# OGC API-Tiles

**Open Geospatial Consortium**

Submission Date: <yyyy-mm-dd>

Approval Date: <yyyy-mm-dd>

Publication Date: 2019-03-06

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

Internal reference number of this OGC® document: 20-057

Version: 0.0.1

Category: OGC® Implementation Specification

Editor: Joan Masó

# OGC API Tiles

**Warning**

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

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

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

# Table of Contents

## i. Abstract

The OGC has started a focused effort to extend their service standards into the Resource Oriented Architecture world. As part of this effort, this standard defines an Application Programming Interface (API) for tiled data and map tiles. The API described in this standard builds on the Web Map Tile Service (WMTS) OGC standard. WMTS provides 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. Whereas WMTS focused on map tiles, the OGC API - Tiles standard has been designed to support any form of tiled data. For example, the API can support map tiles, vector tiles, tiled gridded coverages and other tiled data.

## ii. Keywords

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

ogcdoc, OGC document, tiling, WMTS, map tiles, vector tiles, tiled feature data

## iii. Preface

This document defines the OGC API - Tiles standard. An API conforming to this standard can serve tiles of spatially referenced data or maps 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

## iv. Submitting organizations

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

Organization name(s)

## v. Submitters

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

| **Name** | **Affiliation** |
|---|---|
| Joan Masó | UAB-CREAF |
| Chuck Heazel | Heazel Tech |
| Jeff Harrison | AGC |
| Jérôme Jacovella-St-Louis | Ecere Corporation |
| TBA | TBA |
| TBA | TBA |

# Chapter 1. Scope

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

## 1.1. Current scope:

- Discovery operations allow the API to be interrogated to determine its capabilities and retrieve information (metadata) about this distribution of tiled data and map tiles. This includes the API definition as well as metadata about the data collections provided through the API and the Tile Matrix Sets (also referred to as TileMatrixSets) supported by this service.
- Retrieval of tiles
- Query about a point in a map or a tile
- Retrieval of 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 applications and services claiming compliance to the OGC API - Tiles standard have to implement.

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

**Capture 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 19-072, OGC API 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. geospatial resource

A resource that consists in a set of geospatial data

## 4.2. geospatial representation

An resources that represents an aspect or data model (e.g. feature items, tiles, metadata, schemas,… ) of a more generic geospatial resource (e.g. a collection)

**NOTE** Do not confuse this with a resource representation. While resource representations share the same path and are selected by format negotiation, geospatial representations use different paths. Commonly a geospatial representations is a child path of a geospatial resource

## 4.3. tile

geometric shape with known properties that may or may not be the result of a tiling (tessellation) process. A tile consists of a single connected "piece" without "holes" or "lines" (topological disc).

**NOTE** For the purposes of this OGC ER, a tile is a small rectangular representation of geographic data, often part of a set of such elements, covering a tiling scheme and sharing similar information content and graphical styling. A tile can be uniquely defined in a tile matrix by one integer index in each dimension. Tiles are mainly used for fast transfer (particularly in the web) and easy display at the resolution of a rendering device. Tiles can be grid based pictorial representations, coverage subsets, or feature based representations (e.g., vector tiles).

## 4.4. tile matrix

a grid tiling scheme that defines how space is partitioned into a set of conterminous tiles at a fixed scale.

**NOTE** A tile matrix constitutes a tessellation of the space that resembles a matrix in a 2D space characterized by a matrix width (columns) and a matrix height (rows).

## 4.5. tile matrix set

a tiling scheme composed by collection of tile matrices defined at different scales covering approximately the same area and has a common coordinate reference system.

## 4.6. tile set

set of tiles - a collection of subsets of the space being partitioned.

**NOTE** For the purposes of this OGC ER, a tile is a series of actual tiles contain data and following a common tiling scheme.

## 4.7. tiling scheme

a scheme that defines how space is partitioned into individual tiled units. A tiling scheme defines the spatial reference system, the geometric properties of a tile, which space a uniquely identified tile occupies, and reversely, which unique identifier corresponds to a space satisfying the geometric properties to be a tile.

**NOTE** A tiling scheme is not restricted to a coordinate reference system or a tile matrix set and allows for other spatial reference systems such as DGGS and other organizations including irregular ones.

## 4.8. vector tile

a tile that contains vector information that has been simplified at the tile scale resolution and clipped by the tile boundaries.

## 4.9. Web API

API using an architectural style that is founded on the technologies of the Web [source: OGC API - Features - Part 1: Core]

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

# Chapter 5. Conventions

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

## 5.1. Identifiers

The normative provisions in this standard are denoted by the URI

http://www.opengis.net/spec/{standard}/{m.n}

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

# Chapter 6. Overview

## 6.1. Evolution from OGC Web Services

OGC Web Service (OWS) standards have historically implemented a Remote Procedure Call (RPC) architectural style using Extensible Markup Language (XML) for payloads. This was the state-of-the-art when some of the initial versions of OGC Web Services were originally designed in the late 1990s and early 2000s. This architectural style has now a competing RESTful API style that is proposed as an alternative to the RPC pattern. A RESTful API style is resource-oriented instead of service-oriented. This OGC API - Tiles draft specification specifies an API that follows this Web architecture and in particular the W3C/OGC best practices for sharing Spatial Data on the Web as well as the W3C best practices for sharing Data on the Web.

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

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

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

This approach intends to support two types of client developers:

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

As a result of following a RESTful approach, OGC API implementations are not backwards compatible with OWS implementations per se. However, a design goal is to define OGC APIs in a way that an OGC API interface can be mapped to an OWS implementation (where appropriate). OGC APIs are intended to be simpler and more modern, but still an evolution from the previous versions and their implementations making the transition easy (e.g. by initially implementing facades in front of the current OWS services).

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

## 6.2. Tiles and maps

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

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

This draft specification only describes the core capabilities for the Tiles API. Other extensions to the core will define how to add tile matrix set descriptions, multi-tiles and simple pixel queries. To produce map tiles, some modules of the OGC API - Tiles will be needed.

