OGC API-MapsTiles

Table of Contents

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

Copyright notice

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.

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/). 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 implemented a Remote-Procedure-Call-over-HTTP architectural style using XML for payloads. This was the state-of-the-art when OGC Web Services were originally designed in the late 1990s and early 2000s. This architectural style has fallen out of favor. It has been replaces by a RESTful API style which is resource oriented instead of service oriented. This Map Tiles API standard 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 (OAPI) Common specifies the common kernel of an API approach to services that follows current resource-oriented architecture practices. OAPI Common is the foundation upon which OGC APIs will be built. This common API is to be extended by resource-specific API standards. This specification extends OAPI Common to support Map 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 document specifies the requirements that are relevant for almost everyone who wants to share or use Map Tiles 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 Map Tiles API should not be seen as a standalone service. Rather it should be viewed as a collection of API building blocks which together implement Map Tile capabilities. A corollary of 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 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 read an OGC standard (they may need to learn a little bit about geometry, etc.);
- Those that want to write a "generic" client that can access OGC APIs, i.e. are not specific for a particular API.

As a result of this modernization, OGC API implementations are not backwards compatible with OWS implementations per se. However, a design goal is to define OGC APIs in a way so that an OAPI interface can be mapped to a OWS implementation. OGC APIs are intended to be simpler and more modern, but still an evolution from the previous versions and their implementations.

6.2. Encodings

NOTE	Discuss if different encodings of map tiles are anticipated.

6.3. Examples

NOTE	Update the following content once a Map Tile example has been agreed on.

This document uses a simple example throughout the document: The dataset contains buildings and the API provides access to them through a single feature collection ("buildings") and two encodings, GeoJSON and HTML.

The buildings have a few (optional) properties: the polygon geometry of the building footprint, a name, the function of the building (residential, commercial or public use), the floor count and the timestamp of the last update of the building feature in the dataset.

This example serves to illustrate the concepts underlying OGC APIs. It does not indicate that OGC APIs are always feature based. Other resource types can and will be implemented as well. But the basic capapablities described in this specification will apply to all.

Chapter 7. Requirement Class "Tiles Core"

7.1. Overview

Requirements Class	
http://www.opengis.net/spec/ogcapi-tiles-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/tilematrixset/1.0/req/tilematrixset2d
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

An API that implements this conformance class provides access to tiled resources of a dataset. In other words, the API is a distribution of that dataset. The OGC API features, for example, would be another distribution.

The entry point is a `Landing page` (path `/`).

The `Landing page` provides links to * the `API definition` (path `/api`, link relation `service`), * the `Conformance declaration` (path `/conformance`, link relation `conformance`), and * the `Collections` (path `/collections`, link relation `data`).

The `API definition` describes the capabilities of the API that can be used by clients to retrieve resources from the API or by development tools to support the implementation of API servers and clients. Accessing the `API definition` using HTTP GET returns a description of the API.

The `Conformance declaration` states the requirements classes from standards or community specifications, identified by a URI, that the API conforms to. Clients can, but are not required to, use this information. Accessing the `Conformance declaration` using HTTP GET returns the list of URIs of requirements classes implemented by the API.

The tiles core does not mandate the inclusion of an explicit definition of any TileMatrixSet. The standard assumes that clients and services know about the eight TileMatrixSets defined in OGC 17-083r2 annex D and there is no need to communicate these definitions. An extension to the core provides the capability to include definitions of flexible TileMatrixSets.

This standard assumes that data is organized into one or more collections. `Collections` provides information about and access to the collections.

This document does not specify requirements for collections and they can consist on features, coverages, a resource that does not represent data per-se (e.g. an annotation) any other resource that can be represented in a tile. collectionId replaces the concept of layer in WMS and WMTS. Maps or tiles can be generated from one collection (or a combination of collections as an

extension).

Accessing `Collections` using HTTP GET returns a response that contains at least the list of collections. Accessing `Collections/{collectionId}` using HTTP GET returns a description of a collection with an indication if the collection can be retrieved as a map or a tile or both. Accessing the items of a collections is out of the scope of this standard and you need to refer to other document standard describing OGC APIs for features or coverages. For each `Collection`, a link to metadata about the collection is available (path `/collections/{collectionId}`) with key information about the collection. This information includes:

- A local identifier for the collection that is unique for the dataset;
- An optional title and description for the collection;
- An optional extent that can be used to provide an indication of the spatial and temporal extent of the collection - typically derived from the data;
- A list of TileMatrixSetLink to the available tiling schemas supported by the collection (From the linked TileMatrixSet member the client can determine the coordinate reference systems (CRS) in which tiles may be returned by the API)

The `Collection` resource is available at path `/collections/{collectionId}`, often with more details than included in the `Collections` response.

7.2. General

Requirement 1	**/req/tiles/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 http://www.opengis.net/spec/OAPI_Common/1.0/req/collections Requirements Classes of the OGC API-Common version 1.0 Standard.

In practice, this means that the landing page and the conformance page follows OGC API Common core and collections requirements (STILL TBD but mostly equivalent to the more general parts of WFS3.0 requirements but with the text generalized to other resource types). This standard provides extra additions to the OWS common requirements that are particular to tiles.

7.3. API landing page

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

7.3.1. Response

No variations are required in the landing page. With a /collections successful response it is possible to retrieve the list of collectionId and links to the /collections/{collectionId}. With a

/collections/{collectionId} successful response, it is possible to discover the links to retrieve some tiles. The other tiles can be retrieved by following the /collections/{collectionId}/tiles/{tileMatrixSetId}/{tileMatrix}/{tileRow}/{tileCol} template. Note that other resources an also be retrieved as collections.

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 in a list of links. OWS Commmon already requires some links.

Requirement 2	**/req/tiles/core/conformance-success**
A	The API conformance path SHALL advertise the tiles core conformance class as links to http://www.opengis.net/spec/ogcapi-tiles-1/1.0/req/core.

In the conformance page (commonly in JSON format) the links are following the link schema defined in the OGC API Common. The following is an example fragment of the response of a OGC API tiles 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-tiles-1/1.0/req/core"
  ]
}
```

7.5. Collections

TBD: What will happen with collections of coverages?

This standard includes dependencies on OWS Common collections. Collections are mandatory in the core of this standard because collections are the object that will be included in a tile.

Collections will enumerate the collectionId 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 3	/req/tiles/core/tc-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 individual collection (relation: `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 WMTS.

7.6. Collection

This standard includes dependencies on OWS Common collection. The response of the operation is extended with a new link for the tiles description.

7.6.1. Collection Links

Requirement 4	/req/tiles/core/stc-tiles-example
A	If the resource can be retrieved as tiles, a `200`-response SHALL include at least a link to a tiles description for this resource.
B	These links SHALL provide a URL template with the fragment `/tiles`.

7.7. Tiles description

The response of this operation contains the necessary information to later formulate a tile request for a collection.

7.7.1. Operation

Requirement 5	/req/tiles/core/sct-op

A	Every resource available as tiles SHALL support an operation to retrieve the description of the tiles it can provide, available as a HTTP GET request to a URI that will be composed by two parts: a initial part is the URI of a resource that can be represented as tiles and the final part follows the pattern /tiles. Only the resources or collection that supports this operation can be retrieved as tiles.

The request of this operation has no parameters.

7.7.2. Response

A successful response of a tiles request for a collection that can be retrieved as tiles will respond with a data structure with specific information necessary to get tiles representing the resource collection. In this core, the response only informs about in which tile matrix sets tiles can be retrieved and the URL template to a tile.

Requirement 6	**/req/tiles/core/stc-tmxslink**

A	The content of the response of a successful execution SHALL contain a property called *tileMatrixSetLinks* with a list of *tileMatrixSetLink* objects following a data model defined in the clause 7.3 of OGC 17-083r2. In the core *tileMatrixSetLink* is only used for referencing the supported TileMatrixSets for the tiles limiting it to the following schema (expressed in OpenAPI Specification 3.0 fragment):
	
	`tileMatrixSetLink-set:`
	`  description: This list tileMatrixSetLink as defined in OGC 17-083r2 supported by this collectionId.`
	`  type: array`
	`  items:`
	`  $ref: '#/components/schemas/tileMatrixSetLink-entry'`
	`tileMatrixSetLink-entry:`
	`  type: object`
	`  required:`
	`  - tileMatrixSet`
	`  properties:`
	`  tileMatrixSet:`
	`  type: string`
	`  example: 'WebMercatorQuad'`
	`  tileMatrixSetURI:`
	`  type: string`
	`  format: uri`
	`  example: 'http://www.opengis.net/def/tilematrixset/OGC/1.0/WebMercatorQuad'`

C	This core does not provide any mechanism to defined TileMatrixSets so if this mechanism is not provided in an extension, the tileMatrixSetURI should point to one of the 8 URIs defined in the OGC 17-083r2 Annex D.

Example 2. Example of a tiles property in the tiles response.

```
{
  "tileMatrixSetLinks": [
    {
      "tileMatrixSet": "WebMercatorQuad",
      "tileMatrixSetURI":
"http://www.opengis.net/def/tilematrixset/OGC/1.0/WebMercatorQuad"
    }
  ]
  ...
}
```

Requirement 7	**/req/tiles/core/sct-tile-examples**
A	The content of the response of a successful execution SHALL include at least a link to a tile URI template (relation: `tiles`).
B	These links SHALL provide a URL template with the fragment `/tiles` followed by the variables {tileMatrixSetId}, {tileMatrix}, {tileRow} and {tileCol}. Once the variables are substituted by their respective valid values, a URL to a tile is obtained.
C	There SHALL be a link to a tile URI template for each format that the server supports (the format is indicates in the 'type' attribute of the link)

One common order used in URL templates for tiles is .../tiles/{tileMatrixSetId}/{tileMatrix}/{tileRow}/{tileCol}, but this standard allows for other URL template dispositions.

Table 1. URI template variables for tiles and possible values

URL template variable	**Meaning**	**Possible values**	**TileMatrixSetId**
tile matrix set identifier	the ones included in Annex D of OGC 17-083r2 or defined by extensions of this core.	TileMatrix	tile matrix identifier

URL template variable	Meaning	Possible values	TileMatrixSetId
identifier of the tile matrix (representing a zoom level, a.k.a. a scale) listed in the TileMatrixSet definition	TileRow	row index of tile matrix	a non-negative integer between 0 and the MatrixHeight – 1. If there is a TileMatrixSetLimits the value is limited between MinTileRow and MaxTileRow

Example 3. API tiles response fragment

