A textual identifier (e.g., which figures in the resource path);
    - A title (short name / description);
    - Access point for the dataset (could be hosted locally or remote);
    - Geospatial & temporal extent;
    - Resolution/scale;
    - Units/Range/Bit-Depth/Channels/Dimensions etc. for imagery/coverages; and
    - A description of queryables, if applicable.
- Keywords/Tags, and longer descriptions are also a commonly useful piece of metadata information.
- Any other ISO 19115 metadata fields can also be associated with the dataset, but are nowhere near as essential to discovering and using geospatial data as those mentioned above. Metadata containing, at a minimum, those essential elements can always be retrieved in ISO 19115-based encodings and potentially other formats.

5.2. Processes

Processes take one or more datasets as input, add parameters, and produce one or more datasets as output. The following are examples of types of processes that can all run on a server and be invoked by a remote client application:

1. A complex process built as a container or executable, as typical of implementations of the Web Processing Service (WPS) Standard;
2. Process description languages such as the Web Coverage Processing Service (WCPS); and
3. Pre-defined named processes such as 'vectorization', 'buffering', 'rasterization' or 'rendering of a styled map'.

All of these kinds of processes could share aspects such as taking in an OGC API dataset as input and their output being usable as an OGC API dataset, for direct access and/or asynchronous delivery, and support multiple data partitioning/access mechanisms, estimates/billing elements, and so on.

5.3. Resource Paths

The resources exposed through implementations of OGC API Standards are accessed through standardized URL templates. These "paths" and their associated resources are described in Table 1. This table documents a selection of the paths and resources as of December 2022. This information will be updated as these and other OGC API Standards mature.

Table 1 — A selection of OGC Dataset Paths from OGC API Standards approved as of 2022-11-08

PATH TEMPLATE	RESOURCE
Common	
/collections	Metadata describing the spatial collections available from this API.
/collections/{collectionId}	Metadata describing the collection which has the unique identifier `{collectionId}`.
Features	

PATH TEMPLATE	RESOURCE
/collections/{collectionId}	The Feature Collection resource identified by the `{collectionId}` parameter.
/collections/{collectionId}/items	The individual Features in a Feature Collection.
Spatio-temporal environmental data resources	
/collections/{collectionId}	Identifies a collection of spatio-temporal data with the unique identifier {collectionId}.
/collections/{collectionId}/{queryType}	Identifies an Information Resource of type {queryType} associated with the {collectionId} collection.
Tiles	
Note: A tile is associated with a resource. {resource} is a place holder for a path segment appropriate for a resource type.	
/collections/{collectionId}/tiles/{tileMatrixSetId}/{tileMatrix}/{tileRow}/{tileCol}	Feature tile representation of real-world elements at a given resolution restricted by the selected Tile Matrix Set.
/collections/{collectionId}/tiles/{tileMatrixSetId}	A description of the TileMatrixSet identified by the {tileMatrixSetId} identifier.
/collections/{collectionId}/tiles	An enumeration of the feature tileset list.
Processes	
/process	Lists the processes this API offers.
/processes/{process-id}	Returns a detailed description of a process.
/processes/{process-id}/jobs	Returns the running and finished jobs for a process (GET), Executes a process, i.e. creates a new job. Inputs and outputs will have to be specified in a JSON document that needs to be send in the POST body. (POST)
/processes/{process-id}/jobs/{job-id}	Returns the status of a job of a process.
/processes/{process-id}/jobs/{job-id}/results	Returns the result of a job of a process.

6

IDENTIFIERS

6 IDENTIFIERS

The Architecture of the World Wide Web establishes the URI as the single global identification system for the Web [8]. Therefore, URIs or URI Templates are used in OGC Web API standards to identify key entities in those standards. In accordance with OGC policy, only the Uniform Resource Locator (URL) form of URIs is used.

The normative provisions in the OGC API – Common – Part 1: Core standard are denoted by the URI `http://www.opengis.net/spec/ogcapi-common-1/1.0`. All Requirements, Conformance Modules, and Conformance Classes that appear in the standard are denoted by partial URIs that are relative to this base.