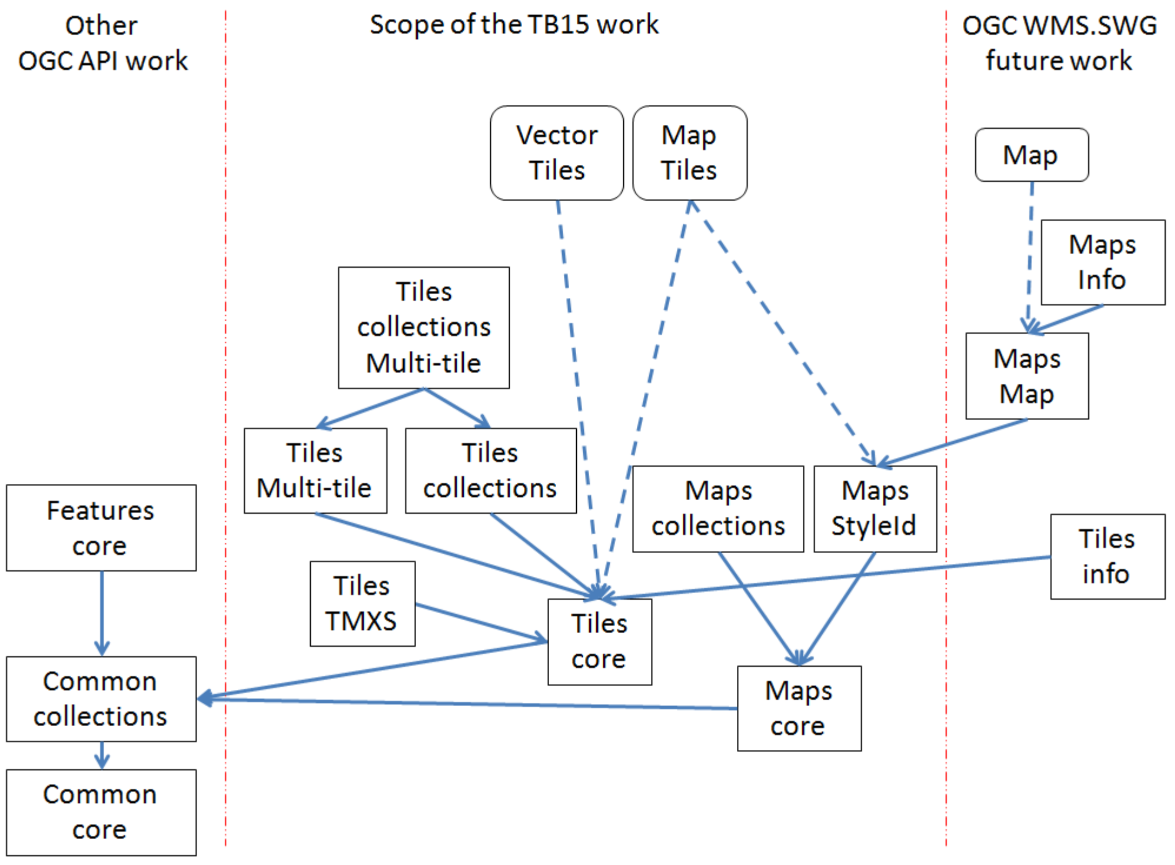

*Figure 1. Modular approach in the Maps and Tiles draft specification*

## 6.3. How to approach an OGC API

This specification cannot be implemented alone and should be considered a building block that could be applied to one or more existing resources in the API to get access spatial subsets of existing resources. This core defines two ways to get tiles specified in separate conformance classes. Developers are free to implement one of the approaches or both:

- tiles are applied as a transformation of a resource to obtain another resource as tiles.
- tiles are resources that are the result to combine one or more resources that characterized by their URLS.

# 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 |

This is a building block for the an OGC API that is able to provide geospatial resources. When applying the building block to a geospatial resource, it becomes available as tiles. The server can select which resources are available as tiles and will advertise which resources are available as tiles.

This building block does not specify how to get an API definition, the conformance class list or the geospatial resources lists. The standard assumes that the first two are defined by an API specification (e.g. OGC API Common) and the later by an OGC API for geospatial resource (e.g. OGC API - Features).

The core of the OGC API - Tiles core draft specification does not mandate the inclusion of an explicit definition of any TileMatrixSet. This draft specification assumes that clients and services know about the eight TileMatrixSets defined in OGC 17-083r2 annex D (or compatible future update of it) and there is no need to define new TileMatrixSets. An extension to the core provides the capability to include definitions of flexible TileMatrixSets that are explicitly defined.

This draft specification assumes that data is organized into one or more geospatial resources (e.g. the "collections" in OGC API - Features - Part 1: Core [http://www.opengis.net/doc/IS/ogcapi-features-1/1.0]). Geospatial resources are referenced using URIs.

This document does not specify any requirements for the type of geospatial resource that should be supported. Provided that the geospatial resources can be organized into tiles, they can be supported regardless of whether they are features, coverages, a resource that does not represent data per-se (e.g. an annotation) and so forth. The resource path replaces the concept of layer in WMS and WMTS. In this core tiles can be generated from only one geospatial resource (tiles that are generated as a combination of geospatial resources will be defined as an extension).

Accessing the geospatial resource content (other than as tiles) or its descriptions is out of the scope of this draft specification. If a description of the geospatial resource exists and it has a mechanism to add links to it, this specification will indicate the need to add a link to the tile representation description.

The tile representation description will include metadata about tiles as well as links to other resources including at least one with a template to get individual tiles.

## 7.2. General

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

This building block stays flexible and does not require implementation OGC API - Common, allowing for other API architectures outside the OGC API framework to adopt it. However, a server under the OGC APIs is expected to implement OGC API - Common. If so, in practice, this means that the landing page and the conformance page follow OGC API - Common core and collections requirement classes when used. Temporarily, it is also possible to combine this building block with OGC API - Features - Part 1: Core, version 1.0 [http://www.opengis.net/doc/IS/ogcapi-features-1/1.0] that is not tied to OGC API - Common.

## 7.3. Declaration of conformance classes

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

### 7.3.1. Response

The conformance page mainly consists of a list of links.

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

If the server declares also conformity to OGC API - Common or to OGC API - Features - Part 1: Core, version 1.0 [http://www.opengis.net/doc/IS/ogcapi-features-1/1.0], then it has to consider the OGC API - Common requirements for declaring conformance, i.e. the use of a the conformance page. In the JSON format the conformance page is an array of links following the link schema defined in the OGC API - Common or in OGC API - Features - Part 1: Core, version 1.0 [http://www.opengis.net/doc/IS/ogcapi-features-1/1.0]. Below is an example fragment of a conformance information page of an API conformant to OGC API - Common and OGC API - Tiles.

*Example 1. Conformance Information Page fragment*

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

## 7.4. Geospatial resources

This draft specification does not specify how geospatial resources are exposed in the API and if they can be retrieved as geospatial data (e.g. feature items). For example OGC API - Features - Part 1: Core, version 1.0 [http://www.opengis.net/doc/IS/ogcapi-features-1/1.0] includes the definition of collections and each collection is available in the /collections/{collectionId} path. OGC API - Common will provide a similar mechanism. Other paths in the API could also give access to geospatial resources.

**NOTE** The concept of geospatial resource path substitutes the concept of "layer" in WMTS 1.0 but it is intended to give a better integration between data visualization and data access.

## 7.5. Tiles description

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

### 7.5.1. Tiles description path

| **Requirement 2** | **/req/tiles/core/sct-op** |
|---|---|
| A | Every geospatial resource available as tiles SHALL support an path URL and a HTTP GET operation to retrieve the description of the tiles the API implementation can provide |
| B | The URI shall be composed by two parts: the initial part is the URI of the geospatial resource that can be represented as tiles and the final part follows the pattern `/tiles` |

This standard does not specify the need for any additional query parameter in the GET request.