```
links:
[
    {
      "href":
"http://data.example.com/collections/buildings/tiles/{tileMatrixSetId}/{tileMatrix
}/{tileRow}/{tileCol}",
      "rel": "tiles",
      "type": "image/png",
    }
]
```

7.8. Tiled data from one collection

The core provides a mechanism to select and retrieve a tile from a TileMatrixSet. If the service does not advertise any other TileMatrixSet (this core does not provide any mechanism to do that, but extensions can do it) only the TileMatrixSet identifiers specified in the Annex D.1 of the OGC 17-083r2 standard can be used.

7.8.1. Operation

Requirement 8	/req/tiles/core/tc-op
A	Every tile SHALL be available as a HTTP GET request to a URI that will be composed by two parts: a initial part is the URI of a resource that can be represented as tiles and the final part follows the pattern /tiles/{tileMatrixSetId}/{tileMatrix}/{tileRow}/{tileCol}

Typical resources that can be retrieved as tiles are: features (/collections/{collectionId}), coverages (/collections/{collectionId}/coverage/{coverageId} or /coverage/{coverageId}) or maps (/collections/{collectionId}/map/styleId}).

7.8.2. Parameter tileMatrixSetId

Unresolved directive in clause_7_tile_core.adoc - include::requirements/tiles/core/REQ_tc-tilematrixsetid-definition.adoc[]

7.8.3. Parameter tileMatrix

Unresolved directive in clause_7_tile_core.adoc - include::requirements/tiles/core/REQ_tc-tilematrix-definition.adoc[]

7.8.4. Parameter tileRow

Unresolved directive in clause_7_tile_core.adoc - include::requirements/tiles/core/REQ_tc-tilerow-definition.adoc[]

7.8.5. Parameter tileCol

Unresolved directive in clause_7_tile_core.adoc - include::requirements/tiles/core/REQ_tc-tilecol-definition.adoc[]

7.8.6. Response

A successful response of a tile request will be consistent with the media type of resource requested. This standard does not impose any media type. For example; for features can be geojson of Mapbox vector tiles; coverages it my be a geotiff and for maps it can be a jpeg or a png.

Requirement 9	**/req/tiles/core/tc-success**
A	A successful execution of the operation SHALL be reported as a response with a HTTP status code `200`.
B	The content of that response SHALL be consistent with the format requested and represent elements inside or intersecting with the spatial extent of the geographical area of the tile identified by TileMatrixSet, TileMatrix, TileRow and TileCol.

Normally, the content partially outside the tile bounding box will be clipped and this is particularly true when tiles are in raster format. Nevertheless, tiles containing features in vector format may not clip the features partially outside.

Recommendation 1	**/rec/tiles/core/tc-success-scale**
A	The content of that response should be simplified to comply with the scale denominator represented by the TileMatrix identified. Full resolution geographical elements will only be provided for the lower values of scale denominators.

7.8.7. Error situations

A general summary of the HTTP status codes can be found in OWS Common.

If the parameter value `tileMatrixSetId` is not available by the server for this resource or the parameters values `tileMatrix`, `tileRow`, `tileCol` are out-of-range, the status code of the response will be 404.

Chapter 8. Requirement Class "Tiles from more than one collection"

8.1. Overview

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

In previous clauses we have been discussion about tiles that are produced from one and only one resource. This is achieved by concatenating the tile path to a resource (e.g. a feature collection). This extension discusses the combination of more than one resource to create a tile. This is achieved by adding the tile path to the root of the service.

8.2. API landing page

The landing page provides links to start exploring the resources offered by the API. It mainly consists of a list of links. The core of this standard does not add anything to the links required by OWS Common. This extension for TileMatrixSet requires new one for the description of the tiles from more than one collection on top of the common ones.

8.2.1. Response

Requirement 10	/req/tiles/collections/root-success
A	The API SHALL advertise a URI to retrieve tiles definitions defined by this service as links to the descriptions paths with rel=tiles.

In the landing page, in JSON format, the links are following the link schema defined in the OGC API Common. The following is an example fragment of the response of a OGC API tiles landing page

Example 4. API Landing Page fragment

```
{
  links: [
    ...,
    {
       "href": "http://data.example.org/tiles",
       "rel": "tiles",
       "type": "application/json",
       "title": "Link to information on map tiles combining more than one
collection",
    }
  ]
}
```

8.3. Declaration of conformance classes

8.3.1. Response

The conformance page mainly consists in a list of links. OWS Commmon already requires some links.

Requirement 11	**/req/tiles/collections/conformance-success**
A	The API conformance path SHALL advertise the capability of generating tiles from multiple collections adding the conformance class for this capability as a link to http://www.opengis.net/spec/ogcapi-tiles-1/1.0/req/collections.

In the conformance page (commonly in JSON format) the links are following the link schema defined in the OGC API Common. The following is an example fragment of the response of a OGC API tiles 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-tiles-1/1.0/req/core"
    "http://www.opengis.net/spec/ogcapi-tiles-1/1.0/req/collections"
  ]
}
```

8.4. Tiles description

The response of this operation contains the necessary information to later formulate a tile request from more than one collection.

8.4.1. Operation

Requirement 12	**/req/tiles/collections/ts-op**
A	The server SHALL support an operation to retrieve the description of the tiles from more than one collection, available as a HTTP GET request to a URI that will be composed by two parts: a initial part is the URI of a resource that can be represented as tiles (e.g. `/map` or simply `/`) and the final part follows the pattern `/tiles`.

The request of this operation has no parameters.

8.4.2. Response

A successful response of a tiles request for more than one collection will respond with a data structure with specific information necessary to get tiles representing the resource collection. In this extension, the response only provides the URL template to retrieve a tile.

Requirement 13	**/req/tiles/collections/ts-tile-examples**
A	The content of the response of a successful execution SHALL include at least a link to a tile URI template (relation: `tiles`).
B	These links SHALL provide a URL template with the fragment `/tiles` followed by the variables {tileMatrixSetId}, {tileMatrix}, {tileRow} and {tileCol}. Once the variables are substituted by their respective valid values, a URL to a tile is obtained.
C	There SHALL be a link to a tile URI template for each format that the server supports (the format is indicates in the 'type' attribute of the link)

One common order used in URL templates for tiles is ... /tiles/{tileMatrixSetId}/{tileMatrix}/{tileRow}/{tileCol}, but this standard allows for other URL template dispositions.

Table 2. URI template variables for tiles and possible values

URL template variable	Meaning	Possible values	TileMatrixSetId
tile matrix set identifier	the ones included in Annex D of OGC 17-083r2 or defined by extensions of this core.	TileMatrix	tile matrix identifier
identifier of the tile matrix (representing a zoom level, a.k.a. a scale) listed in the TileMatrixSet definition	TileRow	row index of tile matrix	a non-negative integer between 0 and the MatrixHeight – 1. If there is a TileMatrixSetLimits the value is limited between MinTileRow and MaxTileRow

Example 6. API tiles response fragment

```
links:
[
    {
      "href":
"http://data.example.com/tiles/{tileMatrixSetId}/{tileMatrix}/{tileRow}/{tileCol}"
,
      "rel": "tiles",
      "type": "image/png",
    }
]
```

In general, the `tileMatrixSetLinks` and the `tileMatrixSetLimits` can be determined by examine this information in the individual collections. In some cases the server could also include the `tileMatrixSetLinks` data structure as part of the response of this operation. Clients should be prepared to determined by examine the tileMatrixSet values and limits from the information in the individual collections if a `tileMatrixSetLinks` data structure is not provided as a response here.

8.5. Tiles from more than one collection

8.5.1. Operation

Requirement 14	/req/tiles/collections/tcs-op
A	The server SHALL support the HTTP GET operation at the path /tiles/{tileMatrixSetId}/{tileMatrix}/{tileRow}/{tileCol}

8.5.2. Parameter tileMatrixSetId

Unresolved directive in clause_7_tile_collections.adoc - include::requirements/tiles/collections/REQ_tcs-tilematrixsetid-definition.adoc[]

8.5.3. Parameter tileMatrix

Unresolved directive in clause_7_tile_collections.adoc - include::requirements/tiles/collections/REQ_tcs-tilematrix-definition.adoc[]

8.5.4. Parameter tileRow

Unresolved directive in clause_7_tile_collections.adoc - include::requirements/tiles/collections/REQ_tcs-tilerow-definition.adoc[]

8.5.5. Parameter tileCol

Unresolved directive in clause_7_tile_collections.adoc - include::requirements/tiles/collections/REQ_tcs-tilecol-definition.adoc[]

8.5.6. Parameter Collections

Requirement 15	**/req/tiles/collections/tcs-collections-definition**

A	The operation SHALL support an optional parameter `collections` with the following characteristics (using an 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 type=tiles in the /collections/{collectionId} SHALL be included.
D	Only the collections that support the same TileMatrixSetId parameter value SHALL be included

E	If `collections` is missing, the service SHALL create a representation based collections on all or some of the collections supporting the TileMatrixSetId parameter value.

Recommendation 2	**/req/tiles/collections/tcs-collections-definition**
A	If `collections` is missing and when it is possible and sensible, all collections SHOULD be represented in the tiles
B	If `collections` is missing, the collections present in the tiles SHOULD be listed in the description of the parameter in the API definition

Permission 1	**/per/tiles/collections/tcs-collections-definition**
A	If `collections` is missing and if it is not possible and sensible to represent all collections in tiles (e.g. it compromises performance or tiles are become packed with too much elements), the server is allowed to select a subset of the collections for simplification purposes

8.5.7. Response

A successful response of a tile request will be consistent with the media type of resource requested. This standard does not impose any media type. For example; for features can be geojson of Mapbox vector tiles; coverages it may be a geotiff and for maps it can be a jpeg or a png.

Requirement 16	**/req/tiles/collections/tcs-success**
A	A successful execution of the operation SHALL be reported as a response with a HTTP status code `200`.
B	The content of that response SHALL be consistent with the format requested and represent elements inside or intersecting with the spatial extent of the geographical area of the tile identified by TileMatrixSet, TileMatrix, TileRow and TileCol.
C	The content of that response SHALL be simplified to comply with the scale denominator represented by the TileMatrix identified. Full resolution geographical elements will only be provided for the lower values of scale denominators.

8.5.8. Error situations

If the parameter value `tileMatrixSetId` is not available by the server for this resource or the parameters values `tileMatrix`, `tileRow`, `tileCol` are out-of-range, the status code of the response will be 404.

If the parameter value of `collections` does not exist on the server, the status code of the response will be 404.

If the parameter value of `collections` has a wrong format or combines collections and some of them are not compatible with the `tileMatrixSetId` value, the status code of the response will be 500.

Chapter 9. Requirement Class "Tiles Tile Matrix Set"

9.1. Overview

Requirements Class	
http://www.opengis.net/spec/ogcapi-tiles-1/1.0/req/tmxs	
Target type	Web API
Dependency	http://www.opengis.net/spec/ogcapi-tiles-1/1.0/req/core
Dependency	http://www.opengis.net/spec/tilematrixset/1.0/req/json-tilematrixset2d
Dependency	http://www.opengis.net/spec/tilematrixset/1.0/req/tilematrixsetlimits2d
Dependency	http://www.opengis.net/spec/tilematrixset/1.0/req/json-tilematrixsetlimits2d
Dependency	http://www.opengis.net/spec/tilematrixset/1.0/req/tilematrixsetlink2d
Dependency	http://www.opengis.net/spec/tilematrixset/1.0/req/json-tilematrixsetlink2d

The *tiles core* clause states that if there is no information on TileMatrixSet, the service can behave as a service that supports the eight TileMatrixSets defined in the Annex D.1 of the OGC 17-083r2 standard. This extension adds all necessary elements to support other TileMatrixSets.

The entry point is a `Landing page` (path `/`).

The `Landing page` provides links to * the `API definition` (path `/api`, link relation `service`), * the `Conformance declaration` (path `/conformance`, link relation `conformance`), and * the `Collections` (path `/collections`, link relation `data`). * the `TileMatrixSets` (path `/tileMatrixSets`, link relation `tileMatrixSet`).

9.2. API landing page

The landing page provides links to start exploring the resources offered by the API. It mainly consists in a list of links. The core of this standard does not add anything to the links required by OWS Common. This extension for TileMatrixSet requires new ones for TileMatrixSets on top of the common ones.

9.2.1. Response

Requirement 17	/req/tiles/tmxs/root-success
A	The API SHALL advertise a URI to retrieve the list of TileMatrixSets defined by this service as links to the descriptions paths with rel=tileMatrixSets..

In the landing page, in JSON format, the links are following the link schema defined in the OGC API Common. The following is an example fragment of the response of a OGC API tiles landing page

Example 7. API Landing Page fragment

```
{
  links: [
    ...,
    {
      "href": "http://data.example.org/tileMatrixSet?f=json",
      "rel": "tileMatrixSets",
      "type": "application/json",
      "title": "List of tileMatrixSets implemented by this API in JSON",
    },
    {
      "href": "http://data.example.org/tileMatrixSet?f=html",
      "rel": "tileMatrixSets",
      "type": "text/html",
      "title": "List of tileMatrixSets implemented by this API in HTML",
    }
  ]
}
```

With a `/collections` successful response it is possible to retrieve the list of collectionId and links to the /collections/{collectionId}. With a /collections/{collectionId} successful response is s possible to discover the links to retrieve some tiles. The other tiles can be retrieved by following the /collections/{collectionId}/tiles/{tileMatrixSetId}/{tileMatrix}/{tileRow}/{tileCol} template. Other tile paths are defined for other resource types that are not collections.

