

OGC API - Maps - Part 1

Core

Open Geospatial Consortium

Submission Date: <yyyy-mm-dd>

Approval Date: <yyyy-mm-dd>

Publication Date: 2019-03-06

External identifier of this OGC® document: <http://www.opengis.net/doc/{doc-type}/{standard}/{m.n}>

Internal reference number of this OGC® document: 20-058

Version: 0.0.1

Category: OGC® Implementation Specification

Editor: Joan Masó

OGC API - Maps - Part 1: Core

Copyright notice

Copyright © 2019 Open Geospatial Consortium

To obtain additional rights of use, visit <http://www.opengeospatial.org/legal/>

Warning

This document is not an OGC Standard. This document is distributed for review and comment. This document is subject to change without notice and may not be referred to as an OGC Standard.

Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Document type:
OGC®ImplementationSpecification
Document subtype: if applicable
Document stage: Draft
Document language: English

License Agreement

Permission is hereby granted by the Open Geospatial Consortium, ("Licensor"), free of charge and subject to the terms set forth below, to any person obtaining a copy of this Intellectual Property and any associated documentation, to deal in the Intellectual Property without restriction (except as set forth below), including without limitation the rights to implement, use, copy, modify, merge, publish, distribute, and/or sublicense copies of the Intellectual Property, and to permit persons to whom the Intellectual Property is furnished to do so, provided that all copyright notices on the intellectual property are retained intact and that each person to whom the Intellectual Property is furnished agrees to the terms of this Agreement.

If you modify the Intellectual Property, all copies of the modified Intellectual Property must include, in addition to the above copyright notice, a notice that the Intellectual Property includes modifications that have not been approved or adopted by LICENSOR.

THIS LICENSE IS A COPYRIGHT LICENSE ONLY, AND DOES NOT CONVEY ANY RIGHTS UNDER ANY PATENTS THAT MAY BE IN FORCE ANYWHERE IN THE WORLD.

THE INTELLECTUAL PROPERTY IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE DO NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE INTELLECTUAL PROPERTY WILL MEET YOUR REQUIREMENTS OR THAT THE OPERATION OF THE INTELLECTUAL PROPERTY WILL BE UNINTERRUPTED OR ERROR FREE. ANY USE OF THE INTELLECTUAL PROPERTY SHALL BE MADE ENTIRELY AT THE USER'S OWN RISK. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR ANY CONTRIBUTOR OF INTELLECTUAL PROPERTY RIGHTS TO THE INTELLECTUAL PROPERTY BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM ANY ALLEGED INFRINGEMENT OR ANY LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR UNDER ANY OTHER LEGAL THEORY, ARISING OUT OF OR IN CONNECTION WITH THE IMPLEMENTATION, USE, COMMERCIALIZATION OR PERFORMANCE OF THIS INTELLECTUAL PROPERTY.

This license is effective until terminated. You may terminate it at any time by destroying the Intellectual Property together with all copies in any form. The license will also terminate if you fail to comply with any term or condition of this Agreement. Except as provided in the following sentence, no such termination of this license shall require the termination of any third party end-user sublicense to the Intellectual Property which is in force as of the date of notice of such termination. In addition, should the Intellectual Property, or the operation of the Intellectual Property, infringe, or in LICENSOR's sole opinion be likely to infringe, any patent, copyright, trademark or other right of a third party, you agree that LICENSOR, in its sole discretion, may terminate this license without any compensation or liability to you, your licensees or any other party. You agree upon termination of any kind to destroy or cause to be destroyed the Intellectual Property together with all copies in any form, whether held by you or by any third party.

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to indicate compliance with any LICENSOR standards or specifications. This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

Table of Contents

1. Scope	7
2. Conformance	8
3. References	10
4. Terms and Definitions	11
4.1. coordinate reference system	11
4.2. coordinate system	11
4.3. geographic information	11
4.4. map	11
4.5. portrayal	11
4.6. Web API	11
5. Conventions	12
5.1. Identifiers	12
5.2. Link relations	12
6. Overview	14
6.1. Evolution from OGC Web Services	14
6.2. Relationship to OGC API - Tiles	15
6.3. How to approach an OGC API	16
7. Requirement Class "Map Core"	19
7.1. Overview	19
7.2. General	19
7.3. Declaration of conformance classes	20
7.3.1. Response	20
7.4. Geospatial resources	20
7.5. Map description	21
7.5.1. Map description response	21
7.6. Maps	22
7.6.1. Operation	22
7.6.2. Parameter styleId	22
8. Requirement Class "Map Styles"	24
8.1. Overview	24
8.2. Declaration of conformance classes	24
8.2.1. Response	24
8.3. Collection	25
8.3.1. Collection Links to styles	25
8.4. Maps description	28
8.5. Maps	28
8.5.1. Operation	28
8.5.2. Parameter styleId	28