### 7.5.2. Tiles description Link

| **Requirement 3** | **/req/tiles/core/tc-tile-desc-links** |
|---|---|
| A | If the API has a mechanism for their geospatial resources to expose links to geospatial representations (e.g. feature items), the API SHALL include a `link` with the `href` pointing to a the description of the tiles that presents a tile representation of this geospatial resource and with `rel: "tiles"`. |

For example, an implementation of the OGC API - Features - Part 1: Core [http://www.opengis.net/doc/IS/ogcapi-features-1/1.0] returns a list of links that include geospatial representations for each geospatial resource in the /collections/{collectionId} path. OGC API - Common is expected to provide a similar mechanism. In the JSON response, the array `links` is the place for adding a resource reference to the 'tiles' description.

*Example 2. Fragment of a collection with a links array with one item of the array pointing to a tiles description.*

```
    "id": "buildings",
    "title": "Buildings in the city of Bonn",
    "description": "This collection contains buildings",
    "attribution": "OpenStreetMap",
    "extent": {
      ...
    }
    "links": [
      ...
      {
        "href": "http://data.example.com/collections/buildings/tiles",
        "rel": "tiles",
        "type": "application/json",
      }
    ]
  }
```

### 7.5.3. Response

A successful GET response to a tiles description resource will respond with a data structure with the specific information necessary to build a complete GET request to the tiles representing the geospatial resource. In this core draft specification, the response is only required to inform about from which tile matrix sets the tiles can be retrieved and the URL template for retrieving the tiles.

| **Requirement 4** | **/req/tiles/core/sct-tmxslink** |
|---|---|

| | |
|---|---|
| A | If the tiles are available in a tile matrix set different from WebMercatorQuad, the content of the response to a successful execution to a tiles description 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 specification *tileMatrixSetLink* is only used for referencing the supported TileMatrixSets for the tiles, limiting it to the following schema (expressed as an OpenAPI Specification 3.0 fragment):<br>`tileMatrixSetLink-set:`<br>`  description: This list of tileMatrixSetLink objects, as defined in OGC 17-083r2 supported by this collectionId.`<br>`  type: array`<br>`  items:`<br>`    $ref: '#/components/schemas/tileMatrixSetLink-entry'`<br>`tileMatrixSetLink-entry:`<br>`  type: object`<br>`  required:`<br>`    - tileMatrixSet`<br>`  properties:`<br>`    tileMatrixSet:`<br>`      type: string`<br>`      example: 'WebMercatorQuad'`<br>`    tileMatrixSetURI:`<br>`      type: string`<br>`      format: uri`<br>`      example: 'http://www.opengis.net/def/tilematrixset/OGC/1.0/WebMercatorQuad'` |

| **Recommendation 2** | **/rec/tiles/core/sct-tmxslink** |
|---|---|
| A | This core requirements class does not provide any mechanism to define 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 [http://docs.opengeospatial.org/is/17-083r2/17-083r2.html#61]. |

| | |
|---|---|
| B | The server SHOULD do a effort to provide to the client a way to get full description of the TileMatrixSet. Even if the TileMatrixSet is not directly defined by the API, when a full definition of the TileMatrixSet is available as a resolvable URL, a resolvable URL SHOULD be used as the value of the tileMatrixSetURI. |
| C | This standard recommends the use of the TileMatrixSets defined in Annex D [http://docs.opengeospatial.org/is/17-083r2/17-083r2.html#61] of OGC 17-083r2. In the case of variable-width tiles, the standard recommends the use of the TileMatrixSets defined in Annex H [http://docs.opengeospatial.org/is/17-083r2/17-083r2.html#104] of OGC 17-083r2. |

TileMatrixSetLink is mandatory as part of the response of a ‘tiles’ description. Clients or servers are not required to support a specific default TileMatrixSet. The Example of a tiles metadata response shows how a TileMatrixSet can be referenced.

Resolvable URLs for the 8 URIs defined in the OGC 17-083r2 Annex D are available in the OGC schemas repository in XML, JSON and RDF formats. For example, JSON descriptions can be found here: http://schemas.opengis.net/tms/1.0/json/examples/

| **Requirement 5** | **/req/tiles/core/sct-tile-examples** |
|---|---|
| A | The content of the response to a successful execution of a tile description SHALL include at least a link to a tile URI template (rel: `item`). |
| 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 file format that the server supports (the format is indicated in the 'type' attribute of the link) |
| D | A property `templated` SHALL be part of the link properties to indicate that the link needs to be processed to substitute the templated variables with valid values before being used as a URL to a tile. |

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

**NOTE** The geospatial resource URL is expected to be the first part of the URL template (presented by the '…' in the previous note) but this standard does not mandate this.

*Example 3. Example of a tiles metadata response*

```
{
  "tileMatrixSetLinks": [
    {
      "tileMatrixSet": "WorldMercatorWGS84Quad",
      "tileMatrixSetURI":
"http://schemas.opengis.net/tms/1.0/json/examples/WorldMercatorWGS84Quad.json"
    }
  ],
  ...
  "links": [
    ...
    {
     "href":
"http://data.example.com/collections/buildings/tiles/{tileMatrixSetId}/{tileMatrix
}/{tileRow}/{tileCol}.png",
     "templated": true,
     "rel": "item",
     "type": "image/png",
   }
   ...
  ]
}
```

**NOTE** The use of "templated" is inspired by the JSON Hypertext Application Language (HAL), https://tools.ietf.org/html/draft-kelly-json-hal-08

The following table explains the meaning of the URI template variables.

*Table 1. URI template variables for tiles and valid values*

| URL template variable | Meaning | Possible values |
|---|---|---|
| TileMatrixSetId | tile matrix set identifier | One of the identifiers included in Annex D of OGC 17-083r2 or an identifier 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 |

| URL template variable | Meaning | Possible values |
|---|---|---|
| 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 |
| TileCol | column index of tile matrix | A non-negative integer between 0 and the MatrixWidth – 1. If there is a TileMatrixSetLimits the value is limited between MinTileCol and MaxTileCol |

## 7.6. A tile from a geospatial resource

A tile resource is a geospatial representation of a fragment of a geospatial resource that is spatially constrained at the boundaries of the selected tile in a tile matrix set.

### 7.6.1. Tile path and link

As described before a tile path is obtained by extracting a tile URL templated from one of the links with `rel`: `item` in tiles description document and substituting the templated variables of with valid values.

### 7.6.2. Operation

| **Requirement 6** | **/req/tiles/core/tc-op** |
|---|---|
| A | A tile that contains available data SHALL be available as a HTTP GET request to a URI that will be composed by two parts: The first part is the URI of a geospatial resource that can be represented as tiles and the second part follows the pattern /tiles/{tileMatrixSetId}/{tileMatrix}/{tileRow}/{tileCol} |

Typical geospatial resources that can be retrieved as tiles are: features (in OGC API - Features - Part 1: Core [http://www.opengis.net/doc/IS/ogcapi-features-1/1.0] represented by /collections/{collectionId}), or full maps (specified in the OGC API Maps).