9.3. Declaration of conformance classes

9.3.1. Response

The conformance page mainly consists in a list of links. OWS Commmon already requires some links.

Requirement 18	**/req/tiles/tmxs/conformance-success**
A	The API conformance path SHALL advertise the capability of generating tiles from multiple collections adding the conformance class for this capability as a link to http://www.opengis.net/spec/ogcapi-tiles-1/1.0/req/tmxs.

In the conformance page (commonly in JSON format) the links are following the link schema defined in the OGC API Common. The following is an example fragment of the response of a OGC API tiles conformance information page

Example 8. 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-tiles-1/1.0/req/core"
    "http://www.opengis.net/spec/ogcapi-tiles-1/1.0/req/tmxs"
    "http://www.opengis.net/spec/tilematrixset/1.0/req/tilematrixset2d"
    "http://www.opengis.net/spec/tilematrixset/1.0/req/json-tilematrixset2d"
    "http://www.opengis.net/spec/tilematrixset/1.0/req/tilematrixsetlimits2d"
    "http://www.opengis.net/spec/tilematrixset/1.0/req/json-tilematrixsetlimits2d"
  ]
}
```

9.4. TileMatrixSets

9.4.1. Operation

Requirement 19	**/req/tiles/tmxs/tmxs-tilematrixsets-op**
A	The server SHALL support the HTTP GET operation at the path /tileMatrixSets.

9.4.2. Response

Requirement 20	**/req/tiles/tmxs/tmxs-tilematrixsets-success**
A	A successful execution of the operation SHALL be reported as a response with a HTTP status code 200.
B	The body of the response SHALL be a tileMatrixSets object listing the tilematrixsets supported by this server other than the eight ones defined in the Annex D of OGC 17-083r2 standard.
C	For each TileMatrixSet the response SHALL contain a TileMatrixSet id and a link to request the TileMatrixSet description.

Example 9. Schema for the TileMatrixSets resource

```
tileMatrixSets:
  type: array
  items:
    $ref: 'https://api.swaggerhub.com/domains/UAB-CREAF/ogc-api-
common/1.0.0#/components/schemas/id-link'
```

Example 10. Schema for id-link from OGC API Common used in TileMatrixSets resource and reproduced here for completeness.

id-link: type: object description: |- Reusable object that contains an id to a resource and links where the object is described or a representation retrieved. Typically it is useful for paths like `\resources` and `\resources{resourceId}`. `\resources` will respond an array of id-link listing the `resourceId` and the links to get it. \collections and \collections{collectionId} is an exception to this pattern. The fact that `links` is an array can be used to advertise the same object representation in different formats. required: - id - links properties: id: type: string title: type: string links: type: array minItems: 1 items: $ref: '#/components/schemas/link'

Example 11. Example for the TileMatrixSets resource

```
[
  {
    "id": "MyWebMercatorQuad",
    "title": "My Google Maps Compatible for the World",
    "links": [
      {
        "href": "https://data.example.org/tileMatrixSet/MyWebMercatorQuad",
        "rel": "tileMatrixSet",
        "type": "application/json"
      }
    ]
  }
]
```

9.5. TileMatrixSet

9.5.1. Operation

Requirement 21	**/req/tiles/tmxs/tmxs-tilematrixset-op**
A	The server SHALL support the HTTP GET operation at the path /tileMatrixSet/{tileMatrixSetId}.

A	The parameter tileMatrixSetId is each id property in the tileMatrixSets response.

9.5.2. Response

Requirement 22	**/req/tiles/tmxs/tmxs-tilematrixset-op**
A	A successful execution of the operation SHALL be reported as a response with a HTTP status code 200.
B	The body of the response SHALL follow the TileMatrixSet data model defined in the http://www.opengis.net/spec/tilematrixset/1.0/req/tilematrixset2d requirements class of the Clause 7 in the OGC 17-083r2 standard.
C	The body of the response SHAL be encoded in JSON following the requirements class http://www.opengis.net/spec/tilematrixset/1.0/req/json-tilematrixset2d of the Clause 9 in the OGC 17-083r2 standard.

Recommendation 3	**/rec/tiles/tmxs/tilematrixset-response**
A	The server may support that the tileMatrixSetId contains one of the eight TileMatrixSets defined in the Anned D of OGC 17-083r2 and return a successfull response with the description of it.

9.6. Tiles

This extension modifies the content of the tiles description returned by a successful \collection{collectionId}\tiles request.

9.6.1. Collection extra properties

Requirement 23	**/req/tiles/tmxs/stc-limits**

A	If the extent of the available tiles in the server is smaller than the extent of the TileMatrixSet, the object *tileMatrixSetLinks* in the *tiles* successful execution SHALL contain a property called *tileMatrixSetLimits* that is an array that specifies the limitations in the area available for this collection for each TileMatrix. *tileMatrixSetLink* object follows a data model defined in the clause 7.3 of OGC 17-083r2 that can be encoded in the following schema (expressed in OpenAPI Specification 3.0 fragment):

```
{
  "tileMatrixSetLinks": [
    {
      "type": "tileMatrixSetLink",
      "tileMatrixSet":
"http://www.opengis.net/def/tilematrixset/OGC/1.0/WebMercatorQuad",
      "tileMatrixSetLimits": [
        {
          "type": "tileMatrixSetLimits",
          "tileMatrix": "5",
          "minTileRow": 0,
          "maxTileRow": 1,
          "minTileCol": 3,
          "maxTileCol": 4
        }
      ]
    }
  ]
}
```



```
      type: object
      required:
        - tileMatrix
        - minTileRow
        - maxTileRow
        - minTileCol
        - maxTileCol
      properties:
        tileMatrix:
          type: string
          format: uri
          example: '5'
        minTileRow:
          type: number
          format: integer
          minimun: 0
          example: 0
        maxTileRow:
          type: number
          format: integer
          minimun: 0
          example: 1
        minTileCol:
          type: number
          format: integer
          minimun: 0
          example: 3
        maxTileCol:
          type: number
```

Chapter 10. Requirement Class "Tiles Info"

10.1. Overview

NOTE	TBD by the WMS.SWG. Here there are some initial thoughts to be elaborated. Please ignore this section.

Unresolved directive in clause_7_tile_info.adoc - include::requirements/tiles/requirements_class_info.adoc[]

This extension makes tiles a little more informative than just "nice pictures" by allowing clients to implement a click user event. By clicking on a tile, the user will receive some textual information describing what is shown in that pixel. For example, by clicking on a tile representing elevation the user will get the elevation value.

When full completed, the new OGC API architecture should be able to integrate several representations of the same resource. This way a digital elevation model could be accessible as a tile and also as a coverage. The coverage part should be able to provide elevation values to the client. When that day arrives, this info extension will no longer be needed as the coverage functionality will provide client with enough data to emulate this extension and some other extra interactions such us the capability to create vertical profiles.

10.2. Overview

Unresolved directive in clause_7_tile_info.adoc - include::requirements/tiles/requirements_class_info.adoc[]

10.3. Declaration of conformance classes

10.3.1. Response

The conformance page mainly consists in a list of links. OWS Commmon already requires some links.

Unresolved directive in clause_7_tile_info.adoc - include::requirements/tiles/limits/REQ_conformance-success.adoc[]

In the conformance page (commonly in JSON format) the links are following the link schema defined in the OGC API Common. The following is an example fragment of the response of a OGC API tiles conformance information page

Example 12. Conformance Information Page fragment format: tgtegel

```
{
  "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-tiles-1/1.0/req/core"
    "http://www.opengis.net/spec/ogcapi-tiles-1/1.0/req/info"
  ]
}
```

10.4. Collection

10.5. Collection

This standard includes dependencies on OWS Common collection. The response of the operation is extended with the necessary information to formulate a tile response for this collection.

10.5.1. Collection Links

Unresolved directive in clause_7_tile_info.adoc - include::recommendations/tiles/core/REC_sfc-tile-examples.adoc[]

Example 13. API collection response fragment

```
links:
[
    {
      "href":
"http://data.example.com/collections/buildings/tiles/WorldMercatorWGS84Quad/0/0/0"
,
      "rel": "tiles",
      "type": "image/png",
    },
    {
      "href":
"http://data.example.com/collections/buildings/tiles/WorldMercatorWGS84Quad/0/0/0/
info",
      "rel": "info",
      "type": "text/html",
    }
]
```

10.5.2. Collection extra properties

TBD. REMOVE FORMATS AS THEY ARE PROVIDED AS LINKS.

Example 14. Example of a tiles property in the collection response.

```
"tiles": {
  "formats": [
    [
      "image/jpeg",
      "image/png"
    ]
  ],
  "infoFormats": [
    [
      "application/geo+json",
      "text/html"
    ]
  ]
}
```

10.6. FeatureInfo

Map Tile APIs may support requests for information about the features present at a particular pixel location on a map tile. Requests for feature information will specify the tile along with a pixel location on that tile. The server will provide information on the features present at or near the location specified by the client request. The server may choose what information to provide about the nearby features.

10.6.1. FeatureInfo document

A FeatureInfo document is the resource representation of a FeatureInfo resource in resource oriented architectural style. The FeatureInfo document SHALL be in the format specified in the request when that format has been advertised in the **ServiceMetadata document** as available for that FeatureInfo resource.

TBD recommend GeoJSON instead.

For a better interoperability between servers and clients we strongly recommend GML Simple Features Profile [06-049r1] as a supported document format for FeatureInfo resources. That standard defines three levels of content in three profiles with different degrees of constraints to the GML flexibility. We strongly recommend support of the most constrained one (level 0) that results in a simpler GML document. In the context of that profile only simple XML types can be used as thematic properties and cardinality greater than one is not allowed. Servers and clients SHALL specify the MIME type "application/gml+xml; version=3.1" as an InfoFormat value and the GML application schema of the response SHOULD conform to GML Simple Features profile level 0 when that GML profile is used. In most cases, only thematic attributes of the features are intended to be

included in a FeatureInfo document but the Simple Feature profiles were evidently intended to include the geometric information of the features in the GML objects. However, it is possible to generate an application schema that does not include feature geometry and only describes non-geometric feature attribute types. This can be very useful to avoid unnecessarily requesting long sequences of position values in line or polygon layers.

Also, to allow easy presentation of the data, support for the HTML format (represented by an InfoFormat MIME type of "text/html") is also recommended.

10.6.2. GetResourceRepresentation

The ServiceMetadata document in the resource oriented architectural style may contain a list of Layer elements and each layer that is available to be retrieved in this architectural style and is queryable SHALL have one or more <ResourceURL> elements with the "resourceType" attribute set to "FeatureInfo" and a template attribute. In this RESTful approach the template attribute contains a URL template that can be converted to a URL by using a template processor and then get the expected FeatureInfo in the format specified by the attribute "format" by requesting the resource with a standard HTTP GET.

Unresolved directive in clause_7_tile_info.adoc - include::requirements/core/REQ_grr.adoc[]

Operation

The GetResourceRepresentation request operation is defined by the following requirement:

Unresolved directive in clause_7_tile_info.adoc - include::requirements/core/REQ_grr-op.adoc[]

An example GetResourceRepresentation request is

```
http://www.maps.bob/etopo2/default/WholeWorld_CRS_84/10m/1/3/86/132.xml
```

It corresponds to the following ResourceURL element: <ResourceURL format="application/gml+xml; version=3.1" resourceType="FeatureInfo" template="http://www.maps.bob/etopo2/default/{TileMatrixSet}/{TileMatrix}/{TileRow}/{TileCol}/{J}/{I}.xml"

Response

A successful GetResourceRepresentation response is defined by the following requirement:

Unresolved directive in clause_7_tile_info.adoc - include::requirements/core/REQ_grr-success.adoc[]

An XML document conforming GML Simple Features Profile [06-049r1] is the most typical representation but other representations and formats are also allowed.

Error situations

An unsuccessful GetResourceRepresentation reponse follows the general guidance for HTTP status codes provided in [http_status_codes] and the following requirement:

Unresolved directive in clause_7_tile_info.adoc - include::requirements/core/REQ_grr-error.adoc[]

Chapter 11. Requirement Class "Tiles Multitiles"

11.1. Overview

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

In this extension, we define a mechanism to request more than one tile from a single collection in a single request. The result can be a document listing the needed tiles to cover bounding box or a package with all tiles inside.

11.2. Declaration of conformance classes

11.2.1. Response

The conformance page mainly consists in a list of links. OWS Commmon already requires some links.

Requirement 24	**/req/tiles/multitiles/conformance-success**
A	The API conformance path SHALL advertise the capability of generating tiles from multiple collections adding the conformance class for this capability as a link to http://www.opengis.net/spec/ogcapi-tiles-1/1.0/req/multitiles.

In the conformance page (commonly in JSON format) the links are following the link schema defined in the OGC API Common. The following is an example fragment of the response of a OGC API tiles conformance information page

Example 15. 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-tiles-1/1.0/req/core"
    "http://www.opengis.net/spec/ogcapi-tiles-1/1.0/req/multitiles"
  ]
}
```

11.3. Tiles description

The response of this operation contains the necessary information to later formulate a tile or a multitile request for a collection.