8.5.3. Parameter transparent	28
8.5.4. Parameter bgcolor	29
9. Requirement Class "Map from more than one collection"	31
9.1. Overview	31
9.2. Declaration of conformance classes	31
9.2.1. Response	31
9.3. Maps from more than one collection	32
9.3.1. Operation	32
9.3.2. Parameter styles	32
9.3.3. Parameter Collections	33
9.3.4. Response	33
Annex A: Conformance Class Abstract Test Suite (Normative)	34
A.1. Conformance Class A	34
A.1.1. Requirement 1	34
A.1.2. Requirement 2	34
Annex B: Revision History	35

i. Abstract

The draft OGC API - Maps - Part 1: Core specification defines a Web API for requesting map images over the Internet. OGC API - Maps makes it easy for a client to request images, changing parameters such as size and coordinate reference systems at the time of request. A server that implements OGC API - Maps provides information about what maps it offers, as well as producing a map and answering queries about the content of the maps. OGC API - Maps addresses use cases similar to those addressed by the [Web Map Service \(WMS\)](http://portal.opengeospatial.org/files/?artifact_id=14416) [http://portal.opengeospatial.org/files/?artifact_id=14416] standard.

ii. Keywords

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

ogcdoc, OGC document, maps, API, OpenAPI, OGC API

iii. Preface

This document defines the draft OGC API - Maps - Part 1: Core specification. A 'map' is portrayal of geographic information as a digital image file suitable for display on a computer screen ([OGC 06-042](http://portal.opengeospatial.org/files/?artifact_id=14416) [http://portal.opengeospatial.org/files/?artifact_id=14416]). Suggested additions, changes and comments on this standard are welcome and encouraged. Such suggestions may be submitted using the online change request form on OGC web site: http://portal.opengeospatial.org/public_ogc/change_request.php

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. Submitting organizations

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

Organization name(s)

v. Submitters

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

Name	Affiliation
Joan Masó	UAB-CREAF
TBA	TBA
TBA	TBA

Name	Affiliation
TBA	TBA
TBA	TBA

Chapter 1. Scope

This draft specification specifies how to access maps and tiles in a manner independent of the underlying data store. The standard makes use of the [OpenAPI](https://www.openapis.org/) [https://www.openapis.org/] specification and specifies resources for discovery, retrieval and querying of maps from a Web API.

Specifically, this draft OGC API - Maps specification supports the following:

- Discovery operations allow the API to be interrogated to determine its capabilities and to retrieve information (metadata) about this distribution of maps. This includes the API definition as well as metadata about the map layers provided through the API and the Coordinate Reference Systems supported by the API.
- Retrieval operations that allow a client application to get a map from the API
- Query operations that allow a client application to retrieve data (e.g. feature attributes) at a point on a map

Chapter 2. Conformance

This standard defines **TBD** requirements / conformance classes.

The standardization targets of all conformance classes are "Web APIs."

The main requirements class is:

- [Core](#).

The Core specifies requirements that all Web APIs have to implement.

Additional requirements classes include:

- [Map resource](#),
- [Bounding Box Subset](#),
- [Dataset Map](#),
- [Geospatial Data Resource Selection](#) and
- [Geospatial Data Resource Map](#).

The Core does not mandate a specific encoding or format for representing maps. TBD requirements classes depend on the Core and specify representations for resources in commonly used encodings for maps on the web:

- [HTML](#),
- [JPEG](#) and
- [PNG](#).

None of these encodings are mandatory and an implementation of the Core may also decide to implement none of them, but to implement another encoding instead.

That said, the Core requirements class includes recommendations to support, where practical, HTML.

The Core does not mandate any encoding or format for the formal definition of the API either. One option is the OpenAPI 3.0 specification and a requirements class has been specified for OpenAPI 3.0, which depends on the Core:

- [OpenAPI specification 3.0](#).

An implementation of the Core requirements class may also decide to use other API definition representations in addition or instead of an OpenAPI 3.0 definition. Examples for alternative API definitions: OpenAPI 2.0 (Swagger), future versions of the OpenAPI specification, or an OWS Common 2.0 capabilities document.

The Core is intended to be a minimal useful API for fine-grained read-access to maps that are referenced to the coordinate reference system WGS 84 with axis order longitude/latitude.

Additional capabilities may be specified in future parts of the OGC API - Maps series or as vendor-specific extensions.

Conformance with this standard shall be checked using all the relevant tests specified in Annex A (normative) of this document. The framework, concepts, and methodology for testing, and the criteria to be achieved to claim conformance are specified in the OGC Compliance Testing Policies and Procedures and the OGC Compliance Testing web site.

Chapter 3. References

The following normative documents contain provisions that, through reference in this text, constitute provisions of this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

- OGC: OGC 19-072, **OGC API - Common - Part 1: Core** (in development)
<http://docs.openeospatial.org/DRAFTS/19-072.pdf>
- OpenAPI Initiative (OAI). **OpenAPI Specification 3.0** [online]. 2020 [viewed 2020-03-16]. The latest patch version at the time of publication of this standard was 3.0.3, available at <http://spec.openapis.org/oas/v3.0.3>
- ISO and IEC: ISO/IEC 10918-1:1994, **Information technology — Digital compression and coding of continuous-tone still images: Requirements and guidelines** (1994)
<https://www.iso.org/standard/18902.html>
- ISO and IEC: ISO/IEC 15948:2004, **Information technology — Computer graphics and image processing — Portable Network Graphics (PNG): Functional specification** (2004)
<https://www.iso.org/standard/29581.html>
- Internet Engineering Task Force (IETF). RFC 8288: **Web Linking** [online]. Edited by M. Nottingham. 2017 [viewed 2020-03-16]. Available at <http://tools.ietf.org/rfc/rfc8288.txt>
- WHATWG. **HTML**, Living Standard [online, viewed 2020-03-16]. Available at <https://html.spec.whatwg.org/>
- schema.org. **Schema.org** [online, viewed 2020-03-16]. Available at <http://schema.org/docs/schemas.html>

Chapter 4. Terms and Definitions

This document uses the terms defined in Sub-clause 5.3 of [OGC 06-121r8], 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 standard.

