OGC API-MapsTiles

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: YY-nnnrx

Version: 0.0.1

Category: OGC® Implementation Specification

Editor: Charles Heazel

OGC API Maps Tiles

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

i. Abstract

The OGC has started a focused effort to extend their service standards into the Resource Oriented Architecture world. As part of this effort, this standard defines an API for Map Tiles.

The Map Tile API described in this standard builds on the Web Map Tile Service (WMTS) OGC standard. WMTS provides a scalable, high performance services for web based distribution of cartographic maps. WMTS, in turn, complements earlier efforts to develop services for the web based distribution of cartographic maps. In particular, it compliments the OGC Web Map Service (WMS). WMS focuses on rendering custom maps and is an ideal solution for dynamic data or custom styled maps (combined with the OGC Style Layer Descriptor (SLD) standard). WMTS trades the flexibility of custom map rendering for the scalability possible by serving of static data (base maps) where the bounding box and scales have been constrained to discrete tiles. Note that an API version of WMS is also under development.

ii. Keywords

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

ogcdoc, OGC document, tiling, WMTS

iii. Preface

This document defines an OGC standard for a Web Map Tile API standard. A Map Tile enabled API can serve map tiles of spatially referenced data using tile images with predefined content, extent, and resolution. 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

Chapter 1. Scope

This International Standard specifies how to access maps and tiles in a manner independent of the underlying data store through [OpenAPI](https://www.openapis.org/ [https://www.openapis.org/]). This standard specifies discovery and query operations.

1.1. Current scope:

- Discovery operations allow the API to be interrogated to determine its capabilities and retrieve information (metadata) about this distribution of tiles and maps. This includes the API definition as well as metadata about the feature collections provided through the API and the TileMatrixSets supported by this service.
- Retrieve of maps as defined by the WMS 1.3
- Retrieve of tiles as defined by the WMTS 1.0
- Query about a point in a map or a tile (GetFeatureInfo)
- Retrieve multiple tiles in a single request.

Chapter 2. Conformance

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

The standardization targets of all conformance classes are "web services".

The main requirements class is:

- Core.

The *Core* specifies requirements that all Map Tile APIs have to implement.

TBD requirements classes depend on the *Core* and <enter their purpose here>:

C**apture additional requirements classes here**

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.

In order to conform to this OGC® interface standard, a software implementation shall choose to implement: * Any one of the conformance levels specified in Annex A (normative). * Any one of the Distributed Computing Platform profiles specified in Annexes TBD through TBD (normative).

All requirements-classes and conformance-classes described in this document are owned by the standard(s) identified.

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 API (OAPI) Common Specification https://github.com/opengeospatial/oapi_common (in the process of elaboration)

OGC: OGC 17-083r2, OGC Two Dimensional Tile Matrix Set Standard (2019)

In addition, this standard is deeply inspired in concepts defined in the following documents. This standard offers and alternative interface to fulfill similar tasks included in these references.

OGC and ISO: OGC 06-042 1.3.0 OpenGIS Web Map Service (WMS) Implementation Specification

OGC: OGC 07-057, OpenGIS® Web Map Tile Service Implementation Standard (2010)

OGC: OGC 13-082, OGC® Web Map Tile Service (WMTS) Simple Profile (2016)

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.

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

4.1. term name

text of the definition

Chapter 5. Conventions

This sections 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/{standard}/{m.n}

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

Chapter 6. Overview

6.1. Evolution from OGC Web Services

OGC Web Service (OWS) standards have historically implemented a Remote-Procedure-Call-over-HTTP architectural style using Extensible Markup Language (XML) for payloads. 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 OGC API - Maps and Tiles draft specification specifies an API that follows this Web 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 OGC API – Common draft specification specifies the common kernel of an API approach to services that follows current resource-oriented architecture practices. The draft OGC API - Common specification is the foundation upon which OGC APIs will be built. This common API is to be extended by resource-specific API standards. This draft specification extends OGC API - Common to support Map and Tile 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. This OGC API – Maps and Tiles draft specification presents the requirements that are relevant for almost everyone who wants to share or use Tiled Map Data on 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 - Tiles should not be seen as a standalone service. Rather it should be viewed as a collection of API building blocks which together implement Map and Tile capabilities. A corollary for this is that it should be possible to implement an API that simultaneously conforms to conformance classes from the Feature, Coverage, Map, Tiles, and other future OGC Web 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 for 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. Tiles and maps

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).

In this draft specification the OGC API - Tiles is almost fully described. It includes the a core and extensions for defining tile matrix sets, tiles from more that one collection, multi-tiles and multitiles from more than one collection. And 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.