11.3.1. Response

A successful response of a tiles request for a collection that can be retrieved as tiles will respond with a data structure with specific information necessary to get tiles or map-tiles representing the resource collection. In this extension, adds the URL template to a multitile.

Requirement 25	**/req/tiles/multitiles/mtc-multitiles-examples**
A	The content of the response of a successful execution SHALL include at least a link to a multitiles URI template (relation: `multitiles`).
B	These links SHALL provide a URL template with the fragment `/tiles` followed by the variables {tileMatrixSetId}. Once the variables are substituted by their respective valid values, a URL to a multitiles is obtained.
C	There SHALL be a link to a multitile URI template for each format that the server supports (the format is indicates in the `type` attribute of the link)

One common order used in URL templates for tiles is .../tiles/{tileMatrixSetId} this standard allows for other URL template dispositions.

Table 3. URI template variables for tiles and possible values

URL template variable	Meaning	Possible values	TileMatrixSetId

Example 16. API tiles response fragment

```
links:
[
    {
      "href":
"http://data.example.com/collections/buildings/tiles/{tileMatrixSetId}",
      "rel": "multitiles",
      "type": "image/png",
    }
]
```

11.4. Multiple tiles from one collection

This extension provides a mechanism to select and retrieve a set of tiles at once following a TileMatrixSet.

11.4.1. Operation

Requirement 26	**/req/tiles/multitiles/mtc-op**
A	Tiles SHALL be available as HTTP GET requests to a URI that will be composed by two parts: a initial part is the URI of a resource that can be represented as tiles and the final part follows the pattern /tiles/{tileMatrixSetId}
B	Only the resources or collections that advertise one of more links with type=tiles SHALL be requested as multiple tiles.

Typical resources that can be retrieved as tiles are: features (/collections/{collectionId}), coverages (/collections/{collectionId}/coverages/{coverageId} or /coverages/{coverageId}) or maps (/collections/{collectionId}/map/styleId}).

11.4.2. Parameter tileMatrixSetId

Requirement 27	**/req/tiles/multitiles/mtc-tilematrixsetid-definition**

A	The operation SHALL support a parameter `tileMatrixSetId` with the following characteristics (using an OpenAPI Specification 3.0 fragment):
	
	`name: tileMatrixSetId`
	`in: path`
	`description: Identifier of a specific tiling scheme. It can be one of the specified in Annex D.1 of the OGC 17-083r2 standard or one defined in this service.`
	`required: true`
	`schema:`
	`  type: string`
	`example: WebMercatorQuad`

11.4.3. Parameter bbox

Requirement 28	**/req/tiles/multitiles/mtc-bbox-definition**

A	The operation SHALL support an optional a parameter `bbox` to filter the area where tiles will be retrieved with the following characteristics (using an OpenAPI Specification 3.0 fragment):
	
	`name: bbox`
	`in: query`
	`description:`
	`'Only elements that have a geometry that intersects the bounding box are selected.`
	`The bounding box is provided as four or six numbers, depending on whether the coordinate reference system includes a vertical axis (elevation or depth):`
	
	`* Lower left corner, coordinate axis 1`
	`* Lower left corner, coordinate axis 2`
	`* Lower left corner, coordinate axis 3 (optional)`
	`* Upper right corner, coordinate axis 1`
	`* Upper right corner, coordinate axis 2`
	`* Upper right corner, coordinate axis 3 (optional)`
	
	`The coordinate reference system of the values is WGS 84 longitude/latitude`
	`(http://www.opengis.net/def/crs/OGC/1.3/CRS84) unless a different coordinate reference system is specified by another parameter in the API (e.g `bbox-crs`).'`
	`required: false`
	`schema:`
	`type: array`
	`minItems: 4`
	`maxItems: 6`
	`items:`
	`type: number`
	`format: double`
	`style: form`
	`explode: false`

B	A TileMatrixSet definition points to a CRS. The coordinates of the bbox SHALL be in the CRS of specified in the definition of the the TileMatrixSet identified by the tileMatrixSetId
C	If the parameter is not specified, the server SHALL assume the whole extent of the tiles are requested.

This definition in inherited from OGC API Common.

11.4.4. Parameter scaleDenominator

Requirement 29	/req/tiles/multitiles/mtc-scaledenominator-definition

A	The operation SHALL support an optional a parameter `scaleDenominator` to filter the scales where tiles will be retrieved with the following characteristics (using an OpenAPI Specification 3.0 fragment):
	
	`name: scaleDenominator`
	`in: query`
	`description:`
	`A range of scale denominators (that can be used to generate a list of tileMatrix names).`
	`required: false`
	`style: form`
	`explode: false`
	`schema:`
	`type: array`
	`minItems: 2`
	`maxItems: 2`
	`items:`
	`type: number`
	`format: double`

B	If the parameter is not specified, the server SHALL assume all TileMatrices (scales) SHALL be returned.

Recommendation 4	/rec/tiles/multitiles/mtc-scaledenominator-definition
A	To prevent mistakes identifying the scale denominator due to precision issues caused by lack of significant digits, the client should apply a tolerance to intervals. If the client wants to specify a single scale denominator, it will use an small interval with enough tolerance.

11.4.5. Parameter multiTileType

Requirement 30	/req/tiles/multitiles/mtc-multitiletype-definition

A	The operation SHALL support an optional a parameter `multitileType` that determines the type of the response and with the following characteristics (using an OpenAPI Specification 3.0 fragment):
	
	`name: multiTileType`
	`in: query`
	`description:`
	`'When successful, the service will respond to a query in one of two ways. It can provide a file with links to each tile or or it will provide the tiles in a package. The package can still contain the description of each tile`
	
	`The allowed values for this parameter are \`url\`, \`tiles\` and \`full\`.'`
	`style: form`
	`schema:`
	`type: string`
	`default: tiles`
	`enum:`
	`- url`
	`- tiles`
	`- full`
	`example: full`

B	If the value of the `multiTileType` parameter is set to `url` the server SHALL return a list of the selected tiles in a format following the `tileSet` schema. Each tile description in the list will contain a URL to download the tile later.
C	If the value of the `multiTileType` parameter is set to `tiles` or if the parameter is not specified in the request, the server SHALL return a package (e.g. a ZIP file) that will include tiles as separated parts in the package.
D	If the value of the `multiTileType` parameter is set to `full` the server SHALL return the tiles and a list of the selected tiles (in a format following the `tileSet` schema) as part of a package.

Permission 2	**/per/tiles/multitiles/mtc-multitiletype-definition**
A	Server MAY only implement a subset of the enumerated values (url, tiles, full) for the parameter `multitileType` and in this case it will only enumerate this subset in its schema.

11.4.6. Formats

In the cases of the multitile response, there are two formats involved. The multitile itself can be returned as a package (e.g. a zip file) that contains the tiles inside. The individual tiles also has its format. The format of the multitile if governed by the format procedure specified in the OGC API common. When the server supports multiple encoding for the individual tiles and the client has a preference for the tiles format, there is a need for communicating this preference to the server. This document does not mandate any particular approach how this is supported but provides the following recommendation.

Recommendation 5	**/rec/tiles/multitiles/mtc-f-tile-definition**
A	When the web interaction allows for HTTP format negotiation `Accept:` header is preferable to specify the required formats. In the case of multitype a composed format is recommended following the pattern application/vnd.ogc.multipart;container={multitile-media-type};tiles={tile-media-type}} (example: application/vnd.ogc.multipart;container=application/x-zip-compressed;tiles=image/png)
A	When the web interaction does not allow for controlling the HTTP format negotiation (e.g. URL in a HTML link), the operation MAY support an optional a parameter `f-tile` to specify the tile media type that the client prefers and a parameter `f` for the media type of the multitile response.
B	The content of these parameters should be specified by the server instance as an enumeration of supported media types in the API description.

11.4.7. Response

A successful response of a set of tiles will be consistent with the media type of resource requested.

Requirement 31	**/req/tiles/multitiles/mtc-success**
A	A successful execution of the operation SHALL be reported as a response with a HTTP status code `200`.
B	The content of that response SHALL be consistent with the format requested and be inside or intersect with the spatial extent of the geographical area represented by the 'bbox' and `scaleDenominator`.

C	If list of the tiles has been requested, the content of that response SHALL contain a tileSet document be based upon the following OpenAPI 3.0 schema:

D	When a package is being returned and the package format supports expressing file paths of its parts (such as the ZIP file), each tile in the package SHALL have a path following the template: `{tileMatrixSetId}/{tileMatrix}/{tileRow}/{tileCol}.{file-extension}`. {file-extension} is the file extension that corresponds to the media type (e.g "jpg" for image/jpeg).

Mainly this extension suggests 3 possible alternatives for a multitile response being the last one (`full`) the combination of the first two (`url` and `package`).

List Response

This format assumes that the client has viewport to represent an geographic area defined by the bounding box and the scale (that defines the pixel size of the viewport). This area should be populated with tiles. The server is expected to enumerate the tiles needed to populate the viewport and optionally to provide information on how to position the tiles in the viewport.

In the following example, we assume that the bounding box and scale provided implies a viewport of 336x446 pixels. The viewport it covered by 4 tiles. The client has requested a `url` type of multitile and negociated a JSON format. The URL of each tile is provided, accompanied with information on the position of the top left corner of each one in the viewport.

Example 17. Example of a tileSet document

```
          type: array
          items:
            $ref: '#/components/schemas/tileSetEntry'
    tileSetEntry:
      description:
        This is an entry on a multiple tiles request.
      type: object
      required:
      - tileURL
      - tileMatrix
      - tileRow
      - tileCol
      properties:
        tileURL:
          type: string
          format: uri
        tileMatrix:
          type: string
```

```
{
  "tileSet": [
    {
      "tileURL":
"http://data.example.com/collections/buildings/tiles/WebMercatorQuad/2/0/0.png",
      "tileMatrix": 0,
      "tileRow": 0,
      "tileCol": 0,
      "width": 256,
      "height": 256,
      "top": -10,
      "left": -20
    },
    {
      "tileURL":
"http://data.example.com/collections/buildings/tiles/WebMercatorQuad/2/0/1.png",
      "tileMatrix": 0,
      "tileRow": 0,
      "tileCol": 1,
      "width": 100,
      "height": 256,
      "top": -10,
      "left": 236
    },
    {
      "tileURL":
"http://data.example.com/collections/buildings/tiles/WebMercatorQuad/2/1/0.png",
      "tileMatrix": 0,
      "tileRow": 1,
      "tileCol": 0,
      "width": 256,
      "height": 200,
      "top": 246,
      "left": -20
    },
    {
      "tileURL":
"http://data.example.com/collections/buildings/tiles/WebMercatorQuad/2/1/1.png",
      "tileMatrix": 0,
      "tileRow": 1,
      "tileCol": 1,
      "width": 100,
      "height": 200,
      "top": 246,
      "left": 236
    }
  ]
}
```

Package Response

This format assumes that the client is interested in the tiles that cover a geographic area defined by the bounding box and the scale (or scales). The client knows what to do with the tiles and it is able to identify the tiles by their path using the URI template of the server as a pattern to extract the TileMatrix, TileRow and TileCol of each one.