This document also uses terms defined in the OGC Standard for Modular specifications ([OGC 08-131r3](https://portal.opengeospatial.org/files/?artifact_id=34762) [https://portal.opengeospatial.org/files/?artifact_id=34762]), 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.

4.1. coordinate reference system

coordinate system that is related to the real world by a datum (source: ISO 19111)

4.2. coordinate system

set of mathematical rules for specifying how coordinates are to be assigned to points (source: ISO 19111)

4.3. geographic information

information concerning phenomena implicitly or explicitly associated with a location relative to the Earth (source: ISO 19101)

4.4. map

portrayal of geographic information as a digital image file suitable for display on a computer screen (source: OGC 06-042)

4.5. portrayal

presentation of information to humans (source: ISO 19117)

4.6. Web API

An Application Programming Interface (API) using an architectural style that is founded on the technologies of the Web (source: OGC 17-069r3)

NOTE

See [Best Practice 24: Use Web Standards as the foundation of APIs](https://www.w3.org/TR/dwbp/#accessAPIs) [https://www.w3.org/TR/dwbp/#accessAPIs] (W3C Data on the Web Best Practices) for more detail.

Chapter 5. Conventions

This section provides details and examples for any conventions used in the document. Examples of conventions are symbols, abbreviations, use of XML schema, or special notes regarding how to read the document.

5.1. Identifiers

The normative provisions in this standard are denoted by the URI <http://www.opengis.net/spec/ogcapi-maps-1/1.0>.

All requirements and conformance tests that appear in this document are denoted by partial URIs which are relative to this base.

5.2. Link relations

To express relationships between resources, [RFC 8288 \(Web Linking\)](#) is used.

The following [registered link relation types \[IANA\]](#) are used in this document.

- **collection**: The target IRI points to a resource which represents the collection resource for the context IRI.
- **describedby**: Refers to a resource providing information about the link's context.
- **next**: Indicates that the link's context is a part of a series, and that the next in the series is the link target.
- **license**: Refers to a license associated with this context.
- **prev**: Indicates that the link's context is a part of a series, and that the previous in the series is the link target.
 - This relation is only used in examples.
- **self**: Conveys an identifier for the link's context.
- **service-desc**: Identifies service description for the context that is primarily intended for consumption by machines.
 - API definitions are considered service descriptions.
- **service-doc**: Identifies service documentation for the context that is primarily intended for human consumption.

In addition the following link relation types are used for which no applicable registered link relation type could be identified.

- **items**: Refers to a resource that is comprised of members of the collection represented by the link's context.
- **conformance**: Refers to a resource that identifies the specifications that the link's context conforms to.

Each resource representation includes an array of links. Implementations are free to add additional links for all resources provided by the API.

Chapter 6. Overview

6.1. Evolution from OGC Web Services

OGC Web Service (OWS) standards have historically implemented a Remote-Procedure-Call (RPC) architectural style using Extensible Markup Language (XML) for payloads over HTTP. This was the state-of-the-art when some of the initial versions of OGC Web Services were originally designed in the late 1990s and early 2000s. This architectural style has now a competing RESTful API style that is proposed as an alternative to RPC pattern. A RESTful API style is resource-oriented instead of service-oriented. This draft OGC API - Maps specification defines a Web API that follows this resource-oriented architecture and in particular 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.

The draft OGC API - Common specification is the foundation upon which OGC APIs will be built, including this draft OGC API - Maps specification. The draft OGC API – Common specification specifies the common kernel of an API approach to services that follows current resource-oriented architecture practices. This common API is to be extended by resource-specific API standards. This draft specification therefore extends OGC API - Common to support Map resources.

Beside the general alignment with the architecture of the Web (e.g., consistency with HTTP/HTTPS, hypermedia controls), another goal for OGC API standards is modularization. This goal has several facets:

- Clear separation between Core requirements and more advanced capabilities. The draft OGC API – Maps - Part 1: Core specification presents the requirements that are relevant for almost everyone who wants to share or use maps at a fine-grained level. Additional capabilities that several communities are using today will be specified as extensions to the Core API.
- Technologies that change more frequently are decoupled and specified in separate modules ("requirements classes" in OGC terminology). This enables, for example, the use/re-use of new encodings for spatial data or API descriptions.
- Modularization is not just about a single "service". OGC APIs will provide building blocks that can be reused in APIs in general. In other words, a server supporting the OGC API - Maps draft specification should not be seen as a standalone service. Rather it should be viewed as a collection of API building blocks that together implement OGC API - Maps capabilities. A corollary for this is that it should be possible to implement an API that simultaneously conforms to conformance classes from the Features, Coverages, Maps, Tiles, and other future OGC API standards.

This approach intends to support two types of client developers:

- Those that have never heard about OGC. Developers should be able to create a client using the API definition without the need to adopt a specific OGC approach (they no longer need to read how to implement a GetCapabilities, allowing them to focus on the geospatial aspects).
- Those that want to write a "generic" client that can access OGC APIs. In other words, they are not specific to a particular API.