### 7.6.3. Parameter tileMatrixSetId

| **Requirement 7** | **/req/tiles/core/tc-tilematrixsetid-definition** |
|---|---|

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

The core of the OGC API -Tiles standard provides a mechanism to select and retrieve a tile in a TileMatrixSet. If the service does not advertise any other TileMatrixSet (this core does not describe any mechanism to do that, but an extension will do it) the TileMatrixSet identifiers possible are limited to the ones specified in the Annex D.1 of the OGC 17-083r2 standard.

### 7.6.4. Parameter tileMatrix

| **Requirement 8** | **/req/tiles/core/tc-tilematrix-definition** |
|---|---|
| A | The operation SHALL support a parameter `tileMatrix` with the following characteristics (shown as OpenAPI Specification 3.0 fragment):<br><br>`name: tileMatrix`<br>`in: path`<br>`description: Identifier selecting one of the scales defined in the TileMatrixSet and representing the scaleDenominator the tile.`<br>`required: true`<br>`schema:`<br>`  type: string`<br>`example: '11'` |

### 7.6.5. Parameter tileRow

| **Requirement 9** | **/req/tiles/core/tc-tilerow-definition** |
|---|---|

|   |   |
|---|---|
| A | The operation SHALL support a parameter `tileRow` with the following characteristics (shown as OpenAPI Specification 3.0 fragment): |

```
  name: tileRow
  in: path
  description: Row index of the tile on the selected
TileMatrix. It cannot exceed the MatrixWidth-1 for the
selected TileMatrix
  required: true
  schema:
    type: integer
    minimum: 0
  example: '827'
```

## 7.6.6. Parameter tileCol

| **Requirement 10** | **/req/tiles/core/tc-tilecol-definition** |
|---|---|
| A | The operation SHALL support a parameter `tileCol` with the following characteristics (shown as OpenAPI Specification 3.0 fragment): |

```
  name: tileCol
  in: path
  description: Column index of the tile on the selected
TileMatrix. It cannot exceed the MatrixHeight-1 for the
selected TileMatrix.
  required: true
  schema:
    type: integer
    minimum: 0
  example: 1231
```

## 7.6.7. Response

A successful response to a tile GET operation will be consistent with the media type of resource requested. This draft specification does not impose any media type or file format. For example:

- For features the media type may be GeoJSON or Mapbox Vector Tiles;
- For coverages the response may be a GeoTIFF;
- For maps the response may be a JPEG or a PNG.

| **Requirement 11** | **/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. |

| **Permission 1** | **/per/tiles/root/tc-core-tile-encoding** |
|---|---|
| A | This draft specification does not impose any media type on the encoding of a response containing tiled feature data. For features the media type MAY be GeoJSON, Mapbox vector tiles or other format. |
| B | This draft specification does not impose any media type on the encoding of a response containing tiled coverage data. For coverages it MAY be a GeoTIFF or other format. |
| C | This draft specification does not impose any media type on the encoding of a map tile response. For maps it MAY be a JPEG, PNG or other format. |

Normally, the content partially outside the tile bounding box will be clipped at the extent of the bounding box. This can be done efficiently when tiles are in raster format (e.g. map tiles). However, tiles containing features in vector format may not clip features that are partially outside to ensure continuity of features for performance.

| **Recommendation 3** | **/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 are only expected for the lower values of scale denominators. |

To enable search engines to easily discover the content offered by an implementation of OGC API - Tiles, as well as to enable web browsers to easily display the content offered by the APIs, this specification allows for responses to operations to be encoded in HTML.

| **Permission 2** | **/per/tiles/root/tc-core-html** |
|---|---|

| | |
|---|---|
| A | Every `200`-response of an operation of the server MAY support the media type `text/html`. |

### 7.6.8. Error conditions

A general summary of the HTTP status codes can be found in OGC API - Features - Part 1: Core, version 1.0 [http://www.opengis.net/doc/IS/ogcapi-features-1/1.0] as well as in OGC API - 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, of the tile is not provided due to lack of data in the area, the status code of the response will be `404`.

# Chapter 8. Requirement Class "Tiles root"

## 8.1. Overview

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

In previous clauses tiles are produced from one, and only one geospatial resource available in this API. This scenario is achieved by concatenating the tile path to the geospatial resource.

This OGC API requirements class is an extension of the core requirements class that defines how to create tiles that combine one or more geospatial resources in any way that is decided in the client side. This is achieved by having the tile path available at the root of the service.

It has been argued that this approach is too flexible. In an API that has several geospatial resources, the number of potential combinations of geospatial resources may be too big to be efficiently handle. If the implementers see a potential performance issue, they may not choose to declare conformity to this requirements class.

## 8.2. General

This building block stays flexible and does not require implementation of OGC API - Common, allowing for other API architectures outside the OGC API framework to adopt it. However, a server under the OGC APIs is expected to implement OGC API - Common. If so, in practice, this means that the landing page and the conformance page follow OGC API - Common - Part 1: Core.

## 8.3. 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 to root resources. This standard extension requires a new link in the landing page for getting a description of the URL that allows for retrieving tiles of one or more resources.

### 8.3.1. Response

| **Requirement 12** | **/req/tiles/root/root-success** |
|---|---|
| A | If the API has mechanism to expose root resources (e.g. a landing page), the API SHALL advertise a URI to retrieve tile definitions defined by this service as links to the descriptions paths with rel: `tiles`. |

In the landing page, in JSON format, the links follow the link schema defined in the OGC API - Common or in OGC API - Features v1. Below you can find an example fragment of the response to an OGC API - Tiles landing page showing the link to root tiles.

*Example 4. API Landing Page fragment that advertises the path to get tiles for more than one collection*

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

## 8.4. Declaration of conformance classes

### 8.4.1. Response

The conformance page mainly consists of a list of links.

| **Requirement 13** | **/req/tiles/root/conformance-success** |
|---|---|
| A | If the API has a mechanism to advertise conformance classes, the API 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/conf/root. |

On the conformance page (typically in JSON format) the links follow the link schema defined in the OGC API – Common draft specification. The following is an example fragment from the response to an OGC API - Tiles conformance information page showing the support for *tiles from more than one collection*

*Example 5. Conformance Information Page fragment*

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

## 8.5. Root tiles description

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

### 8.5.1. Operation

| **Requirement 14** | **/req/tiles/root/ts-op** |
|---|---|
| A | The server SHALL support an operation to retrieve the description of the root tiles available as a HTTP GET request to a URI that is composed by two parts: the first part is the URI of a resource that can be represented as tiles (e.g. `/map` or simply `/`) and the second part follows the pattern `/tiles`. |

The request of this operation has no parameters.

### 8.5.2. Response

A successful response to a tiles request for a root tiles will return a data structure with a link to get tiles representing the resources and other relevant resources. This requirements class, the response only requires a URL template to retrieve a tile.

| **Requirement 15** | **/req/tiles/root/ts-tile-examples** |
|---|---|
| A | The content of the response to a successful execution SHALL include at least one link to a tile URI template (rel: `item`). |
| B | These links SHALL provide a URL template composed by the URL of this resource 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 indicated in the 'type' attribute of the link) |

One common order used in URL templates for tiles is: tiles/{tileMatrixSetId}/{tileMatrix}/{tileRow}/{tileCol}. However, this draft specification allows for other URL template composition.