Assuming that the client has requested a scale that fits with TileMatrix "2" and a bounding box that requires 2x2 tiles and that he client has requested a `package` type of multitile and negociated a ZIP format, a ZIP file is produced and sent by the server with the following files and paths:

Table 4. Content of a package containing 4 tiles

File	Path	TileMatrix	TileRow	TileCol
0.png	WebMercatorQuad/2/0	2	0	0
1.png	WebMercatorQuad/2/0	2	0	1
0.png	WebMercatorQuad/2/1	2	1	0
1.png	WebMercatorQuad/2/1	2	1	1

11.4.8. Error situations

A general summary of the HTTP status codes can be found in OWS Common.

If the parameter value `tileMatrixSetId` is not available by the server for this resource or the parameters values `bbox` or `scaleDenominator` are out-of-range, the status code of the response will be 404.

Chapter 12. Requirement Class "Tiles Collections Multitiles"

12.1. Overview

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

In this extension, we define a mechanism to request more than one tile from more than one collection in a single request. The result can be a document listing the needed tiles to cover bounding box or a package with all tiles inside. This section shares most of the content with the previous one and intents to provide similar mechanism. The main difference is the capability to request tiles that include elements of multiple collections provided by the parameter 'collections'.

12.2. API landing page

The landing page provides links to start exploring the resources offered by the API. It mainly consists in a list of links. The core of this standard does not add anything to the links required by OWS Common. The collections extension requires new link for the description of the tiles from more than one collection on top of the common ones that is inherited and needed by this extension.

12.3. Declaration of conformance classes

12.3.1. Response

The conformance page mainly consists in a list of links. OWS Commmon already requires some links.

Requirement 32	/req/tiles/cols-multitiles/conformance-success
A	The API conformance path SHALL advertise the capability of generating tiles from multiple collections adding the conformance class for this capability as a link to http://www.opengis.net/spec/ogcapi-tiles-1/1.0/req/cols-multitiles.

In the conformance page (commonly in JSON format) the links are following the link schema defined in the OGC API Common. The following is an example fragment of the response of a OGC API tiles conformance information page

Example 18. 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-tiles-1/1.0/req/core"
    "http://www.opengis.net/spec/ogcapi-tiles-1/1.0/req/collections"
    "http://www.opengis.net/spec/ogcapi-tiles-1/1.0/req/cols-multitiles"
  ]
}
```

12.4. Tiles description

The response of this operation contains the necessary information to later formulate a tile request from more than one collection as described in the `collections` extension. This extension add an extra link for the multitiles

12.4.1. Response

A successful response of a tiles request for more than on collection will respond with a data structure with specific information necessary to get tiles representing the resource collection. In this extension, the response informs about the URL template to retrieve multitiles.

Requirement 33	**/req/tiles/cols-multitiles/mtcs-multitiles-examples**
A	The content of the response of a successful execution SHALL include at least a link to a multitiles from multiple collections URI template (relation: `multitiles`).
B	These links SHALL provide a URL template with the fragment `/tiles` followed by the variables {tileMatrixSetId}. Once the variables are substituted by their respective valid values, a URL to a multitiles is obtained.
C	There SHALL be a link to a multitile URI template for each format that the server supports (the format is indicates in the `type` attribute of the link)

One common order used in URL templates for tiles is .../tiles/{tileMatrixSetId}, but this standard allows for other URL template dispositions.

Table 5. URI template variables for tiles and possible values

URL template variable	Meaning	Possible values	TileMatrixSetId

Example 19. API tiles response fragment

```
links:
[
    {
      "href": "http://data.example.com/tiles/{tileMatrixSetId}",
      "rel": "tiles",
      "type": "image/png",
    }
]
```

12.5. Multiple tiles from more than one collection

This extension provides a mechanism to select and retrieve a set of tiles at once from a TileMatrixSet.

12.5.1. Operation

Requirement 34	**/req/tiles/multitiles/mtcs-op**
A	Tiles SHALL be available as HTTP GET requests to a URI that will be composed by two parts: a initial part is the URI of a resource that can be represented as tiles and the final part follows the pattern /tiles/{tileMatrixSetId}
B	Only the resources or collections that advertise one of more links with type=tiles SHALL be requested as multiple tiles.

12.5.2. Parameter tileMatrixSetId

Requirement 35	**/req/tiles/multitiles/mtcs-tilematrixsetid-definition**

A	The operation SHALL support a parameter `tileMatrixSetId` with the following characteristics (using an OpenAPI Specification 3.0 fragment):
	
	`name: tileMatrixSetId`
	`in: path`
	`description: Identifier of a specific tiling scheme. It can be one of the specified in Annex D.1 of the OGC 17-083r2 standard or one defined in this service.`
	`required: true`
	`schema:`
	`  type: string`
	`example: WebMercatorQuad`

12.5.3. Parameter bbox

Requirement 36	**/req/tiles/multitiles/mtcs-bbox-definition**

A	The operation SHALL support an optional a parameter `bbox` to filter the area where tiles will be retrieved with the following characteristics (using an OpenAPI Specification 3.0 fragment):
	
	`name: bbox`
	`in: query`
	`description:`
	`'Only elements that have a geometry that intersects the bounding box are selected.`
	`The bounding box is provided as four or six numbers, depending on whether the coordinate reference system includes a vertical axis (elevation or depth):`
	
	`* Lower left corner, coordinate axis 1`
	`* Lower left corner, coordinate axis 2`
	`* Lower left corner, coordinate axis 3 (optional)`
	`* Upper right corner, coordinate axis 1`
	`* Upper right corner, coordinate axis 2`
	`* Upper right corner, coordinate axis 3 (optional)`
	
	`The coordinate reference system of the values is WGS 84 longitude/latitude (http://www.opengis.net/def/crs/OGC/1.3/CRS84) unless a different coordinate reference system is specified by another parameter in the API (e.g `bbox-crs`).'`
	`required: false`
	`schema:`
	`type: array`
	`minItems: 4`
	`maxItems: 6`
	`items:`
	`type: number`
	`format: double`
	`style: form`
	`explode: false`

B	A TileMatrixSet definition points to a CRS. The coordinates of the bbox SHALL be in the CRS of specified in the definition of the the TileMatrixSet identified by the tileMatrixSetId
C	If the parameter is not specified, the server SHALL assume the whole extent of the tiles are requested.

This definition in inherited from OGC API Common.

12.5.4. Parameter scaleDenominator

Requirement 37	/req/tiles/multitiles/mtcs-scaledenominator-definition

A	The operation SHALL support an optional a parameter `scaleDenominator` to filter the scales where tiles will be retrieved with the following characteristics (using an OpenAPI Specification 3.0 fragment):
	
	`name: scaleDenominator`
	`in: query`
	`description:`
	`'A range of scale denominators (that can be used to generate a list of tileMatrix names).'`
	`required: false`
	`style: form`
	`explode: false`
	`schema:`
	`type: array`
	`minItems: 2`
	`maxItems: 2`
	`items:`
	`type: number`
	`format: double`

B	If the parameter is not specified, the server SHALL assume all TileMatrices (scales) SHALL be returned.

Recommendation 6	/rec/tiles/multitiles/mtcs-scaledenominator-definition
A	To prevent mistakes identifying the scale denominator due to precision issues caused by lack of significant digits, the client should apply a tolerance to intervals. If the client wants to specify a single scale denominator, it will use an small interval with enough tolerance.

12.5.5. Parameter multiTileType

Requirement 38	/req/tiles/multitiles/mtcs-multitiletype-definition

A	The operation SHALL support an optional a parameter `multitileType` that determines the type of the response and with the following characteristics (using an OpenAPI Specification 3.0 fragment):
	
	`name: multiTileType`
	`in: query`
	`description:`
	`'When successful, the service will respond to a query in one of two ways. It can provide a file with links to each tile or or it will provide the tiles in a package. The package can still contain the description of each tile`
	
	`The allowed values for this parameter are 'url', 'tiles' and 'full'.'`
	`style: form`
	`schema:`
	`type: string`
	`default: tiles`
	`enum:`
	`- url`
	`- tiles`
	`- full`
	`example: full`

B	If the value of the `multiTileType` parameter is set to `url` the server SHALL return a list of the selected tiles in a format following the `tileSet` schema. Each tile description in the list will contain a URL to download the tile later.
C	If the value of the `multiTileType` parameter is set to `tiles` or if the parameter is not specified in the request, the server SHALL return a package (e.g. a ZIP file) that will include tiles as separated parts in the package.
D	If the value of the `multiTileType` parameter is set to `full` the server SHALL return the tiles and a list of the selected tiles (in a format following the `tileSet` schema) as part of a package.

Permission 3	**/per/tiles/multitiles/mtcs-multitiletype-definition**
A	Server MAY only implement a subset of the enumerated values (url, tiles, full) for the parameter `multitileType` and in this case it will only enumerate this subset in its schema.

12.5.6. Parameter Collections

Requirement 39	/req/tiles/collections/mtcs-collections-definition

A	The operation SHALL support an optional parameter `collections` with the following characteristics (using an 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 type=tiles in the /collections/{collectionId} SHALL be included.
D	Only the collections that support the same TileMatrixSetId parameter value SHALL be included
C	If `collections` is missing, all collections supporting the TileMatrixSetId parameter value will be considered.

12.5.7. Formats

In the cases of the multitile response, there are two formats involved. The multitile itself can be returned as a package (e.g. a zip file) that contains the tiles inside. The individual tiles also has its format. The format of the multitile if governed by the format procedure specified in the OGC API common. When the server supports multiple encoding for the individual tiles and the client has a preference for the tiles format, there is a need for communicating this preference to the server. This document does not mandate any particular approach how this is supported but provides the following recommendation.

Recommendation 7	/rec/tiles/multitiles/mtcs-f-tile-definition

A	The operation MAY support an optional a parameter `f-tile` to specify the tile format that the client prefers as parts of the multitile response.
B	The content of this parameter should be specified by the server instance as an enumeration of supported media types.

12.5.8. Response

A successful response of a set of tiles will be consistent with the media type of resource requested. This standard does not impose any media type but suggest the use of a package format.

Requirement 40	**/req/tiles/cols-multitiles/mtcs-success**
A	A successful execution of the operation SHALL be reported as a response with a HTTP status code `200`.
B	The content of that response SHALL be consistent with the format requested and be inside or intersect with the spatial extent of the geographical area represented by the 'bbox' and `scaleDenominator`.

C	If list of the tiles has been requested, the content of that response SHALL contain a tileSet document be based upon the following OpenAPI 3.0 schema:

D	When a package is being returned and the package format supports expressing file paths of its parts (such as the ZIP file), each tile in the package SHALL have a path following the template: `{tileMatrixSetId}/{tileMatrix}/{tileRow}/{tileCol}.{file-extension}`. {file-extension} is the file extension that corresponds to the media type (e.g "jpg" for image/jpeg).

12.5.9. Error situations

A general summary of the HTTP status codes can be found in OWS Common.

If the parameter value `tileMatrixSetId` is not available by the server for this resource or the parameters values `bbox` or `scaleDenominator` are out-of-range, the status code of the response will be 404.