As a result of following a RESTful approach, OGC API implementations are not backwards

compatible with OWS implementations per se. However, a design goal is to define OGC APIs in a way that an OGC API interface can be mapped to an OWS implementation (where appropriate). OGC APIs are intended to be simpler and more modern, but still an evolution from the previous versions and their implementations making the transition easy (e.g. by initially implementing facades in front of the current OWS services).

This document provides simple examples throughout the document. The examples are based on a dataset that contains buildings and the API provides access to the datasets via a single feature collection ("buildings") and two encodings: JSON and Hypertext Markup Language (HTML).

6.2. Relationship to OGC API - Tiles

WMS and WMTS share the concept of a map and the capability to create and distribute maps at a limited resolution and size. In WMS the number of rows and columns can be selected by the user within limits and in WMTS the number of rows and columns of the response is predefined in the tile matrix set.

With time, the concept of a tile has been generalized to other data models such as feature data (some vendors use the expression *vector tiles*) and even to coverage data. This draft specification presents an approach to tiles that can be applied to almost every resource type that returns data representations. If applied in conjunction with the OGC API - Features standard and on top of a feature collection, the expected result is tiled feature data. If applied in conjunction with the OGC API - Maps draft specification and on top of a collection that is transformed into a map by applying a style, the result should be map tiles (usually in PNG or JPEG format). The relationship between the Maps and Tiles capabilities of OGC API - Maps and OGC API - Tiles was illustrated by the OGC Testbed-15 initiative as shown in [Figure 1](#).

NOTE

OGC API - Tiles includes a core specification and extensions for defining tile matrix sets, tiles from more than one collection, multi-tiles and multitiles from more than one collection. An info extension is foreseen but not fully developed. In contrast, OGC API - Maps is only partially described based on Testbed-15 requirements. The Maps API is described only to the extent to allow for map tiles to be created on top of a map created by selecting a collection with style or multiple collections with styles. This draft specification contains a section for retrieving a map of an arbitrary number of rows and columns but is not fully formulated. Other extensions for maps are also foreseen. In the future, the WMS SWG could take this document and complete the missing capabilities.

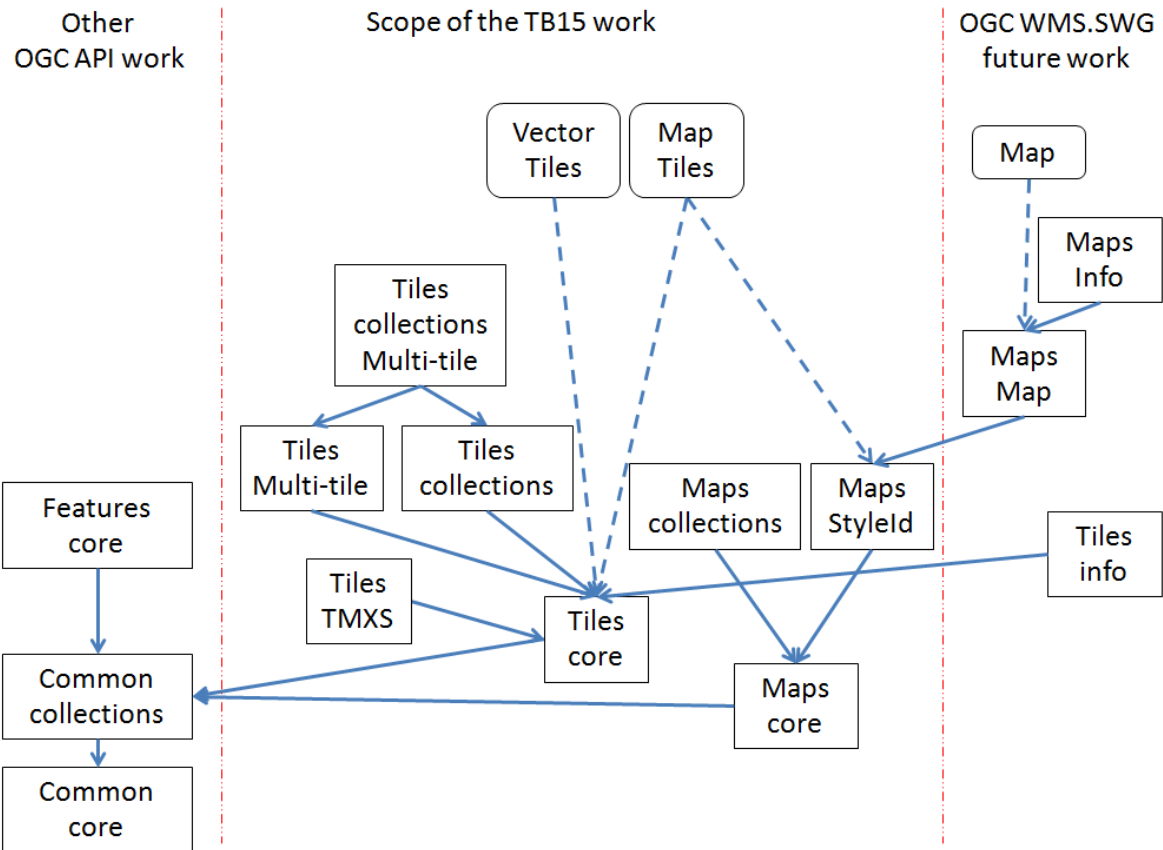


Figure 1. Modular approach in the Maps and Tiles draft specification

6.3. How to approach an OGC API

There are two ways to approach an OGC API.