Resources described in the standard are denoted by partial URIs that are relative to the `root` node of the API. This node serves as the head of the resource tree exposed through an API. In OpenAPI, the root node is identified by the `url` field of the Server Object. In the OGC API – Common standard the tag `{root}` designates the root node of a URI.

The partial URIs used to identify Resources in the OGC API – Common standard are referred to by the resource `path`. The purpose of a resource `path` is to identify the referenced resource within the context of the standard. Implementors are encouraged to use these partial URIs in their implementations, thereby providing a common look and feel to implementations of OGC API Standards.

This standard defines Resources which may appear in more than one place in the API. These Resource Types are identified by name rather than by URI.

Summary for Developers:

RFC 3986 defines a URI in Backus-Naur Form (BNF) as follows:

```
URI = scheme ":" hier-part [ "?" query ] [ "#" fragment ]

        hier-part     = "//" authority path-abempty
                      / path-absolute
                      / path-rootless
                      / path-empty

        authority     = [ userinfo "@" ] host [ ":" port ]

        path-abempty  = *( "/" segment )

        path-absolute = "/" [ segment-nz *( "/" segment ) ]

        path-rootless = segment-nz *( "/" segment )

        path-empty    = 0<pchar>
```

Figure 1 — Backus-Naur Definition of URI

The following rules should be used when interpreting the BNF [6] for use with this standard:

- `scheme` is assumed to be `HTTP` or `HTTPS`

- `authority` is provided by the API developer
- `{root}` designates the `scheme`, `authority`, and `path` to the root node of the API implementation.
- only the `path-absolute` and `path-rootless` patterns are used
- parameters passed as part of an operation are encoded in the `query`.
- parameters passed in HTTP headers or as cookies are out of scope for this Standard.

The following example shows a URI categorized according to RFC 3986 and OGC Web API standards.

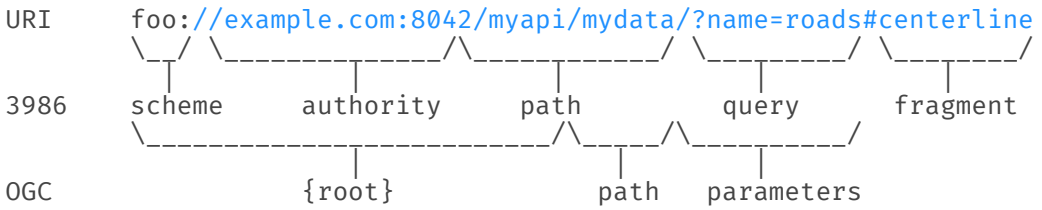

Figure 2 — Example URI and Components

The OGC API – Common – Part 1: Core standard does not restrict the lexical space of URIs used in the API beyond the requirements of the HTTP and URI Syntax IETF RFCs. If URIs include reserved characters that are delimiters in the URI subcomponent, these have to be percent-encoded. See Clause 2 of RFC 3986 for details.

NOTE: OGC Web API standards may include a community-defined identifier as part of a URI (ex. image id or feature id). Definition of the format of those identifiers is out of scope for these standards. Implementers should take care that these identifiers are properly encoded (see RFC 3986) in the URIs for all hosted resources.

7

LINK RELATIONS

7 LINK RELATIONS

This section provides a discussion of link relations, the governing policies and Standards, as well as how they should be used.

Links between elements of an OGC API are encoded using the link relation schema provided in Figure 3.

```
type: object
required:
  - href
  - rel
properties:
  href:
    type: string
    description: Supplies the URI to a remote resource (or resource fragment).
    example: http://data.example.com/buildings/123
  rel:
    type: string
    description: The type or semantics of the relation.
    example: alternate
  type:
    type: string
    description: A hint indicating what the media type of the result of
dereferencing the link should be.
    example: application/geo+json
  hreflang:
    type: string
    description: A hint indicating what the language of the result of
dereferencing the link should be.
    example: en
  title:
    type: string
    description: Used to label the destination of a link such that it can be
used as a human-readable identifier.
    example: Trierer Strasse 70, 53115 Bonn
  length:
    type: integer
```

Figure 3 — Link Schema

7.1. Attributes of Link Relations

The attributes supported by link relations are presented in this subsection.