```
        type: array
        items:
          $ref: '#/components/schemas/tileSetEntry'
    tileSetEntry:
      description:
        This is an entry on a multiple tiles request.
      type: object
      required:
        - tileURL
        - tileMatrix
        - tileRow
        - tileCol
      properties:
        tileURL:
          type: string
          format: uri
        tileMatrix:
          type: string
        tileRow:
          type: number
        tileCol:
          type: number
        width:
          type: number
          description:
            The width of the tile in rendering device
pixels. If it exceeds the visual display area be should
cut when displayed
        height:
          type: number
          description:
            The height of the tile in rendering device
pixels. If it exceeds the visual display area be should
cut when displayed
        top:
          type: number
          description:
            Vertical position from the top of the visual
display area in pixels. Negative value means that the
left side of the tile is outside the top-left corner of
the display and should be cut when displayed
        left:
          type: number
```

Chapter 13. Requirement Class "Map Core"

13.1. 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 as a pictorial representation of a dataset or the collections it has been divided in. To create a pictorial representation a style is added to the collections. Then, a map can be retrieved by specifying a set of parameters that will determine its resolution (width, height, boundingbox and CRS) or can be retrieved as tiles.

This clause defines the core part of the OGC API Maps that allows defining a map representation of a collection. To retrieve a fragment of the map, this clause need to be combined with a tiles extension or a map extension.

To keep the core simple, it will introduce a mechanism to select the default style but it will not introduce any mechanism to define or select other styles. The core only assumes that the service is capable of dealing with a default style.

13.2. General

Requirement 41	**/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 follows OGC API Common core and collections requirements (STILL TBD but mostly equivalent to the more general parts of WFS3.0 requirements but with the text generalized to other resource types). This standard provides extra additions to the OWS common requirements that are particular to maps.

13.3. API landing page

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

13.3.1. Response

No variations are required in the landing page. With a /collections successful response it is possible to retrieve the list of collectionId and links to the /collections/{collectionId}. With a /collections/{collectionId} successful response, it is possible to discover the links to retrieve some maps. Note that other resources can also be retrieved as collections (e.g. coverages)

13.4. 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.

13.4.1. Response

The conformance page mainly consists in a list of links. OWS Commmon already requires some links.

Requirement 42	**/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 (commonly in JSON format) the links are following the link schema defined in the OGC API Common. The following is an example fragment of the response of a OGC API maps conformance information page

Example 20. 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"
  ]
}
```

13.5. Collections

TBD: What will happen with collections of coverages?

This standard includes dependencies on OWS Common collections. Collections are mandatory in the core of this standard because collections are the object that will be included in a map.

Collections will enumerate the collectionId 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 43	**/req/tiles/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 (relation: `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.

13.6. Maps description resource

The maps core defines a `maps` resource that is associated to an operation contains the necessary information to later formulate a map request for a collection. Nevertheless, the core does not require an information mandatory since the map core alone does not specify how to retrieve a map. It is not possible to exemplify completely how this maps description looks like without considering other extensions.

13.6.1. Map description response

This core does not mandate a map description operation. Nevertheless, if it is defined by an extension, it introduces the need to have a specific properties.

In here, two properties inherit from WMS that are recommended here: cascade and opaque.

Recommendation 8	**/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)

```
the left side of the tile is outside the top-left corner
```

Recommendation 9	**/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)

13.7. Maps

Requirement 44	**/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: a initial part is the URI of a resource that can be represented as a map, a middle part following the pattern /map/{styleId} and a final part completing the retrieve parameters
B	Only the resources or collection that advertise one of more links following the pattern .../map/{styleId}... can be retrieved as maps.

13.7.1. Parameter styleId

Requirement 45	**/req/maps/core/mc-styleId-definition**

A	The operation SHALL support a parameter `styleId` with the characteristics defined (using an 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. It is up to the server to decide which is the default style. Other values might be defined in extensions

Chapter 14. Requirement Class "Map Styles"

14.1. 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 maps core introduces the possibility to create a map by assigning an style to a resource (e.g. a collectionId) but does not specify how to declare the styles supported by each collection. The core is only capable to request the default style and the client does not know anything about it. This extension discusses about the possibility to declare style names that can be used to request maps. It is foreseen that a OGC API Styles will allow to retrieve the full information about the style or to send styles to the server.

14.2. Declaration of conformance classes

14.2.1. Response

The conformance page mainly consists in a list of links. OWS Commmon already requires some links.

Requirement 46	/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 (commonly in JSON format) the links are following the link schema defined in the OGC API Common. The following is an example fragment of the response of a OGC API tiles conformance information page

Example 21. 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"
  ]
}
```

14.3. Collection

This standard includes dependencies on OWS Common collection. The response of the operation is extended with the necessary information to formulate a map response for this collection.

14.3.1. Collection Links

Requirement 10	**/req/maps/styles/smc-map-examples**
A	If the collection can be represented as maps, a 200-response SHALL include at least a link to an example of map URI for each style. This example can be a distribution of the map as tiles or as maps.
B	The link SHALL follow and illustrate the result of applying the ... /map/{styleId}... pattern

In practice, since the map core alone does not specify how to retrieve a map, it is not possible to exemplify completely how the link looks like without considering other extensions. If the server also conforms to an extension to distribute the map as maps, the example will look like this.

Example 22. API collection response fragment

```
links:
[
    {
      "href": "http://data.example.com/collections/buildings/map/brown",
      "rel": "maps",
      "type": "image/png",
    }
]
```

14.3.2. Collection extra properties

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

Requirement 47	**/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 *sytles* is an object that SHALL conform with the following data mode (using an 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 23. 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 a a value for {styleId}.

The optional links element are useful to connect to a OGC API styles that allows to retrieve the styles description.

Recommendation 11	**/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 *sytles* is an object that conforms with the following data mode (using an OpenAPI Specification 3.0 fragment):
	`default-style:`
	`  type: string`
	`  description:`

Example 24. API collection response fragment

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

14.4. Maps

This extension extends the map operation by adding two parameters (`trasnparent` and `bgcolor`) to the `styleId` defined in the core.

14.4.1. Parameter styleId

A part from the `default` style, this extention introduces the values for the styleId that were presented in the `collectionId` definition.

14.4.2. Parameter transparent

Requirement 48	**/req/maps/styles/mc-transparent-definition**

A	The operation SHALL support an optional a parameter `transparent` to force a transparent background with the characteristics defined (using an 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`.

14.4.3. Parameter bgcolor

Requirement 49	**/req/maps/styles/mc-bgcolor-definition**

A	The operation SHALL support an optional a parameter `bgcolor` to define a background color with the characteristics defined (using an 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 want to force an opaque color, a part 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 in the map.

Chapter 15. Requirement Class "Map Maps"

NOTE	TBD by the WMS.SWG. Here there are some initial thoughts to be elaborated

15.1. Overview

Unresolved directive in clause_8_map_maps.adoc - include::requirements/maps/requirements_class_maps.adoc[]

This extension describes how a map can be retrieved by specifying a set of parameters that will determine its resolution (width, height, boundingbox and CRS).

The map can use the default style or it can select one of the styles available if the right extension is also added to the core.

15.2. General

Requirement 50	**/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 follows OGC API Common core requirements (STILL TBD but mostly equivalent to the more general parts of WFS3.0 requirements but with the text generalized to other resource types). This standard provides extra additions to the OWS common requirements that are particular of tiles.

15.3. API landing page

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

15.3.1. Response

No variations are required in the landing page. With a /collections successful response it is possible to retrieve the list of collectionId and links to the /collections/{collectionId}. With a /collections/{collectionId} successful response, it is possible to discover the links to retrieve some maps. Note that other resources can also be retrieved as collections (e.g. coverages)

15.4. 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.

15.4.1. Response

The conformance page mainly consists in a list of links. OWS Commmon already requires some links.

Unresolved directive in clause_8_map_maps.adoc - include::requirements/maps/maps/REQ_conformance-success.adoc[]

In the conformance page (commonly in JSON format) the links are following the link schema defined in the OGC API Common. The following is an example fragment of the response of a OGC API Maps with the maps extension conformance information page.

Example 25. 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/maps"
  ]
}
```

15.5. Collections

TBD: What will happen with collections of coverages?

This standard includes dependencies on OWS Common collections. Collections are mandatory in the core of this standard because collections are the object that will be included in a tile.

Collections will enumerate the collectionId 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. to retrieve more information, you should use /collections/{collectionId}

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

15.6. Collection

This standard includes dependencies on OWS Common collection. The response of the operation is extended with a new link for the maps description.

15.6.1. Collection Links

Unresolved directive in clause_8_map_maps.adoc - include::requirements/maps/core/REQ_smc-maps-example.adoc[]

15.6.2. Collection extra parameters

A collection that can be retrieved as maps will add specific object in the collection information about that will contain specific information necessary to get tiles in the resource collection response. In this extension, the *maps* section informs about the formats the maps can be retrieved,...

Unresolved directive in clause_8_map_maps.adoc - include::requirements/maps/maps/REQ_sfc-formats.adoc[]

Example 26. Example of a tiles property in the collection response.

```
"maps": {
  "formats": [
    [
      "image/jpeg",
      "image/png"
    ]
  ]
}
```

15.7. Maps from one collection

This standard specifies how to get maps from a single collection.

15.7.1. Operation

Unresolved directive in clause_8_map_maps.adoc - include::requirements/maps/maps/REQ_mc-op.adoc[]

Typical resources that can be retrieved as maps are: features (/collections/{collectionId}), coverages (/collections/{collectionId}/coverage/{coverageId} or /coverage/{coverageId}).

15.7.2. Parameter crs

Unresolved directive in clause_8_map_maps.adoc - include::requirements/maps/maps/REQ_mc-crs-definition.adoc[]

15.7.3. Parameter bbox

Unresolved directive in clause_8_map_maps.adoc - include::requirements/maps/maps/REQ_mc-bbox-definition.adoc[]

15.7.4. Parameter width

Unresolved directive in clause_8_map_maps.adoc - include::requirements/maps/maps/REQ_mc-width-definition.adoc[]

15.7.5. Parameter height

Unresolved directive in clause_8_map_maps.adoc - include::requirements/maps/maps/REQ_mc-height-definition.adoc[]

15.7.6. Response

A successful response of a tile request will be consistent with the type of resource requested. For features can be geojson of Mapbox vector tiles; coverages it my be a geotiff and for maps it can be a jpeg or a png.

Unresolved directive in clause_8_map_maps.adoc - include::requirements/tiles/maps/REQ_tc-success.adoc[]

15.7.7. Error situations

A general summary of the HTTP status codes can be found in OWS Common.

If the parameter tileMatrixSetId is not available by the server for this resource or the parameters tileMatrix, tileRow, tileCol are out-of-range, the status code of the response will be 404.

Chapter 16. Requirement Class "Map from more than one collection"

16.1. 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 we have been discussion about maps that are produced form one and only one resource. This is achieved by concatenating the map path to a resource (e.g. a feature collection). This extension discusses about the possibility to combine more than one resource to create a map. This is achieved by using by adding the map path to the root of the service.

16.2. Declaration of conformance classes

16.2.1. Response

The conformance page mainly consists in a list of links. OWS Commmon already requires some links.