- Read the landing page, look for links, follow them and discover new links until the desired resource is found
- Read an API definition document that will specify a list of paths to resources.

For the first approach, 'rel' properties identifying link relation types are used to determine the meaning of a relation represented by a link. The following figure illustrates those the links, including others from OGC API - Tiles.

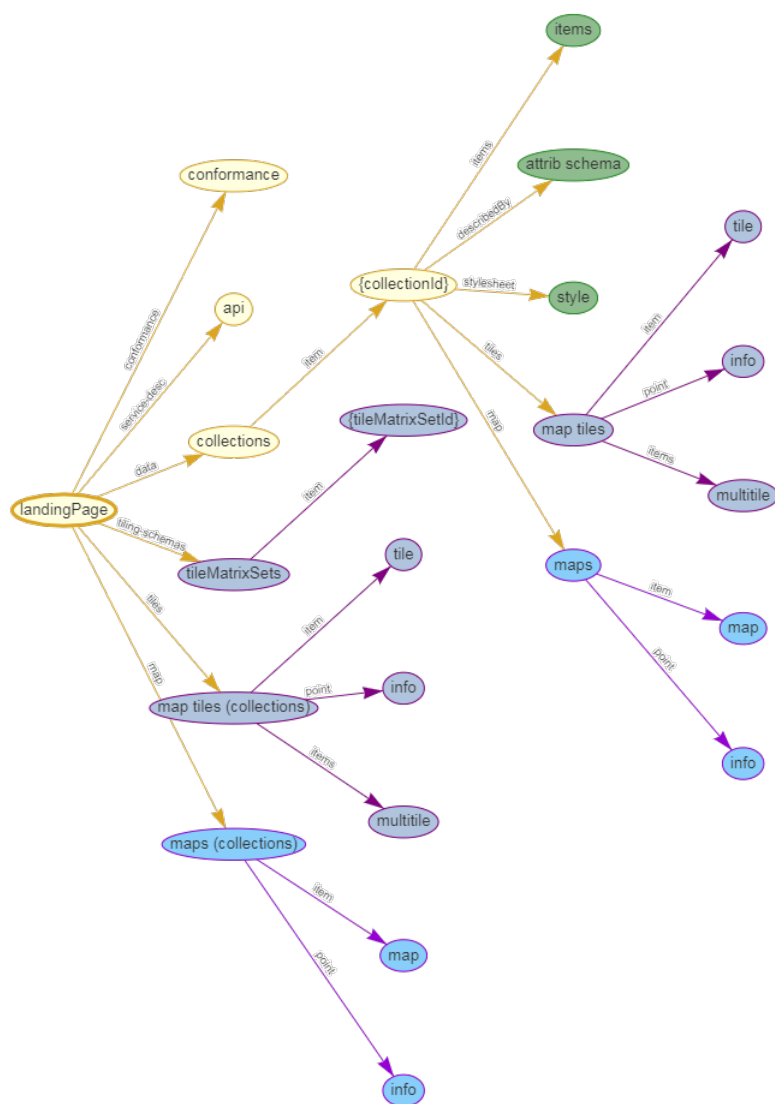


Figure 2. Resources and relations to them via links

For the second approach, the section [\[OpenAPIExamples\]](#) provides some examples of OpenAPI definition documents that enumerate the paths to the necessary resources.

Table 1. Overview of resources and common direct links defined in OGC API - Maps

Resource name	Common path
Landing page	/
Conformance declaration	/conformance
Collections	/collections
Collection	/collections/{collectionId}
Maps description	/collections/{collectionId}/map
Maps description collections ¹	/map

When OGC API - Maps is used along with the OGC API - Tiles specification, the resources listed in [Table 2](#) become available.

Table 2. Overview of map tile resources and common direct links defined in OGC API - Tiles

Resource name	Common path
Tiling Schemas	/tileMatrixSets
Tiling Schema	/tileMatrixSets/{tileMatrixSetId}
Map tiles description	/collections/{collectionId}/map/
Map tiles description collections ¹	/map/tiles
Map tile with style	/collections/{collectionId}/map/{styleId}/tiles/{tileMatrixSetId}/{tileMatrix}/{tileRow}/{tileCol}
Map tile collections ¹	/map/tiles/{tileMatrixSetId}/{tileMatrix}/{tileRow}/{tileCol}

¹: In first column of the table, the word "collections" means "from more than one collection"

Chapter 7. Requirement Class "Map Core"

7.1. Overview

Requirements Class	
http://www.opengis.net/spec/ogcapi-maps-1/1.0/req/core	
Target type	Web API
Dependency	RFC 2616 (HTTP/1.1)
Dependency	RFC 2818 (HTTP over TLS)
Dependency	RFC 8288 (Web Linking)

A map distribution of a dataset is a pictorial representation of geospatial resources. To create a pictorial representation a style is added to the data in the geospatial resource. Styles are defined internally and have a identifier. New styles can be added or modified if the OGC API - Maps draft specification works in combination with the OGC API - Styles draft specification. After associating styles to geospatial resources, a map can be retrieved by specifying a set of parameters that will determine its resolution (width, height, bounding box and CRS) or can be retrieved as tiles.