*Table 2. URI template variables for tiles and possible values*

| URL template variable | Meaning | Possible values |
|---|---|---|
| TileMatrixSetId | tile matrix set identifier | The identifiers included in Annex D of OGC 17-083r2 or defined by extensions of the core requirements class. |
| 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 |
| TileCol | column index of tile matrix | A non-negative integer between 0 and the MatrixWidth – 1. If there is a TileMatrixSetLimits the value is limited between MinTileCol and MaxTileCol |

*Example 6. API tiles response fragment with the link to retrieve tiles*

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

In general, the `tileMatrixSetLinks` and the `tileMatrixSetLimits` can be determined by examining this information in the individual geospatial resource `tiles` of geospatial resource. In some cases, the server could also include the `tileMatrixSetLinks` data structure as part of the response to this operation. Clients should be prepared to determine if a `tileMatrixSetLinks` data structure is not provided in certain combinations of geospatial resources by examining the tileMatrixSet values and limits from the information in the individual geospatial resources and calculating the limits as the most restrictive intersection of them.

## 8.6. Tiles

This operation allows retrieving a single tile that represents information coming from one or more geospatial resources.

### 8.6.1. Operation

| **Requirement 16** | **/req/tiles/root/tcs-op** |
|---|---|
| A | The server SHALL support a set of HTTP GET operations following a URL template composed by the the root tile resource URL followed by values that substitute the variables {tileMatrixSetId}, {tileMatrix}, {tileRow} and {tileCol}. |

One common order used in URL templates for tiles is: tiles/{tileMatrixSetId}/{tileMatrix}/{tileRow}/{tileCol}. However, this draft specification allows for other URL template composition.

### 8.6.2. Parameter tileMatrixSetId

| **Requirement 17** | **/req/tiles/root/tcs-tilematrixsetid-definition** |
|---|---|
| A | The operation SHALL support a parameter `tileMatrixSetId` with the following characteristics (shown as OpenAPI Specification 3.0 fragment):<br>`name: tileMatrixSetId`<br>`in: path`<br>`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.`<br>`required: true`<br>`schema:`<br>`  type: string`<br>`example: WebMercatorQuad` |

### 8.6.3. Parameter tileMatrix

| **Requirement 18** | **/req/tiles/root/tcs-tilematrix-definition** |
|---|---|

| | |
|---|---|
| A | The operation SHALL support a parameter `tileMatrix` with the following characteristics (shown as OpenAPI Specification 3.0 fragment):<br><br>`name: tileMatrix`<br>`in: path`<br>`description: Identifier selecting one of the scales defined in the TileMatrixSet and representing the scaleDenominator the tile.`<br>`required: true`<br>`schema:`<br>`  type: string`<br>`example: '11'` |

## 8.6.4. Parameter tileRow

| **Requirement 19** | **/req/tiles/root/tcs-tilerow-definition** |
|---|---|
| A | The operation SHALL support a parameter `tileRow` with the following characteristics (shown as OpenAPI Specification 3.0 fragment):<br><br>`name: tileRow`<br>`in: path`<br>`description: Row index of the tile on the selected TileMatrix. It cannot exceed the MatrixWidth-1 for the selected TileMatrix`<br>`required: true`<br>`schema:`<br>`  type: integer`<br>`  minimum: 0`<br>`example: '827'` |

## 8.6.5. Parameter tileCol

| **Requirement 20** | **/req/tiles/root/tcs-tilecol-definition** |
|---|---|

| | |
|---|---|
| A | The operation SHALL support a parameter `tileCol` with the following characteristics (shown as OpenAPI Specification 3.0 fragment):<br><br>`name: tileCol`<br>`in: path`<br>`description: Column index of the tile on the selected TileMatrix. It cannot exceed the MatrixHeight-1 for the selected TileMatrix.`<br>`required: true`<br>`schema:`<br>`  type: integer`<br>`  minimum: 0`<br>`example: 1231` |

## 8.6.6. Parameter Resources

| **Requirement 21** | **/req/tiles/root/tcs-root-definition** |
|---|---|
| A | The operation SHALL support an optional parameter `resources` with the following characteristics (shown as OpenAPI Specification 3.0 fragment)<br><br>`name: resources`<br>`in: query`<br>`required: false`<br>`style: form`<br>`explode: false`<br>`schema:`<br>`  type: array`<br>`  items:`<br>`    type: string` |
| B | The parameter `resources` SHALL contain a comma-separated list of geospatial resource identifiers (collectionId’s) or a comma-separated list of full URLs to geospatial resource identifiers. |
| C | Only the geospatial resource identifiers that advertise a link with type=`tiles` in the geospatial resource description SHALL be included. |
| D | Only geospatial resources that support the same TileMatrixSetId parameter value SHALL be included. |

| Recommendation 4 | /rec/tiles/root/tcs-root-definition |
|---|---|
| A | If the parameter `resources` is missing, and when it is possible and sensible, all geospatial resources supporting the TileMatrixSetId parameter value SHOULD be represented in the tiles. |

| Permission 3 | /per/tiles/root/tcs-root-definition |
|---|---|
| A | If the parameter `resources` is missing and if it is not possible and sensible to represent all geospatial resources in tiles (e.g. it compromises performance or tiles are become packed with too many elements), the server is allowed to select only the most significant geospatial resources. |

### 8.6.7. Response

A successful response to a tile request is consistent with the media type of resource requested. This draft specification does not impose any media type. For example, for features the media type may be GeoJSON or Mapbox Vector Tiles, for coverages it may be a GeoTIFF, and for maps it may be a JPEG or a PNG.

| Requirement 22 | /req/tiles/root/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. |

### 8.6.8. Error conditions

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

If the value of the parameter `resources` contains a resource id of URI that does not exist on the API, the status code of the response is 404.

If the value of the parameter `resources` has a wrong format or combines one of more geospatial resources that are not compatible with the `tileMatrixSetId` value, the status code of the response is 500.