Requirement 51	**/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 (commonly in JSON format) the links are following the link schema defined in the OGC API Common. The following is an example fragment of the response of a OGC API tiles conformance information page

Example 27. 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"
  ]
}
```

16.3. Maps from more than one collection

16.3.1. Operation

Requirement 52	**/req/maps/collections/mcs-op**
A	The server SHALL support the HTTP GET operation at the path /maps

16.3.2. Parameter styles

Requirement 53	**/req/maps/collections/mcs-styles-definition**

A	The operation SHALL support an optional parameter `styles` with the characteristics defined (using an 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

16.3.3. Parameter Collections

Requirement 54	**/req/maps/collections/mcs-collections-definition**

A	The operation SHALL support an optional parameter `collections` with the following characteristics (using an 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 `crsId` 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.

16.3.4. Response

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

Chapter 17. Requirement Class "Checkpoint core"

17.1. Overview

Requirements Class	
http://www.opengis.net/spec/ogcapi-checkpoint-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

This extension is a building block a prior applicable to any data resource that add a mechanism to request for server changes in a resource collection since a previous checkpoint. It should be implemented as a GET but it will be a *partial* GET.

In order to implement it the service needs to keep an audit of every change in the resources and assign a checkpoint to it. At some point in time the client will request a resource collection and receive it with a checkpoint identifier. After a while, the client will send a request for updates and notify the server about the checkpoint the client received. The server will inform the client about the changes in the resources up to a current checkpoint and the client will update its copy of the resources.

While dealing with atomic resources, implementors can consider to use e-tags to detect changes on cached version of them as defined in https://tools.ietf.org/html/rfc7232. The e-tag mechanism permits the client to provide a fast "not modified" response in case the resource has not changed. In case the resource has been modified it is completely downloaded. In contrast, this requirement class adds the capability for partial update of a collection and the server will only retrieve the resource completely if all its parts has changed.

17.2. General

Requirement 55	**/req/checkpoint/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 follows OGC API Common core and collections requirements (STILL TBD but mostly equivalent to the more general parts of

WFS3.0 requirements but with the text generalized to other resource types). This standard provides extra additions to the OWS common requirements that are particular to maps.

17.3. API landing page

The checkpoint has been designed as query parameters that can potentially be added to any relevant URL. The landing page provides links to start exploring the resources offered by the API but does not expose checkpoints. It mainly consists in a list of links. OWS Common already requires some common links that are enough for this core.

17.3.1. Response

No variations are required in the landing page.

17.4. 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.

17.4.1. Response

The conformance page mainly consists in a list of links. OWS Commmon already requires some links.

Requirement 56	**/req/checkpoint/core/conformance-success**
A	The API conformance path SHALL advertise the checkpoint core conformance class as links to http://www.opengis.net/spec/ogcapi-checkpoint-1/1.0/req/core.

In the conformance page (commonly in JSON format) the links are following the link schema defined in the OGC API Common. The following is an example fragment of the response of a OGC API map tiles conformance information page with checkpoint support.

Example 28. 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",
    "http://www.opengis.net/spec/ogcapi-tiles-1/1.0/req/core",
    "http://www.opengis.net/spec/ogcapi-checkpoint-1/1.0/req/core",
  ]
}
```

17.5. Resource retrieval

17.5.1. Headers

A header in the response of resource retrieval will be added to a successful GET request with the checkpoint. This header will be added to a successful response of complete GET request of with a checkpoint parameter.

Requirement 57	**/req/checkpoint/core/resource-success**
A	The body of a successful response SHALL include a header x-checkpoint with the value of the current server checkpoint.

This is necessary for the client to be able to formulate a checkpoint update request later.

17.6. Checkpoint extra parameters

A resource retrieval GET request can have additional query parameters to specify a filter to get only the changes in a resource collections from the last client request. The client has saved checkpoint associated with the resources it has saved or cached and it will communicate it to the server. The server will be able to know what elements the client has and deliver only the updated elements (and the deleted elements).

For a low connectivity environment, a parameter will be able to select the priority level of the update changes and request only the most significant changes.

An additional parameter allows for selecting the amount of content and the format of the response.

17.6.1. Parameter Checkpoint

Requirement 58	**/req/checkpoint/core/cp-prioriy-definition**

A	The operation SHALL support an optional parameter `priority` with the following characteristics (using an OpenAPI Specification 3.0 fragment)
	
	`name: priority`
	`in: query`
	`required: false`
	`style: form`
	`explode: false`
	`schema:`
	`  type: string`
	`  default: all`
	`  enum:`
	`  - low`
	`  - medium`
	`  - high`
	`  - all`
	`example: high`

B	If the parameters is not specified in the request, the response SHALL contain all changes whatever their priority.

17.6.2. Parameter priority

Requirement 59	**/req/checkpoint/core/cp-checkpoint-definition**

A	The operation SHALL support an optional parameter `checkpoint` with the following characteristics (using an OpenAPI Specification 3.0 fragment)
	
	`name: checkPoint`
	`in: query`
	`required: false`
	`schema:`
	`  type: string`

B	If the parameters is not specified in the request, the response SHALL contain all changes since auditing began.

17.6.3. Parameter changeSetType

Requirement 60	**/req/checkpoint/core/cp-changesettype-definition**

A	The operation SHALL support an optional parameter `changeSetType` that determines the type of the response and with the following characteristics (using an OpenAPI Specification 3.0 fragment) and determines the type of the response
	
	`name: changeSetType`
	`in: query`
	`required: false`
	`style: form`
	`explode: false`
	`schema:`
	`  type: string`
	`  default: full`
	`  enum:`
	`  - summary`
	`  - full`
	`  - package`
	`example: full`

B	If the value of the `changeSetType` parameter is set to `full` or if the parameter is not specified in the request, the server SHALL return the complete changeSet file following the `changeSet` schema.
C	If the value of the `changeSetType` parameter is set to `summary` the server SHALL return a summary of the changes. The summary SHALL follow the `changeSetSummary` schema.
D	If the value of the `changeSetType` parameter is set to `package` the server SHALL return a package (e.g. a ZIP file) that will include a summary of the changes (The summary follows the `changeSetSummary` schema) and the changes themselves as separated parts in the package. Use this value for requesting changes in tiled resources.

17.7. ChangeSet response

17.7.1. ChangeSet response with changes

A success response of a resource request that has the checkpoint query filter will change to a format that contains a changeSet report and eventually the relevant changes.

Requirement 61	**/req/checkpoint/core/cp-success**

A	A successful execution of the operation that has found changes SHALL be reported as a response with a HTTP status code 200.
B	If the `checkPoint` property is present in the responds it SHALL contain the checkpoint value from which new changes are requested.
C	The response SHALL contain a property `summaryOfChangedItems` containing a count of changes corresponding to each specific priority label.
D	If the `extentOfChangedItems` property is present in the responds it SHALL contain a list with one or more of bounding boxes affected by changes produced and in which crs the coordinates of the bounding boxes are produced.
E	If the `numberOfReturnedItems` property is present in the responds it SHALL contain the number of changed items that are presented in the response.
F	If the `changedItems` property is present in the responds it SHALL contain a list with the new or modified resources (embedded or linked).
G	If the `deletedItems` property is present in the responds it SHALL contain a list of identifiers of deleted resources.

H	If a summary of changes has been requested, the content of that response SHALL contain a changeSetSummary report be based upon the following OpenAPI 3.0 schema:
	
	`title: changeSetSummary`
	`description: A document containing a summary of the updates since a specified checkpoint`
	`type: object`
	`required:`
	`- summaryOfChangedItems`
	`properties:`
	`summaryOfChangedItems:`
	`description: a per-priority list of change counts`
	`type: array`
	`items:`
	`description: a count of changes corresponding to a specific priority label`
	`type: object`
	`required:`
	`- priority`
	`- count`
	`properties:`
	`priority:`
	`description: the priority label`
	`type: string`
	`count:`
	`description: the count of changes tagged with the corresponding priority label`
	`type: integer`
	`example:`
	`- priority: high`
	`count: 2`
	`- priority: medium`
	`count: 4`
	`- priority: low`
	`count: 67`

I	If a changes report has been requested, the content of that response SHALL contain a changeSet report be based upon the following OpenAPI 3.0 schema:

```
  title: changeSet
  description: A document containing the updates since a
specified checkpoint
  allOf:
  - $ref: '#/components/schemas/changeSetSummary'
  - type: object
    properties:
      checkPoint:
        description: the checkpoint value from which new
changes are requested. This checkpoint is part of the
request to get this file.
        type: string
        example: "cp143756"
      extentOfChangedItems:
        description: extent of the new, changed or
deleted items
        type: object
        required: bbox
        properties:
          bbox:
            type: array
            items:
              $ref:
'https://api.swaggerhub.com/domains/UAB-CREAF/ogc-api-
common/1.0.0#/components/schemas/bbox'
          crs:
            $ref:
'https://api.swaggerhub.com/domains/UAB-CREAF/ogc-api-
common/1.0.0#/components/schemas/crs'
      numberOfReturnedItems:
        description: the number of changed items that
are presented in the response; this may be less than the
total number of changes
        type: integer
        example: 6
      changedItems:
        description: a list of new or modified
resources. Useful for features. Not recommended for
tiles
        type: array
        items:
          description: a representations of or reference
to the changed resource; e.g.
            a GeoJSON-encoded feature
          type: object
          required:
```

17.7.2. ChangeSet response with no changes

Requirement 62	/req/checkpoint/tiles/cp-not-modified
A	A successful execution of the operation that has NOT found changes SHALL be reported as a response with a HTTP status code 304.

```
            items:
          properties:
                oneOf:
                - type: object
                - $ref:
'https://api.swaggerhub.com/domains/UAB-CREAF/ogc-api-
common/1.0.0#/components/schemas/link'
        example:
          - priority: high
            items: []
          - priority: medium
            items: []
      deletedItems:
        description: a list of identifiers of deleted
resources. Useful for features. Not recommended for
tiles.
        type: array
        items:
          description: the identifier of a deleted
feature
          type: object
          required:
          - items
          properties:
            priority:
              description: a priority label
              type: string
            items:
              type: array
              items:
                type: string
        example:
          - priority: medium
            items:
            - "/collections/BUILDINGS/items/F4674"
            - "/collections/BUILDINGS/items/F37465819"
```

Chapter 18. Requirement Class "Checkpoint tiles"

18.1. Overview

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

This extension describes what is specific for tiles in checkpoints. It extents http://www.opengis.net/spec/ogcapi-tiles-1/1.0/req/multitiles, http://www.opengis.net/spec/ogcapi-tiles-1/1.0/req/cols-multitiles" and "http://www.opengis.net/spec/ogcapi-checkpoint-1/1.0/req/core".

18.2. API landing page

No changes in the The landing page are required.

18.3. Declaration of conformance classes

18.3.1. Response

The conformance page mainly consists in a list of links. OWS Commmon already requires some links.

Requirement 63	/req/checkpoint/tiles/conformance-success
A	The API conformance path SHALL advertise the capability of generating tiles from multiple collections adding the conformance class for this capability as a link to http://www.opengis.net/spec/ogcapi-checkpoint-1/1.0/req/tiles.

In the conformance page (commonly in JSON format) the links are following the link schema defined in the OGC API Common. The following is an example fragment of the response of a OGC API tiles conformance information page