This section defines the core part of the OGC API - Maps standard that allows defining a map representation for a geospatial resources. To retrieve a fragment of the map, this core needs to be combined with an OGC API - Tiles core standard or with the OGC API - Maps - Bbox extension standard.

To keep the core of the OGC API - Maps standard simple, the core only includes a mechanism to select the default style but it does not define any mechanism to declare, define or select a style other than the default one. The core only assumes that the service is capable of knowing which is the default style while the client ignores all the details about it (including its name).

7.2. General

Recommendation 1	/rec/maps/core/api-common
A	An implementation this standard should consider to implement the requirements specified in the http://www.opengis.net/spec/OAPI_Common/1.0/req/core and http://www.opengis.net/spec/OAPI_Common/1.0/req/collections Requirements Classes of the OGC API-Common version 1.0 Standard.

This building block stays flexible and does not require to implement OWS Common, allowing for other API architectures outside the OGC API framework to adopt it. However, a server under the OGC APIs is expected to implement OGC API Common. If so, in practice, this means that the landing page and the conformance page follow OGC API - Common core and collections requirement classes will be used. Temporarily, it is also possible to combine this building block with OGC API features v1 that is not tied to OWS Common.

7.3. Declaration of conformance classes

To support "generic" clients that want to access multiple OGC API standards and extensions - and not "just" a specific API / server, the API has to declare the requirements classes it implements and conforms to.

7.3.1. Response

The conformance page mainly consists of a list of links.

Requirement 1	/req/maps/core/conformance-success
A	If the API has a mechanism to advertise conformance classes, the API SHALL advertise the maps core conformance class with a link to http://www.opengis.net/spec/ogcapi-maps-1/1.0/req/core .

If the server declares also conformity to OGC API - Common or to OGC API - Features v1, then it has to consider the OGC API - Common requirements for declaring conformance, i.e. the use of a the conformance page. In the JSON format the conformance page is an array of links following the link schema defined in the OGC API - Common or in OGC API - Features v1. Below is an example fragment of a conformance information page of an API conformant to OGC API - Common and OGC API - Maps.

Example 1. Conformance Information Page fragment

```
{
  "conformsTo": [
    "http://www.opengis.net/spec/ogcapi-common-1/1.0/req/core",
    "http://www.opengis.net/spec/ogcapi-common-1/1.0/req/collections",
    "http://www.opengis.net/spec/ogcapi-maps-1/1.0/req/core"
  ]
}
```

7.4. Geospatial resources

This draft specification does not specify how geospatial resources are exposed in the API and if they have the possibility to be retrievable as geospatial data (e.g. feature items). For example OGC API features v1 includes the definition of collections and each collection is available in the /collections/{collectionId} path. OGC API common will provide a similar mechanism. Other paths in the API could also give access to geospatial resources.

NOTE

The concept of geospatial resource path substitutes the concept of "layer" in WMS but it intended to give a better integration between data visualization and data access.

Requirement 2	/req/maps/core/mc-md-collection-links
A	For each collection included in the response, a links property of the collection SHALL include a link to the description of the collection (rel: item) (in addition to other links specified in OGC API Commons).

NOTE

In WMS layers have a hierarchical dependency. At the time of writing this draft specification neither OGC API - Features nor OGC API - Common do not consider his possibility.

7.5. Map description

A map description contains the necessary metadata to enable a client application to formulate a map request from a single geospatial resource.

The maps core defines a **map** resource that is associated with an operation that contains the necessary information to later formulate a map request for a collection. Nevertheless, the core does not require any mandatory information since the map core alone does not specify how to retrieve a map. This core does not mandate a map description operation. The map description cannot be described without considering other OGC API - Maps extensions.

7.5.1. Map description response

This core does not mandate a map description operation. Nevertheless, if it is defined by an OGC API - Maps extension, the core introduces recommendations for having two specific properties in the response of a map description that are inherited from WMS: cascade and opaque.

Recommendation 2	/rec/maps/core/smc-opaque
A	The server may include a boolean property in the maps description response that contains a boolean property opaque .
B	'false' means that map data represents vector features that probably do not completely fill space. 'true' means map data are mostly or completely opaque.
C	If the property is not provided, it should be interpreted as false (the default value)

Recommendation 3	/rec/maps/core/smc-cascaded
A	The server may include a numeric property in the map description response with the name cascaded .

B	0 means that the collection maps have not been retransmitted another map service or API. A positive number indicates how many times the collection map has been retransmitted.
C	If the property is not provided, it should be interpreted as 0 (the default value)

7.6. Maps

This OGC API - Maps core draft specification does not specify how to retrieve a map but it does specify that in order for a map to be retrievable a parameter styleId should be added to any operation that retrieves a map as maps or as tiles.

7.6.1. Operation

Requirement 3	/req/maps/core/mc-map-op
A	Every map SHALL be available as a HTTP GET request to a URI that will be composed by three parts: the first part is the URI of a resource that can be represented as a map, the second part following the pattern /map/{styleId} and the third part completing the retrieval parameters
B	Only the resources (e.g. collection) that advertise one of more links following the pattern .../map/{styleId}... in the maps metadata can be retrieved as maps.