# 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 Core

| **Conformance Class** | |
|---|---|
| http://www.opengis.net/spec/ogcapi-tiles-1/1.0/conf/core | |
| Target type | Web API |
| Requirements class | http://www.opengis.net/spec/ogcapi-tiles-1/1.0/req/core |

### A.1.1. Declaration of conformance classes

#### A.1.1.1. Response

| **Abstract Test 1** | **/ats/core/conformance-success** |
|---|---|
| Test Purpose | Validate that the Conformance Declaration response complies with the required structure and contents. |
| Requirement | /req/tiles/core/conformance-success |
| Test Method | 1. Validate the response document against OpenAPI 3.0 schema confClasses.yaml<br>2. Validate that the document includes the conformance class "http://www.opengis.net/spec/ogcapi-tiles-1/1.0/conf/core"<br>3. Validate that the document lists all OGC API conformance classes that the API implements. |

### A.1.2. Tiles description

#### A.1.2.1. Tiles description path

| **Abstract Test 2** | **/ats/core/sct-op** |
|---|---|
| Test Purpose | Validate that information about the Tiles can be retrieved from the expected location. |

|  |  |
|---|---|
| Requirement | /req/tiles/core/sct-op |
| Test Method | 1. Issue an HTTP GET request to the URL {geospatial resource}/tiles<br>2. Validate that a document was returned with a status code 200 |

### A.1.2.2. Tiles description Link

| **Abstract Test 3** | **/ats/core/tc-tile-desc-links** |
|---|---|
| Test Purpose | Validate that the description of the tiles presents a tile representation of a geospatial resource and with rel: "tiles" |
| Requirement | /req/tiles/core/tc-tile-desc-links |
| Test Method | Verify that the response document includes:<br>1. a link to this response document (relation: self),<br>2. a link to the response document in every other media type supported by the server (relation: alternate).<br>3. a link with the href pointing to the description of the tiles that presents a tile representation of this geospatial resource and with relation: tiles. |

### A.1.2.3. Response

| **Abstract Test 4** | **/ats/core/sct-tmxslink** |
|---|---|
| Test Purpose | Verify that the response to a successful execution to a tiles description contains tileMatrixSetLinks |
| Requirement | /req/tiles/core/sct-tmxslink |

| | |
|---|---|
| Test Method | 1. Validate that the response document contains a property tileMatrixSetLinks<br>2. Validate the document against the schema using an JSON Schema validator.<br><br>`tileMatrixSetLink-set:`<br>`description: This list of tileMatrixSetLink objects, as defined in OGC 17-083r2 supported by this collectionId.`<br>`      type: array`<br>`      items:`<br>`$ref: '#/components/schemas/tileMatrixSetLink- entry'`<br>`tileMatrixSetLink-entry: type: object`<br>`required:`<br>`- tileMatrixSet properties:`<br>`        tileMatrixSet:`<br>`          type: string`<br>`          example: 'WebMercatorQuad'`<br>`        tileMatrixSetURI:`<br>`          type: string`<br>`          format: uri`<br>`          example:`<br>`'http://www.opengis.net/def/tilematrixset/OGC/1.0/WebMercatorQuad'` |

| **Abstract Test 5** | **/ats/core/sct-tile-examples** |
|---|---|
| Test Purpose | Verify that the response to a successful execution to a tiles description includes the required tile URI templates. |
| Requirement | /req/tiles/core/sct-tile-examples |

| | |
|---|---|
| Test Method | 1. Verify that the content of the response to a successful execution of a tile description includes at least a link to a tile URI template (rel: item).<br>2. Verify that the links provide a URL template with the fragment /tiles followed by the variables {tileMatrixSetId}, {tileMatrix}, {tileRow} and {tileCol}.<br>3. Verify that there is a link to a tile URI template for each file format that the server supports (the format is indicated in the 'type' attribute of the link).<br>4. Verify that a property 'templated' is part of the link properties to indicate that the link needs to be processed to substitute the templated variables with valid values before being used as a URL to a tile. |

### A.1.3. A tile from a geospatial resource

#### A.1.3.1. Operation

| **Abstract Test 6** | **/ats/core/tc-op** |
|---|---|
| Test Purpose | Validate that a tile can be retrieved from the expected location. |
| Requirement | /req/tiles/core/tc-op |
| Test Method | 1. Issue an HTTP GET request to the URL with pattern /tiles/{tileMatrixSetId}/{tileMatrix}/{tileRow}/{tileCol}.<br>2. Validate that a content was returned with a status code 200<br>3. Validate the contents of the returned feature using test /ats/core/tc-success. |

#### A.1.3.2. Parameter tileMatrixSetId

| **Abstract Test 7** | **/ats/core/tc-tilematrixsetid-definition** |
|---|---|
| Test Purpose | Validate that the tileMatrixSetId parameters are constructed correctly. |
| Requirement | /req/tiles/core/tc-tilematrixsetid-definition |

| | |
|---|---|
| Test Method | Verify that the tileMatrixSetId parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):<br><br>`name: tileMatrixSetId`<br>`in: path`<br>`description: Identifier of a specific tiling scheme. It can be one of those specified in Annex D.1 of the OGC 17-083r2 standard or one defined in this service.`<br>`    required: true`<br>`    schema:`<br>`      type: string`<br>`    example: WebMercatorQuad` |

### A.1.3.3. Parameter tileMatrix

| **Abstract Test 8** | **/ats/core/tc-tilematrix-definition** |
|---|---|
| Test Purpose | Validate that the tileMatrix parameters are constructed correctly. |
| Requirement | /req/tiles/core/tc-tilematrix-definition |
| Test Method | Verify that the tileMatrix parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):<br><br>`name: tileMatrix`<br>`    in: path`<br>`    description: Identifier selecting one of the scales defined in the TileMatrixSet and representing the scaleDenominator the tile.`<br>`    required: true`<br>`    schema:`<br>`      type: string`<br>`    example: '11'` |

### A.1.3.4. Parameter tileRow

| **Abstract Test 9** | **/ats/core/tc-tilerow-definition** |
|---|---|
| Test Purpose | Validate that the tileRow parameters are constructed correctly. |
| Requirement | /req/tiles/core/tc-tilerow-definition |

|  |  |
|---|---|
| Test Method | Verify that the tileRow parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):<br><pre>name: tileRow<br>   in: path<br>   description: Row index of the tile on the selected<br>TileMatrix. It cannot exceed the MatrixWidth-1 for the<br>selected TileMatrix<br>   required: true<br>   schema:<br>     type: integer<br>     minimum: 0<br>   example: '827'</pre> |

### A.1.3.5. Parameter tileCol

| **Abstract Test 10** | **/ats/core/tc-tilecol-definition** |
|---|---|
| Test Purpose | Validate that the tileCol parameters are constructed correctly. |
| Requirement | /req/tiles/core/tc-tilecol-definition |
| Test Method | Verify that the tileCol parameter complies with the following definition (using an OpenAPI Specification 3.0 fragment):<br><pre>name: tileCol<br>    in: path<br>    description: Column index of the tile on the<br>selected<br>TileMatrix. It cannot exceed the MatrixHeight-1 for the<br>selected TileMatrix.<br>    required: true<br>    schema:<br>      type: integer<br>      minimum: 0<br>    example: 1231</pre> |

### A.1.3.6. Response

| **Abstract Test 11** | **/ats/core/tc-success** |
|---|---|
| Test Purpose | Validate that the response complies with the required format, structure and contents. |

| | |
|---|---|
| Requirement | /req/tiles/core/tc-success |
| Test Method | 1. Validate that a successful execution of the operation is reported with a HTTP status code 200.<br>2. Validate that the content of that response is consistent with the format requested and represents elements inside or intersecting with the spatial extent of the geographical area of the tile identified by TileMatrixSet, TileMatrix, TileRow and TileCol. |