7.1.1. Href Attribute

The `href` attribute is mandatory.

“The Locator Attribute (href) supplies the data that allows an application to find a remote resource or resource fragment. The value of this attribute is an Internationalized Resource

Identifier (IRI) which serves as the Uniform Resource Identifier for the remote resource." (W3C XLink Version 1.1)

NOTE: An IRI is a sequence of characters from the Universal Character Set (Unicode/ISO 10646). A mapping from IRIs to URIs is defined, which means that IRIs can be used instead of URIs, where appropriate, to identify resources. (RFC 3987)

7.1.2. Rel Attribute

The `rel` attribute is mandatory.

"In the simplest case, a link relation type identifies the semantics of a link. For example, a link with the relation type "copyright" indicates that the current link context has a copyright resource at the link target.

Link relation types can also be used to indicate that the target resource has particular attributes, or exhibits particular behaviors; for example, a "service" link implies that the link target can be used as part of a defined protocol (in this case, a service description).

Relation types are not to be confused with media types [RFC 2046]; they do not identify the format of the representation that results when the link is dereferenced. Rather, they only describe how the current context is related to another resource.

Relation types SHOULD NOT infer any additional semantics based upon the presence or absence of another link relation type, or its own cardinality of occurrence. An exception to this is the combination of the "alternate" and "stylesheet" registered relation types, which has special meaning in HTML for historical reasons.

There are two kinds of relation types: registered and extension." (RFC 8288)

7.1.3. Type Attribute

The `type` attribute is optional.

"The "type" attribute, when present, is a hint indicating what the media type of the result of dereferencing the link should be. Note that this is only a hint; for example, it does not override the Content-Type header field of a HTTP response obtained by actually following the link. The type attribute MUST NOT appear more than once in a given link-value; occurrences after the first MUST be ignored by parsers." (RFC 8288)

7.1.4. Hreflang Attribute

The `hreflang` attribute is optional.

"The "hreflang" attribute, when present, is a hint indicating what the language of the result of dereferencing the link should be. Note that this is only a hint; for example, it does not override the Content-Language header field of a HTTP response obtained by actually following the link.

Multiple hreflang attributes on a single link-value indicate that multiple languages are available from the indicated resource." (RFC 8288)

7.1.5. Title Attribute

The `title` attribute is optional.

"The "title" attribute, when present, is used to label the destination of a link such that it can be used as a human-readable identifier (e.g., a menu entry) in the language indicated by the Content-Language header field (if present). The title attribute MUST NOT appear more than once in a given link; occurrences after the first MUST be ignored by parsers." (RFC 8288)

7.1.6. Length Attribute

The `length` attribute does not appear to be defined in the normative Standards.

7.2. Relation Types

There are two kinds of relation types; registered and extension.

Registered relation types are registered in the IANA register at https://www.iana.org/assignments/link-relations/link-relations.xhtml [5]. Registered relation types are used in OGC API Standards whenever appropriate.

Extension relation types are those which are not registered with IANA. These extension types are in the form of "—- a URI [RFC 3986] that uniquely identifies the relation type. Although the URI can point to a resource that contains a definition of the semantics of the relation type, clients SHOULD NOT automatically access that resource to avoid overburdening its server.

The URI used for an extension relation type SHOULD be under the control of the person or party defining it or be delegated to them.

When extension relation types are compared, they MUST be compared as strings (after converting to URIs if serialized in a different format) in a case-insensitive fashion, character by character. Because of this, all-lowercase URIs SHOULD be used for extension relations.

Note that while extension relation types are required to be URIs, a serialization of links can specify that they are expressed in another form, as long as they can be converted to URIs." (RFC 8288)

Extension relation types used in OGC API Standards are registered at http://www.opengis.net/def/rel

8

WHAT IS A COLLECTION?

8 WHAT IS A COLLECTION?

This section presents a definition for an OGC API Collection, as well as a rationale for such a collection.