7.6.2. Parameter styleId

Requirement 4	/req/maps/core/mc-styleId-definition
----------------------	---

A	<p>The operation SHALL support a parameter <code>styleId</code> with the characteristics defined (shown as OpenAPI Specification 3.0 fragment)</p> <pre> name: styleId in: path description: 'The styleId that should be included in the map or tile. Each collectionId has a valid list of stylesId. To know the valid styleId values of each collectionId use /collections/{collectionId}.' required: true schema: type: string </pre>
B	<p>A map SHALL be available with <code>default</code> as <code>styleId</code> value. The server decides which is the default style. <code>default</code> is the only value defined by the core and other values might be defined as extensions.</p>

Chapter 8. Requirement Class "Map Styles"

8.1. Overview

Requirements Class	
http://www.opengis.net/spec/ogcapi-maps-1/1.0/req/styles	
Target type	Web API
Dependency	http://www.opengis.net/spec/ogcapi-maps-1/1.0/req/core

The core of the OGC API - Maps draft specification introduces the possibility of creating a map by assigning a style to a resource (e.g. a collectionId) but does not specify how to declare the styles supported by each collection. Only with the core specification an API instance is only capable to request the **default** style and the client does not know anything about it. This requirement class extends the core requirements by specifying how to declare style names other than **default** that can be used to request maps. The OGC API - Styles draft specification (also elaborated in Testbed-15) will allow for the retrieval of the complete information about the style or to send new styles to the server.

8.2. Declaration of conformance classes

8.2.1. Response

The conformance page mainly consists of a list of links. OGC API - Common already requires some links.

Requirement 5	/req/maps/styles/conformance-success
A	The API conformance path SHALL advertise the capability of declaring styles by adding the conformance class for this capability as a link to http://www.opengis.net/spec/ogcapi-maps-1/1.0/req/styles .

In the conformance page (typically in JSON format) the links follow the link schema defined in the OGC API - Common. The following is an example fragment of the response of an OGC API - Maps conformance information page that declares support for the core and the styles extension.

```
{
  "conformsTo": [
    "http://www.opengis.net/spec/ogcapi-common-1/1.0/req/core",
    "http://www.opengis.net/spec/ogcapi-common-1/1.0/req/collections",
    "http://www.opengis.net/spec/ogcapi-maps-1/1.0/req/core"
    "http://www.opengis.net/spec/ogcapi-maps-1/1.0/req/styles"
  ]
}
```

8.3. Collection

This draft specification includes dependencies on OGC API - Common collection. The response to the operation is extended with the necessary information to formulate a map response for this collection.

8.3.1. Collection Links to styles

This extension describes how to provide a list of styles in the collection description.

Requirement 6	/req/maps/styles/smc-styles
A	A successful execution SHALL contain a property called <i>styles</i> that enumerates a list of the styles available for the collection.

B

Each style in *styles* is an object that SHALL conform with the following data mode (shown as OpenAPI Specification 3.0 fragment):

```
type: object
required:
  - id
properties:
  id:
    type: string
    nullable: true
  title:
    type: string
    nullable: true
  links:
    type: array
    nullable: true
    minItems: 1
    items:
      $ref: 'https://api.swaggerhub.com/domains/UAB-
      CREAM/ogc-api-common/1.0.0#/components/schemas/link'
```

```

"styles": [
{
  "id": "night",
  "title": "Topographic night style",
  "links": [
    {
      "href": "https://example.com/api/1.0/styles/night?f=sld10",
      "type": "application/vnd.ogc.sld+xml;version=1.0",
      "rel": "stylesheet"
    },
    {
      "href": "https://example.com/api/1.0/styles/night/metadata?f=json",
      "type": "application/json",
      "rel": "describedBy"
    }
  ]
},
{
  "id": "topographic",
  "title": "Regular topographic style",
  "links": [
    {
      "href": "https://example.com/api/1.0/styles/topographic?f=sld10",
      "type": "application/vnd.ogc.sld+xml;version=1.0",
      "rel": "stylesheet"
    },
    {
      "href": "https://example.com/api/1.0/styles/topographic/metadata?f=json",
      "type": "application/json",
      "rel": "describedBy"
    }
  ]
}
]

```

The mandatory element id can be used as a value for {styleId}.

The optional links element is useful for connecting to an OGC API – Styles implementation that allows for retrieving the styles description.

Recommendation 4	/rec/maps/styles/smc-default-style
A	A successful execution may contain a property called <i>defaultStyle</i> points to the default style used when {styleId} is replaced by the word default

C	The value of the default style SHOULD be one of the ids listed in the property <i>styles</i>
B	Each style in <i>styles</i> is an object that conforms with the following data mode (shown as OpenAPI Specification 3.0 fragment): <div> <pre> default-style: type: string description: the style id of a recommended default style to use for this collection. This is informative and optional. example: 'topographic' </pre> </div>

Example 4. API collection response fragment

```
"defaultStyle": "topographic"
```

8.4. Maps description

The core of the OGC API - Maps draft specification defines **maps** resource that is associated with an operation that contains the necessary information to later formulate a map request for a collection. Nevertheless, the core does not require any mandatory information. This requirement class does not require any mandatory information, but the response of the operation is conditioned by the availability of more than one style per collection.

8.5. Maps

This OGC API - Maps style draft specification extension does not specify how to retrieve a map, but it does specify two parameters (**transparent** and **bgcolor**) in addition to the **styleId** defined in the core.