## A.2. Conformance Class Root

| **Conformance Class** | |
|---|---|
| http://www.opengis.net/spec/ogcapi-tiles-1/1.0/conf/root | |
| Target type | Web API |
| Requirements class | http://www.opengis.net/spec/ogcapi-tiles-1/1.0/req/root |

### A.2.1. API landing page

#### A.2.1.1. Response

| **Abstract Test 12** | **/ats/root/root-success** |
|---|---|
| Test Purpose | Validate that the API advertises a URI for retrieving tile definitions defined by the service as links to the descriptions paths with rel: tiles. |
| Requirement | /req/tiles/root/root-success |
| Test Method | Verify that the API advertises a URI, as links to the descriptions paths with `rel: tiles`, for retrieving tile definitions defined by the service. |

### A.2.2. Declaration of conformance classes

#### A.2.2.1. Response

| **Abstract Test 13** | **/ats/root/conformance-success** |
|---|---|

| | |
|---|---|
| Test Purpose | If the API has a mechanism to advertise conformance classes, validate that the API advertises the capability of generating tiles from multiple collections adding the conformance class for this capability |
| Requirement | /req/tiles/root/conformance-success |
| Test Method | Validate that the mechanism the API uses for advertising conformance classes includes the URI `http://www.opengis.net/spec/ogcapi-tiles-1/1.0/conf/root`. |

### A.2.3. Root tiles description

#### A.2.3.1. Operation

| **Abstract Test 14** | **/ats/root/ts-op** |
|---|---|
| Test Purpose | Validate that the server supports retrieval of descriptions of the root tiles |
| Requirement | /req/tiles/root/ts-op |
| Test Method | 1. Validate that the first part of the URL is the URI of a resource that can be represented as tiles (e.g. /map or simply /)<br>2. Validate that the second part of the URL follows the pattern `/tiles`.<br>3. Issue an HTTP GET request to the URL provided for retrieving the description of the root tiles<br>4. Validate that a document was returned with a status code 200 |

#### A.2.3.2. Response

| **Abstract Test 15** | **/ats/root/ts-tile-examples** |
|---|---|
| Test Purpose | Validate that the response to a tiles request for a root tiles returns a data structure with a link to get tiles representing the resources. |
| Requirement | /req/tiles/root/ts-tile-examples |

| | |
|---|---|
| Test Method | 1. Verify that the response includes at least one link labeled with the relation type `rel: item`<br>2. Verify that links with `rel: item` follow a URL template consisting consisting of the variables {tileMatrixSetId}, {tileMatrix}, {tileRow} and {tileCol}<br>3. If multiple links are provided with the same URL template, verify that they have a different `type` attribute value (each indicating a different format) |

### A.2.4. Tiles

#### A.2.4.1. Operation

| **Abstract Test 16** | **/ats/root/tcs-op** |
|---|---|
| Test Purpose | Validate that tiles can be retrieved using the URL templates provided by the server. |
| Requirement | /req/tiles/root/tcs-op |
| Test Method | 1. For every URL template provided by the server, issue an HTTP GET request to the URL, substituting the variables {tileMatrixSetId}, {tileMatrix}, {tileRow} and {tileCol} appropriately.<br>2. Validate that responses are returned with a status code 200.<br><br>Repeat the steps above for each format using the values indicated by the `type` attribute. |

#### A.2.4.2. Parameter tileMatrixSetId

| **Abstract Test 17** | **/ats/root/tcs-tilematrixsetid-definition** |
|---|---|
| Test Purpose | Verify that the operation supports a parameter `tileMatrixSetId`. |
| Requirement | /req/tiles/root/tcs-tilematrixsetid-definition |

| | |
|---|---|
| Test Method | 1. Issue an HTTP GET request with a `tileMatrixSetId` parameter value constructed following schema the below.<br>`name: tileMatrixSetId`<br>`in: path`<br>`description: Identifier of a specific tiling scheme.`<br>`It can be one of the specified in Annex D.1 of the OGC`<br>`17-083r2 standard or one defined in this service.`<br>`    required: true`<br>`    schema:`<br>`      type: string`<br>`    example: WebMercatorQuad`<br>2. Validate that responses are returned with a status code 200. |

### A.2.4.3. Parameter tileMatrix

| **Abstract Test 18** | **/ats/root/tcs-tilematrix-definition** |
|---|---|
| Test Purpose | Verify that the operation supports a parameter `tileMatrix`. |
| Requirement | /req/tiles/root/tcs-tilematrix-definition |
| Test Method | 1. Issue an HTTP GET request with a `tileMatrix` parameter value constructed following schema the below.<br>`name: tileMatrix`<br>`   in: path`<br>`   description: Identifier selecting one of the scales`<br>`defined in the TileMatrixSet and representing the`<br>`scaleDenominator the tile.`<br>`   required: true`<br>`   schema:`<br>`     type: string`<br>`   example: '11'`<br>2. Validate that responses are returned with a status code 200. |

### A.2.4.4. Parameter tileRow

| **Abstract Test 19** | **/ats/root/tcs-tilerow-definition** |
|---|---|
| Test Purpose | Verify that the operation supports a parameter `tileRow`. |

| | |
|---|---|
| Requirement | /req/tiles/root/tcs-tilerow-definition |
| Test Method | 1. Issue an HTTP GET request with a `tileRow` parameter value constructed following schema the below.<br>`name: tileRow`<br>`    in: path`<br>`    description: Row index of the tile on the selected`<br>`TileMatrix. It cannot exceed the MatrixWidth-1 for the`<br>`selected TileMatrix`<br>`    required: true`<br>`    schema:`<br>`      type: integer`<br>`      minimum: 0`<br>`    example: '827'`<br>2. Validate that responses are returned with a status code 200. |

### A.2.4.5. Parameter tileCol

| **Abstract Test 20** | **/ats/root/tcs-tilecol-definition** |
|---|---|
| Test Purpose | Verify that the operation supports a parameter `tileCol`. |
| Requirement | /req/tiles/root/tcs-tilecol-definition |
| Test Method | 1. Issue an HTTP GET request with a `tileCol` parameter value constructed following schema the below.<br>`name: tileCol`<br>`    in: path`<br>`    description: Column index of the tile on the`<br>`selected`<br>`TileMatrix. It cannot exceed the MatrixHeight-1 for the`<br>`selected TileMatrix.`<br>`    required: true`<br>`    schema:`<br>`      type: integer`<br>`      minimum: 0`<br>`    example: 1231`<br>2. Validate that responses are returned with a status code 200. |