8.1. Definition

<OGC API> Collection
A geospatial data resource that may be available as one or more sub-resource distributions that conform to one or more OGC API Standards.

An OGC API Collection is a restriction on the general concept of a collection.

1. An OGC API Collection may only contain data. It does not encompass non-data resources (but the Collection resource may still have related auxiliary resources, such as schemas, or processes, as sub-resources).
2. Geospatial datasets include Spatial Things and/or Temporal Things as data items (but not all data items in the dataset have to have a spatial or temporal extent).
3. An OGC API Collection has one or more sub-resource distributions.
4. A distribution provides access to a representation of a dataset.

8.2. Rationale

Dataset
collection of data

NOTE 1: Published or curated by a single agent, and available for access or download in one or more serializations or formats.

NOTE 2: This definition was adopted as the OGC definition of “dataset” by the OGC Architecture Board (OAB). See OAB issue 1434.

A dataset is a restriction on the general concept of a collection.

1. A dataset is a collection of data.
2. A dataset is published or curated by a single agent.
3. A dataset is available in one or more serializations or formats (distributions).

4. A distribution is synonymous to the REST concept of a representation.

8.3. Key Concepts

- **Collection**
  A body of resources that belong or are used together. An aggregate, set, or group of related resources. (Derived from Webster's Dictionary)

- **Distribution**
  specific representation of a dataset. (DCAT)

  EXAMPLE: a downloadable file, an RSS feed or an API.

- **Sub-resource Distribution**
  a subset of the distribution of a dataset

- **Feature Collection**
  a set of **Features** from a **dataset**

- **Media Type**
  The data format of a representation. [4]

- **Resource**

  1. Any information that can be named. [4]
  2. A conceptual mapping to a set of entities. [4]
  3. A resource R is a temporally varying membership function $M_R(t)$, which for time t maps to a set of entities, or values, which are equivalent. [4]

- **Representation**

  1. A format matching one of an evolving set of standard data types, selected dynamically based on the capabilities or desires of the recipient and the nature of the resource. [4]
  2. The current or intended state of a resource. [4]

- **Spatial Thing**
  anything with spatial extent, (i.e. size, shape, or position) and is a combination of the real-world phenomenon and its abstraction. (W3C/OGC Spatial Data on the Web Best Practice)

- **Temporal Thing**
  Anything with temporal extent, i.e. duration. e.g. the taking of a photograph, a scheduled meeting, a GPS time-stamped track-point. (W3C Basic Geo)

9

USING OPENAPI (INFORMATIVE)

9 USING OPENAPI (INFORMATIVE)

The following section describes the main components of an OpenAPI document and how they should be used to construct API requests. The authoritative source is the OpenAPI 3.0 standard available here.

9.1. OpenAPI Document (root)

The OpenAPI document (Root) contains descriptive information about the API and serves as the root for the other parts of the document.

9.2. Paths

All API resources are accessed through a path. The Paths field of the OpenAPI document defines all of the paths available through this API. The Paths field is a collection of Paths Objects. Each Paths Object includes the URL or URL template for this path, any Server Objects specific to this Path, Parameters which are applicable to this Path, and an Operation Object for each of the HTTP Verbs applicable to this Path.

9.3. Operations

Operation Objects provide the details needed to create an HTTP request and response. Specifically, they provide definitions of the request message (including parameters) and all possible responses. In addition, they define any operation specific Server Objects or Security Requirements.

9.4. Parameters

Parameter Objects describe parameters which can be used in an API. These objects provide the parameter name, where it is passed (query, header, path, or cookie), and a detailed description of its structure (if needed).

Parameter Objects can be defined at the following levels.

- Path — applicable to all operations on this path.
- Operation — only applicable to this operation. Overrides any parameters defined at the Path level which have the same name.

9.5. Servers

An API is not restricted to a single server. Furthermore, the set of valid servers may be different for different sections of the API. An OpenAPI Server Object describes a single server. Most importantly, it provides the URL to that server. Server Objects are grouped into arrays. These arrays provide a list of the servers which can be used to access a section of the API.

```
servers:
  - url: https://dev.example.org/
    description: Development server
  - url: https://data.example.org/
    description: Production server
```

Figure 4 — Example Array of Server Objects

Server Objects can be defined at the following levels.