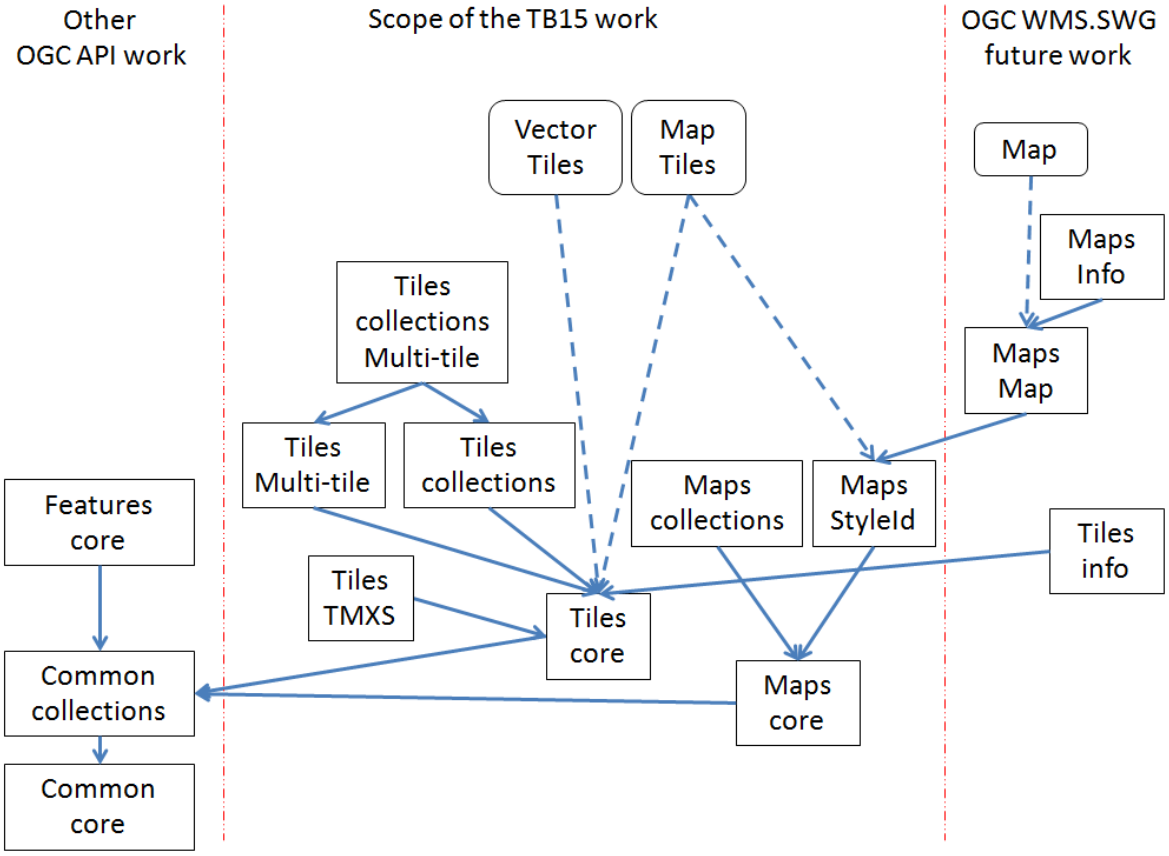

Figure 1. Modular approach in the Maps and Tiles draft specification

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 and API definition document that will specify a list of paths to resources.

For the first approach, many resources in the API include links with rel properties to know the reason for this relation. The following figure illustrates does links.