### A.2.4.6. Parameter Resources

| **Abstract Test 21** | **/ats/root/tcs-root-definition** |
|---|---|
| Test Purpose | Verify that the operation optionally supports a parameter `resources`. |
| Requirement | /req/tiles/root/tcs-root-definition |
| Test Method | 1. Validate that the operation returns an HTTP 200 code for a `resources` parameter constructed following schema the below.<br><br>`name: resources`<br>`    in: query`<br>`    required: false`<br>`    style: form`<br>`    explode: false`<br>`    schema:`<br>`      type: array`<br>`      items:`<br>`        type: string`<br><br>2. Verify that `resources` parameter only contains a comma-separated list of geospatial resource identifiers (collectionId's) or a comma- separated list of full URLs to geospatial resource identifiers.<br>3. Verify that the geospatial resource identifiers advertise links with type=tiles in the geospatial resource description<br>4. Verify that the geospatial resource identifiers support the same TileMatrixSetId parameter value |

### A.2.4.7. Response

| **Abstract Test 22** | **/ats/root/tcs-success** |
|---|---|
| Test Purpose | Verify that the response to a tile request is consistent with the spatial extent and media type of resource requested. |
| Requirement | /req/tiles/root/tcs-success |

| | |
|---|---|
| Test Method | 1. Verify that the format of the response is as requested.<br>2. Verify that the spatial extent of the contents of the response are inside or intersecting with the spatial extent of the geographical area of the tile identified by TileMatrixSet, TileMatrix, TileRow and TileCol in the request.<br>3. Verify that the content of the response complies with the scale denominator represented by the TileMatrix identified.<br>4. Validate that the response is returned with a status code 200. |

# Annex B: Multi-layer Tile Support (Informative)

This draft specification does not impose any limits on the number of data layers that are included in a single tile. The server is therefore allowed to return a tile consisting of multiple data layers, where each individual data layer, or the set of data layers as a whole, may correspond to a collection. Such tiles are referred to as "multi-layer tiles".

Metadata about single or multi-layer tiles may be serialized as JSON, for example using the Mapbox TileJSON [https://github.com/mapbox/tilejson-spec] format. TileJSON conveys information such as the layers found within a tileset, the fields for attribute information, the vector geometry type, the zoom levels as well as a simple URL template for retrieving the tiles themselves. An example TileJSON document is shown in the following section.

## B.1. Example TileJSON document

The following TileJSON could be retrieved from a URL such as: `https://someserver.ogc.org/tiles/collections/vtp_daraa/tiles/{tileMatrixSetId}?f=application%2Fjson`

**NOTE** The OGC Vector Tiles Pilot Phase 2 (VTP2) initiative successfully proved that the TileJSON documents could be served from a URL such as: `https://someserver.ogc.org/tiles/collections/vtp_daraa/tiles/{tileMatrixSetId}/metadata?f=application%2Fjson`

```
{
  "name": "vtp_daraa",
  "scheme": "xyz",
  "tiles": [

"https://someserver.ogc.org/tiles/collections/vtp_daraa/tiles/WebMercatorQuad/{z}/{y}/
{x}?f=application%2Fvnd.mapbox-vector-tile"
  ],
  "center": [
    36.56250000000041,
    34.27502568554792,
    6
  ],
  "bounds": [
    35.8995094299316,
    32.4131851196289,
    36.5781326293945,
    33.1460647583008
  ],
  "vector_layers": [
    {
      "id": "AgricultureSrf",
      "fields": {
        "OTH": "string",
        "PVH": "number",
        "TSCL": "number",
        "ZI005_FNA": "string",
        "CDR": "string",
        "..." : "..."
      },
      "geometry_type": "polygon"
    },
    {
      "id": "VegetationSrf",
      "fields": {
        "LZN": "number",
        "OTH": "string",
        "PVH": "number",
        "TRE": "integer",
        "..." : "..."
      },
      "geometry_type": "polygon"
    },
    {
      "id": "MilitarySrf",
      "fields": {
        "OTH": "string",
        "WD3": "number",
        "FRT": "integer",
        "FRT3": "integer",
        "FRT2": "integer",
        "ZI005_FNA": "string",
        "..." : "..."
      },
      "geometry_type": "polygon"
    },
    "..."
  ]
}
```

The **draft** schema for TileJSON 3.0.0 is presented below for reference.

```
{
    "name": "TileJSON",
    "type": "object",
    "properties": {
        "tilejson": {
            "type": "string",
            "pattern": "\\d+\\.\\d+\\.\\d+\\w?[\\w\\d]*"
        },
        "tiles": {
            "type": "array",
            "items": {
                "type": "string"
            }
        },
        "vector_layers": {
            "type": "array",
            "items": {
                "type": "object",
                "properties": {
                    "id": {
                        "type": "string"
                    },
                    "fields": {
                        "type": "object",
                        "additionalProperties": { "type": "string" }
                    },
                    "description": {
                        "type": "string"
                    },
                    "maxzoom": {
                        "type": "integer"
                    },
                    "minzoom": {
                        "type": "integer"
                    }
                },
                "required": [ "id", "fields" ],
                "additionalProperties": true
            }
        },
        "attribution": {
            "type": "string"
        },
        "bounds": {
            "type": "array",
            "items": {
                "type": "number"
            }
        },
        "center": {
            "type": "array",
            "items": {
                "type": "number"
            }
        },
        "data": {
            "type": "array",
            "items": {
                "type": "string"
            }
        },
        "description": {
            "type": "string"
        },
        "fillzoom": {
            "minimum": 0,
            "maximum": 30,
            "type": "integer"
        },
        "grids": {
            "type": "array",
            "items": {
                "type": "string"
            }
        },
        "legend": {
            "type": "string"
        },
        "maxzoom": {
            "minimum": 0,
            "maximum": 30,
            "type": "integer"
        },
        "minzoom": {
            "minimum": 0,
            "maximum": 30,
            "type": "integer"
        },
        "name": {
            "type": "string"
        },
        "scheme": {
            "type": "string"
        },
        "template": {
            "type": "string"
        },
        "version": {
            "type": "string",
            "pattern": "\\d+\\.\\d+\\.\\d+\\w?[\\w\\d]*"
        }
    },
    "required": ["tilejson", "tiles", "vector_layers"],
    "additionalProperties": true
}
```

# Annex C: Revision History

| Date | Release | Editor | Primary clauses modified | Description |
|---|---|---|---|---|
| 2019-03-21 | Template | C. Heazel | all | initial template |
| 2020-04-15 | 0.0.1 | J. Maso | all | Several |
| 2019-04-21 | 0.0.2 | J. Maso | all | Several |
| 2019-05-21 | 0.0.3 | G. Hobona | Annex A | Fixed Conformance Class URI and added abstract tests |

# Annex D: Bibliography

- W3C/OGC: Spatial Data on the Web Best Practices, W3C Working Group Note 28 September 2017, https://www.w3.org/TR/sdw-bp/
- W3C: Data on the Web Best Practices, W3C Recommendation 31 January 2017, https://www.w3.org/TR/dwbp/
- W3C: Data Catalog Vocabulary, W3C Recommendation 16 January 2014, https://www.w3.org/TR/vocab-dcat/
- IANA: Link Relation Types, https://www.iana.org/assignments/link-relations/link-relations.xml
- Mapbox: Mapbox Vector Tiles Specification, https://docs.mapbox.com/vector-tiles/specification/