8.5.1. Operation

8.5.2. Parameter styleId

A part from the **default** style value, this extension introduces the values for the **styleId** that were presented in the **collectionId** definition.

8.5.3. Parameter transparent

Requirement 7	/req/maps/styles/mc-transparent-definition
---------------	--

A	<p>The operation SHALL support an optional parameter transparent to force a transparent background with the characteristics defined (shown as OpenAPI Specification 3.0 fragment)</p> <pre> name: transparent in: query description: 'Background transparency of map (default=true).' required: false style: form explode: false schema: type: boolean default: true </pre>
B	<p>If transparent is not specified, the server will use true.</p>

8.5.4. Parameter bgcolor

Requirement 8	/req/maps/styles/mc-bgcolor-definition
A	<p>The operation SHALL support an optional parameter bgcolor to define a background color with the characteristics defined (shown as OpenAPI Specification 3.0 fragment)</p> <pre> name: bgcolor in: query description: Hexadecimal red-green-blue[-alpha] color value for the background color. If alpha is not specified a binary opacity will be used depending on the transparent parameter. required: false style: form explode: false schema: type: string default: 0xFFFFFF </pre>
B	<p>If bgcolor is not specified, the server is free to choose a background color that's appropriate for the requested style, or 0xFFFFFF (white) if no such information is available.</p>

If the client wants to force an opaque color, apart from defining the appropriate background color

it should ensure that the parameter `transparent` is set to `false`. For the formats that reserve a color to define transparency, it still makes sense to combine background color and `transparent=true` with the purpose of helping the server to select a color that does not interfere with the actual values and colors in the map.

Chapter 9. Requirement Class "Map from more than one collection"

9.1. Overview

Requirements Class	
http://www.opengis.net/spec/ogcapi-maps-1/1.0/req/collections	
Target type	Web API
Dependency	http://www.opengis.net/spec/ogcapi-maps-1/1.0/req/core

In previous clauses maps that are produced from one and only one resource is discussed. This is achieved by concatenating the map path to a resource (e.g. a feature collection). This extension discusses the possibility of combining more than one resource to create a map. This is achieved by using by adding the map path to the root of the service.

9.2. Declaration of conformance classes

9.2.1. Response

The conformance page mainly consists of a list of links. OGC API - Common already requires some links.

Requirement 9	/req/maps/collections/conformance-success
A	The API conformance path SHALL advertise the capability of generating maps from multiple collections by adding the conformance class for this capability as a link to http://www.opengis.net/spec/ogcapi-maps-1/1.0/req/collections .

In the conformance page (typically in JSON format) the links follow the link schema defined in the OGC API - Common. The following is an example fragment of the response of an OGC API - Maps conformance information page


```
{
  "conformsTo": [
    "http://www.opengis.net/spec/ogcapi-common-1/1.0/req/core",
    "http://www.opengis.net/spec/ogcapi-common-1/1.0/req/collections",
    "http://www.opengis.net/spec/ogcapi-maps-1/1.0/req/core",
    "http://www.opengis.net/spec/ogcapi-maps-1/1.0/req/collections"
  ]
}
```

9.3. Maps from more than one collection

9.3.1. Operation

Requirement 10	/req/maps/collections/mcs-op
A	The server SHALL support the HTTP GET operation at the path /maps

9.3.2. Parameter styles

Requirement 11	/req/maps/collections/mcs-styles-definition
A	<p>The operation SHALL support an optional parameter styles with the characteristics defined (shown as OpenAPI Specification 3.0 fragment)</p> <pre>name: styles in: query required: false style: form explode: false schema: type: string</pre>
B	<p>The parameter value SHALL be a list of comma-separated styles identifiers. If the parameter 'collections' exists, the list should be as long as 'collections' and each style identifier corresponds to one collection identifier. Default style can be represented as a blank name or with the default word</p>

C	If the parameter is missing, the default style is assumed for all collections enumerated
---	---

9.3.3. Parameter Collections

Requirement 12	/req/maps/collections/mcs-collections-definition
A	<p>The operation SHALL support an optional parameter collections with the following characteristics (shown as OpenAPI Specification 3.0 fragment)</p> <pre> name: collections in: query required: false style: form explode: false schema: type: array items: type: string </pre>
B	collections SHALL contain a comma-separated list of collection identifiers.
C	Only the collections that advertise a link following the .../map/{styleId}... in the /collections/{collectionId} SHALL be included.
D	Only the collections that support the same CRS or the same tileMatrixSetId parameter value SHALL be included
C	If collections is missing, all collections supporting the crsId or the TileMatrixSetId parameter value will be considered.

9.3.4. Response

To retrieve the map as a map or a tile another extension is needed. No requirements are provided here.

Annex A: Conformance Class Abstract Test Suite (Normative)

NOTE

Ensure that there is a conformance class for each requirements class and a test for each requirement (identified by requirement name and number)

A.1. Conformance Class A

A.1.1. Requirement 1

Test id:	/conf/conf-class-a/req-name-1
Requirement:	/req/req-class-a/req-name-1
Test purpose:	Verify that...
Test method:	Inspect...

A.1.2. Requirement 2

Annex B: Revision History

Date	Release	Editor	Primary clauses modified	Description
2019-03-21	Template	C. Heazel	all	initial template