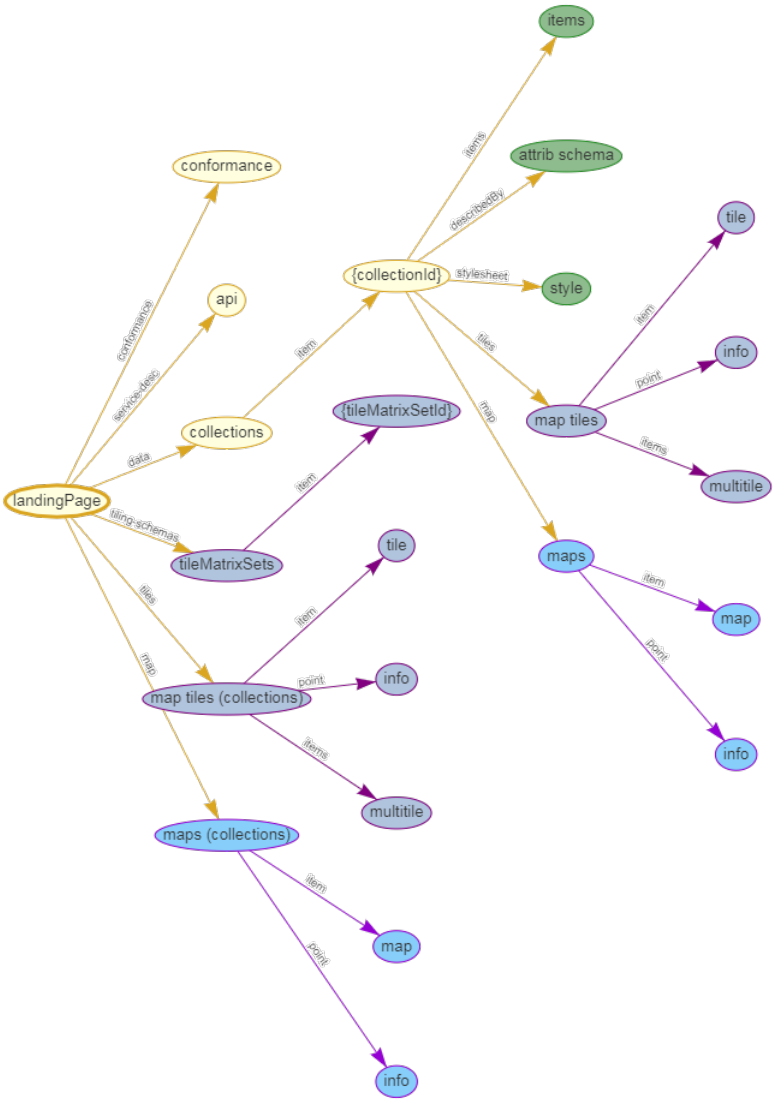

Figure 2. Resources and relations to them via links

For the second approach, the section [OpenAPIExamples] will provide some examples of OpenAPI definition documents that enumerate the paths to get to the necessary resources directly.

Resource name	Common path
Landing page	`/`
Conformance declaration	`/conformance`
Collections	`/collections`
Collection	`/collections/{collectionId}`
Tiling Schemas	`/tileMatrixSets`
Tiling Schema	`/tileMatrixSets/{tileMatrixSetId}`
Tiles	
Vector Tiles description	`/collections/{collectionId}/tiles`

Resource name	Common path
Vector Tiles description from collections	`/tiles`
Vector Tile	`/collections/{collectionId}/tiles/{tileMatrixSetId}/{tileMatrix}/{tileRow}/{tileCol}`
Vector tile collections[1]	`/map/tiles/{tileMatrixSetId}/{tileMatrix}/{tileRow}/{tileCol}`
Vector Multi-tiles	`/collections/{collectionId}/tiles/{tileMatrixSetId}`
Vector Multi-tiles collections[1]	`/tiles/{tileMatrixSetId}`
Map tiles	
Map tiles description	`/collections/{collectionId}/map/`
Map tiles description collections[1]	`/map/tiles`
Map tile	`/collections/{collectionId}/map/{styleId}/tiles/{tileMatrixSetId}/{tileMatrix}/{tileRow}/{tileCol}`
Map tile collections[1]	`/map/tiles/{tileMatrixSetId}/{tileMatrix}/{tileRow}/{tileCol}`
Map tile multi-tiles	`/collections/{collectionId}/map/{styleId}/tiles/{tileMatrixSetId}`
Map tile multi-tiles collections[1]	`/map/tiles/{tileMatrixSetId}`
Maps	
Maps description	`/collections/{collectionId}/map`
Maps description collections[1]	`/map`

Table 1. Overview of resources and common direct links defined in the API

[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 3339 (Date and Time on the Internet: Timestamps)
Dependency	RFC 8288 (Web Linking)
Dependency	http://www.opengis.net/spec/ogcapi-common-1/1.0/req/core
Dependency	http://www.opengis.net/spec/ogcapi-common-1/1.0/req/collections

A map distribution of a dataset is a pictorial representation of the dataset or the collections it has been divided in. To create a pictorial representation a style is added to the data in the collections. 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 (also developed in the Testbed-15). After associating collections to styles, 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 draft specification that allows defining a map representation for a collection. To retrieve a fragment of the map, this section needs to be combined with an OGC API - Tiles draft specification or with the OGC API - Maps - Map extension draft specification.