- Root — applicable to the whole API unless overridden.
- Path — applicable to all operations on this path. Servers defined at the Root level are still valid.
- Operation — only applicable to this operation. Overrides any servers defined at the Path or Root level.

9.5.1. Building URLs

An OpenAPI document can describe a large number of URLs. Extracting all of the URLs is a non-trivial task. The OpenAPI objects used to construct URLs are:

- Server Objects (URL template for the root and variables to populate it);
- Paths Objects (URL template for the path and parameters); and
- Operation Objects (including parameters).

They are organized as follows:

```
{Server Object}/{Path Object}/?{Query Parameters}
```

Server Objects may be found at the OpenAPI document, Path Object, and Operation Object level. Given this potentially large number of servers, how do you create the valid paths?

We can assume that if a Server Object is included, there must be a reason for its presence. So, the Server Objects with the most restrictive scope are the ones we should use. Clients should look for Server Objects in the following order:

1. The Operation Object,
2. Then Path Item,
3. The root.

The first scope where a Server Object is found dictates the behavior completely.

```
IF Server Objects are supplied
     THEN save them for latter
     ELSE create a Server Object for the host of the OpenAPI document
 DO FOR each Path
     IF Server Objects are included, THEN
         Use them instead of those previously identified
     IF Parameter Objects are included, THEN
         save them for latter
     DO FOR Each Operation
         IF Server Objects are included, THEN
             Use them instead of those previously identified
         IF Parameter Objects are included THEN
             IF this parameter was previously defined
                 THEN replace the previous definition
                 ELSE add this parameter to the set.
         DO FOR Each Server Object
             Extract all URL roots
             DO FOR each URL root
                 Concatenate the URL root and Path to create a working URL
                 Concatenate the working URL with the Parameters
                 Save the completed URL for future use
             CONTINUE
         DONE
     DONE
 DONE
```

Figure 5 — Algorithm for building a URL

A

ANNEX A (INFORMATIVE) REVISION HISTORY

A ANNEX A (INFORMATIVE) REVISION HISTORY

Table A.1

DATE	RELEASE	EDITOR	PRIMARY CLAUSES MODIFIED	DESCRIPTION
2021-06-01	0.1	C. Heazel	all	initial version
2022-11-08	0.2	G. Hobona	all	revision

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] IETF RFC 8288: Web Linking, https://tools.ietf.org/html/rfc8288

[2] IETF RFC 3987: Internationalized Resource Identifiers (IRIs), https://tools.ietf.org/html/rfc3987

[3] IETF RFC 2046: Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types, https://tools.ietf.org/html/rfc2046

[4] Fielding, Roy Thomas: **Architectural Styles and the Design of Network-based Software Architectures**. Doctoral dissertation, University of California, Irvine, 2000, https://www.ics.uci.edu/fielding/pubs/dissertation/fielding_dissertation.pdf

[5] IANA: **Link Relation Types**, https://www.iana.org/assignments/link-relations/link-relations.xml

[6] ISO/IEC 14977:1996(E): **Information technology – Syntactic metalanguage – Extended BNF**, available from ISO.

[7] Whiteside, A., Greenwood, J.: **OGC Web Services Common Standard**, version 2.0, OGC 06-121r9

[8] W3C: **Architecture of the World Wide Web, Volume One**, W3C Recommendation, 15 December 2004, https://www.w3.org/TR/webarch/

[9] W3C: **Data Catalog Vocabulary (DCAT) — Version 2**, W3C Recommendation, 04 February 2020, https://www.w3.org/TR/vocab-dcat-2/

[10] W3C: **Data on the Web Best Practices**, W3C Recommendation, 31 January 2017, https://www.w3.org/TR/dwbp/

[11] W3C/OGC: **Spatial Data on the Web Best Practices**, W3C Working Group Note, 28 September 2017, https://www.w3.org/TR/sdw-bp/

[12] IETF RFC 3986: Uniform Resource Identifier (URI): Generic Syntax. https://tools.ietf.org/html/rfc3986