Example 29. 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",
    "http://www.opengis.net/spec/ogcapi-tiles-1/1.0/req/core",
    "http://www.opengis.net/spec/ogcapi-tiles-1/1.0/req/tmxs",
    "http://www.opengis.net/spec/ogcapi-tiles-1/1.0/req/multitiles",
    "http://www.opengis.net/spec/ogcapi-checkpoint-1/1.0/req/core",
    "http://www.opengis.net/spec/ogcapi-checkpoint-1/1.0/req/tiles"
  ]
}
```

18.4. Checkpoint operation

This extension is applied on top of the requirements class "http://www.opengis.net/spec/ogcapi-tiles-1/1.0/req/multitiles" or "http://www.opengis.net/spec/ogcapi-tiles-1/1.0/req/cols-multitiles". The parameters defined in "http://www.opengis.net/spec/ogcapi-checkpoint-1/1.0/req/core" are added to the parameters defined for multitiles or cols-multitiles (multitiles of tiles of multiple collections). For requesting updates to tiles there are some restrictions in the parameters defined in the core that are described in this extenaions.

18.4.1. Parameter changeSetType

Recommendation 12	**/rec/checkpoint/core/cp-changesettiles**
A	The parameter `changeSetType` SHOULD use `summary` or `package`

Note that the checkpoint operation has a different output than the pure multitile operation. In practice this means that the parameter changeSetType overwrites the parameter multiTileType and in case both parameters are present, changeSetType takes precedence and multiTileType should be ignored.

18.5. ChangeSet response

The changeSet report is the one defined in the core but with the addition of a scale range.

Requirement 64	**/req/checkpoint/tiles/cp-success**

A	If the `scalesOfChangedItems` property is present in the request, the response SHALL contain an interval of scales affected by changes. If it is not included all scales are affected SHALL be include

B	When the `changeSetType` parameter forces a response using the `changeSet` schema, the content of that response SHALL use the `changeSetTiles` schema instead (that extents `changeSet` by adding a scale interval). The `changeSetTiles` OpenAPI 3.0 follows the schema:
	
	`allOf:`
	`- $ref: '#/components/schemas/changeSet'`
	`- type: object`
	`  properties:`
	`  scalesOfChangedItems:`
	`  type: object`
	`  required:`
	`  - minScaleDenominator:`
	`  - maxScaleDenominator:`
	`  properties:`
	`  minScaleDenominator:`
	`  type: number`
	`  format: double`
	`  maxScaleDenominator:`
	`  type: number`
	`  format: double`

C	When a package is being returned and the package format supports expressing file paths of its parts (such as the ZIP file), each tile in the package SHALL have a path following the template: `{tileMatrixSetId}/{tileMatrix}/{tileRow}/{tileCol}.{file-extension}`. {file-extension} is the file extension that corresponds to the media type (e.g "jpg" for image/jpeg).

Chapter 19. Requirements classes for encodings

NOTE	THIS SECTION REQUIRES A CAREFUL REVIEW. PLEASE IGNORE IT
NOTE	The following content is from the OAPI Common specification and should be modified for Map Tile APIs. Keep in mind that both Map Tiles and metadata are supplied by the API. So HTML and GeoJSON encoding may be retained for the metadata responses.

19.1. Overview

This clause specifies four pre-defined requirements classes for encodings to be used by an OGC API implementation. These encodings are commonly used encodings for spatial data on the web:

- HTML
- GeoJSON
- Geography Markup Language (GML), Simple Features Profile, Level 0
- Geography Markup Language (GML), Simple Features Profile, Level 2

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

The Core requirements class includes recommendations to support HTML and GeoJSON as encodings, where practical. Clause 6 (Overview) includes a discussion about recommended encodings.

19.2. Requirement Class "HTML"

Geographic information that is only accessible in formats like GeoJSON or GML has two issues:

- The data is not discoverable using the most common mechanism for discovering information, that is the search engines of the Web;
- The data can not be viewed directly in a browser - additional tools are required to view the data.

Therefore, sharing data on the Web should include publication in HTML. To be consistent with the Web, it should be done in a way that enables users and search engines to access all data.

This is discussed in detail in Best Practice 2: Make your spatial data indexable by search engines [SDWBP]. This standard therefore recommends supporting HTML as an encoding.

Unresolved directive in clause_10_encodings.adoc - include::requirements/requirements_class_html.adoc[]

Unresolved directive in clause_10_encodings.adoc - include::requirements/html/REQ_definition.adoc[]

Unresolved directive in clause_10_encodings.adoc - include::requirements/html/REQ_content.adoc[]

Unresolved directive in clause_10_encodings.adoc - include::requirements/html/REC_schema-org.adoc[]

19.3. Requirement Class "GeoJSON"

GeoJSON is a commonly used format that is simple to understand and well supported by tools and software libraries. Since most Web developers are comfortable with using a JSON-based format supporting GeoJSON is recommended, if the feature data can be represented in GeoJSON for the intended use.

Unresolved directive in clause_10_encodings.adoc - include::requirements/requirements_class_geojson.adoc[]

Unresolved directive in clause_10_encodings.adoc - include::requirements/geojson/REQ_definition.adoc[]

Unresolved directive in clause_10_encodings.adoc - include::requirements/geojson/REQ_content.adoc[]

Templates for the definition of the schemas for the GeoJSON responses in OpenAPI definitions are available at featureCollectionGeoJSON.yaml and featureGeoJSON.yaml. These are generic schemas that do not include any application schema information about specific feature types or their properties.

Example 30. A GeoJSON FeatureCollection Object response

In the example below, only the first and tenth feature is shown. Coordinates are not shown.

```
{
  "type" : "FeatureCollection",
  "links" : [ {
    "href" : "http://data.example.com/collections/buildings/items/?f=json",
    "rel" : "self",
    "type" : "application/geo+json",
    "title" : "this document"
  }, {
    "href" : "http://data.example.com/collections/buildings/items/?f=html",
    "rel" : "alternate",
    "type" : "text/html",
    "title" : "this document as HTML"
  }, {
    "href" :
"http://data.example.com/collections/buildings/items/?f=json&startIndex=10&limit=1
0",
    "rel" : "next",
    "type" : "application/geo+json",
    "title" : "next page"
  } ],
  "timeStamp" : "2018-04-03T14:52:23Z",
  "numberMatched" : 123,
  "numberReturned" : 10,
  "features" : [ {
    "type" : "Feature",
    "id" : "123",
    "geometry" : {
      "type" : "Polygon",
      "coordinates" : [ ... ]
    },
    "properties" : {
      "function" : "residential",
      "floors" : "2",
      "lastUpdate" : "2015-08-01T12:34:56Z"
    }
  }, { ...
  }, {
    "type" : "Feature",
    "id" : "132",
    "geometry" : {
      "type" : "Polygon",
      "coordinates" : [ ... ]
    },
    "properties" : {
      "function" : "public use",
      "floors" : "10",
      "lastUpdate" : "2013-12-03T10:15:37Z"
    }
  } ]
}
```

Example 31. A GeoJSON Feature Object response

In the example below, coordinates are not shown.


```
{
  "type" : "Feature",
  "links" : [ {
    "href" : "http://data.example.com/collections/buildings/items/123/?f=json",
    "rel" : "self",
    "type" : "application/geo+json",
    "title" : "this document"
  }, {
    "href" : "http://data.example.com/collections/buildings/items/123/?f=html",
    "rel" : "alternate",
    "type" : "text/html",
    "title" : "this document as HTML"
  }, {
    "href" : "http://data.example.com/collections/buildings/items",
    "rel" : "collection",
    "type" : "application/geo+json",
    "title" : "the collection document"
  } ],
  "id" : "123",
  "geometry" : {
    "type" : "Polygon",
    "coordinates" : [ ... ]
  },
  "properties" : {
    "function" : "residential",
    "floors" : "2",
    "lastUpdate" : "2015-08-01T12:34:56Z"
  }
}
```


19.4. Requirement Class "Geography Markup Language (GML), Simple Features Profile, Level 0"

In addition to HTML and GeoJSON, a significant volume of feature data is available XML-based formats, notably GML. Therefore, this standard specifies requirement classes for GML. The Simple Features Profile, Level 0, is the simplest profile of GML and is typically supported by tools. The GML Simple Features Profile is restricted to data with 2D geometries supported by most tools. In addition, the Level 0 profile is limited to features that can be stored in a tabular data structure.

Unresolved directive in clause_10_encodings.adoc - include::requirements/requirements_class_gmlsf0.adoc[]

Unresolved directive in clause_10_encodings.adoc -

include::requirements/gmlsf0/REQ_definition.adoc[]

Unresolved directive in clause_10_encodings.adoc - include::requirements/gmlsf0/REQ_content.adoc[]

Table 6. Media types and XML elements for each resource

Resource	Path	XML root element
Landing page	`/`	`wfs:LandingPage`
Conformance classes	`/conformance`	`wfs:ConformsTo`
Feature collections metadata	`/collections`	`wfs:Collections`
Feature collection metadata	`/collections/{collectionId}`	`wfs:Collections`, with just one entry for the collection `collectionId`
Feature collection	`/collections/{collectionId}/items`	`wfs:FeatureCollection`
Feature	`/collections/{collectionId}/items/{featureId}`	substitutable for `gml:AbstractFeature`

The namespace prefixes used above are:

- `wfs`: `http://www.opengis.net/wfs/3.0`
- `gml`: `http://www.opengis.net/gml/3.2`
- `atom`: `http://www.w3.org/2005/Atom`
- `xlink`: `http://www.w3.org/1999/xlink`

The API definition resource at path `/api` is not included in Table 6. This requirements class does not prescribe any API definition approach.

The mapping of the content from the responses specified in the Core requirements class to the XML is straightforward. All links are encoded using `atom:link` elements except in GML features where simple Xlinks are used.

Annex C has example responses in XML.

NOTE The `wfs:FeatureCollection` element deliberately goes beyond the permitted content specified in the GML Simple Features Profile, section 8.4.2. This is necessary to support the hypermedia controls and other relevant content for a Web Feature Service API.

19.5. Requirement Class "Geography Markup Language (GML), Simple Features Profile, Level 2"

The difference between this requirement class and the Level 0 requirements class is that non-spatial feature properties are not restricted to atomic values (strings, numbers, etc.).

Unresolved directive in clause_10_encodings.adoc -

include::requirements/requirements_class_gmlsf2.adoc[]

Unresolved directive in clause_10_encodings.adoc - include::requirements/gmlsf2/REQ_definition.adoc[]

Unresolved directive in clause_10_encodings.adoc - include::requirements/gmlsf2/REQ_content.adoc[]

Chapter 20. Requirements class "OpenAPI 3.0"

NOTE THIS SECTION REQUIERES A CAREFUL REVIEW. PLEASE IGNORE IT

NOTE The OAS requirements class should be limited to OpenAPI 3.0 requirements that specific to Map Tile resources. These requirements are an extension to the OpenAPI requirements in OAPI Common.

What follows is from OAPI common. It should be replaced with approapriate Map Tile centered requirements.

20.1. Basic requirements

APIs conforming to this requirements class document themselves by an OpenAPI Document.

Unresolved directive in clause_11_oas.adoc - include::requirements/requirements_class_oas30.adoc[]

Unresolved directive in clause_11_oas.adoc - include::requirements/oas30/REQ_oas-definition-1.adoc[]

CAUTION ISSUE 117
The OpenAPI media type has not been registered yet with IANA and will likely change. We need to update the media type after registration.

Unresolved directive in clause_11_oas.adoc - include::requirements/oas30/REQ_oas-definition-2.adoc[]

CAUTION Related to ISSUE 90
If we have a rigid path pattern there seems to be no need to add requirements for fixed operationId values. However, if the path pattern would be flexible, maybe we should require specific operationIds for selected resources?

Two example OpenAPI documents are included in Annex B.

Unresolved directive in clause_11_oas.adoc - include::requirements/oas30/REQ_oas-impl.adoc[]

20.2. Complete definition

Unresolved directive in clause_11_oas.adoc - include::requirements/oas30/REQ_completeness.adoc[]

Note that APIs that, for example, are access-controlled (see Security), support web cache validation, CORS or that use HTTP redirection will make use of additional HTTP status codes beyond regular codes such as `200` for successful GET requests and `400`, `404` or `500` for error situations. See [http_status_codes].

Clients have to be prepared to receive responses not documented in the OpenAPI definition. For example, additional errors may occur in the transport layer outside of the server.

20.3. Exceptions

Unresolved directive in clause_11_oas.adoc - include::requirements/oas30/REQ_exception-codes.adoc[]

Example 32. An exception response object definition

```
description: An error occurred.
content:
  application/json:
    schema:
      $ref:
https://raw.githubusercontent.com/opengeospatial/OAPI/openapi/schemas/exception.ya
ml
  text/html:
    schema:
      type: string
```

20.4. Security

Unresolved directive in clause_11_oas.adoc - include::requirements/oas30/REQ_security.adoc[]

The OpenAPI specification currently supports the following security schemes:

- HTTP authentication,
- an API key (either as a header or as a query parameter),
- OAuth2's common flows (implicit, password, application and access code) as defined in RFC6749, and
- OpenID Connect Discovery.

CAUTION	ISSUE 41 How does a client determine which security protocols/standards/etc. a server supports?

Unresolved directive in OAPI_MapsTiles.adoc - include::clause_11_media_types.adoc[]

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

Unresolved directive in OAPI_MapsTiles.adoc - include::annex-history.adoc[]

Unresolved directive in OAPI_MapsTiles.adoc - include::annex-bibliography.adoc[]