To keep the core of the OGC API - Maps draft specification simple, only includes a mechanism to select the default style but it does not define any mechanism to define or select a style other than the default one. The core specification 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

Requirement 14	**/req/maps/core/api-common**
A	The OGC API SHALL comply with the requirements specified in the http://www.opengis.net/spec/OAPI_Common/1.0/req/core and collections Requirements Classes of the OGC API-Common version 1.0 Standard.

In practice, this means that the landing page and the conformance page follow OGC API - Common core and collections requirements. This draft specification provides additions to the OGC API - Common requirements that are particular to maps use case.

7.3. API landing page

The landing page provides links to start exploring the resources offered by the API instance. It mainly consists of a list of links. OGC API - Common already requires some common links that are enough for this core.

7.3.1. Response

There are no required variations to the landing page.

7.4. 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.

7.4.1. Response

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

Requirement 15	**/req/maps/core/conformance-success**
A	The API conformance path SHALL advertise the maps core conformance class as links to http://www.opengis.net/spec/ogcapi-maps-1/1.0/req/core.

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.

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"
  ]
}
```

7.5. Collections

This draft specification includes dependencies on OGC API - Common collections. Collections are mandatory in the core of this draft specification because collections are the object that can eventually be included in a map.

Collections enumerate the collectionId identifiers available in this API as well as basic information about each collectionId: id, title, description, extent, crs and links. This common response is considered enough for a general description of the collection.

Requirement 16	**/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).

More specific details about the collection can be found following the link to the individual collections that follow the pattern /collections/{collectionId}

NOTE The collectionId substitutes the concept of "layer" in WMS.

NOTE In WMS layers have a hierarchical dependency. The authors believe it is the responsibility of OGC API - Common to provide this functionally. At the time of writing this draft specification the OWS Common SWG has not consider his possibility yet.

7.6. Maps description

The maps core defines a `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 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.

7.6.1. 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 4	**/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 5	**/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)

7.7. 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.

7.7.1. Operation

Requirement 17	**/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.

7.7.2. Parameter styleId

Requirement 18	**/req/maps/core/mc-styleId-definition**
A	The operation SHALL support a parameter `styleId` with the characteristics defined (shown as OpenAPI Specification 3.0 fragment)

```
   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
```

B	A map SHALL be available with `default` as `styleId` value. The server decides which is the default style. `default` is the only value defined by the core and other values might be defined as extensions.

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 extents 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 19	**/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.

Example 2. 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"
    "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 20	**/req/maps/styles/smc-styles**
A	A successful execution SHALL contain a property called *styles* 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-CREAF/ogc-api-common/1.0.0#/components/schemas/link'`

Example 3. API collection response fragment

```
  "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 6	/rec/maps/styles/smc-default-style
A	A successful execution may contain a property called *defaultStyle* 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 *styles*

B	Each style in *styles* is an object that conforms with the following data mode (shown as OpenAPI Specification 3.0 fragment):
	`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'`

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 21	**/req/maps/styles/mc-transparent-definition**

A	The operation SHALL support an optional parameter `transparent` to force a transparent background with the characteristics defined (shown as OpenAPI Specification 3.0 fragment)
	
	`name: transparent`
	`in: query`
	`description:`
	`'Background transparency of map (default=true).'`
	`required: false`
	`style: form`
	`explode: false`
	`schema:`
	`type: boolean`
	`default: true`

B	If `transparent` is not specified, the server will use `true`.

8.5.4. Parameter bgcolor

Requirement 22	**/req/maps/styles/mc-bgcolor-definition**

A	The operation SHALL support an optional parameter `bgcolor` to define a background color with the characteristics defined (shown as OpenAPI Specification 3.0 fragment)
	
	`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`

B	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.

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 form 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 23	/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

Example 5. 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"
    "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 24	**/req/maps/collections/mcs-op**
A	The server SHALL support the HTTP GET operation at the path /maps

9.3.2. Parameter styles

Requirement 25	**/req/maps/collections/mcs-styles-definition**

A	The operation SHALL support an optional parameter `styles` with the characteristics defined (shown as OpenAPI Specification 3.0 fragment)
	
	`name: styles`
	`in: query`
	`required: false`
	`style: form`
	`explode: false`
	`schema:`
	`  type: string`

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

9.3.3. Parameter Collections

Requirement 26	**/req/maps/collections/mcs-collections-definition**

A	The operation SHALL support an optional parameter `collections` with the following characteristics (shown as OpenAPI Specification 3.0 fragment)
	
	`name: collections`
	`in: query`
	`required: false`
	`style: form`
	`explode: false`
	`schema:`
	`  type: array`
	`  items:`
	`  type: string`

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