OGC® DOCUMENT: YY-XXX

External identifier of this OGC® document: http://www.opengis.net/doc/IS/ogcapi-joins-1/1.0

OGC®
Making location count.

OGC API - JOINS - PART 1: CORE

STANDARD

DRAFT

Version: 1.0.0-SNAPSHOT
Submission Date: yyyy-mm-dd
Approval Date: yyyy-mm-dd
Publication Date: yyyy-mm-dd
Editor: Pekka Latvala

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.

None of the Intellectual Property or underlying information or technology may be downloaded or otherwise exported or reexported in violation of U.S. export laws and regulations. In addition, you are responsible for complying with any local laws in your jurisdiction which may impact your right to import, export or use the Intellectual Property, and you represent that you have complied with any regulations or registration procedures required by applicable law to make this license enforceable.

Suggested additions, changes and comments on this document 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

Copyright notice

Note

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

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

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF RECOMMENDATIONS

I ABSTRACT

This document is the specification for the core module of the OGC API — Joins standard. The core module specifies a service interface that allows attribute data to be joined either with collections that are hosted on the server or directly with inputted files. The core module contains support for CSV and GeoJSON files. The joining of the datasets is executed via common identifiers that are available in both datasets. The OGC API — Joins core module supports also operations for viewing metadata and key values on collections that are available on the server. It contains also operations for accessing and deleting the created joinsd.

II KEYWORDS

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

ogcdoc, OGC document, API, openapi, html, joins

III

PREFACE

This document defines the core module of the OGC API — Joins standard. The specification is a multi-part document that can be extended by specifying extension modules to the core module.

This document does not suggest any updates to the OGC Abstract Specification

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

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

IV SECURITY CONSIDERATIONS

No security considerations have been made for this document.

V SUBMITTING ORGANIZATIONS

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

- Finnish Geospatial Research Institute / National Land Survey of Finland

1

SCOPE

1 SCOPE

This OpenGIS® standard specifies a Web API that defines the core module for the OGC API — Joins specification.

The specification contains operations for obtaining general information on the service implementation. These include operations for accessing the API landing page, the API definition file and the service's conformance information.

The specification contains also functionalities for retrieving metadata and key values on the collections that are available on the server, operation for joining attribute data from attribute data files with these collections or inputted files and accessing and deleting the created joins.

The core module contains support for attribute data files in the CSV format and for spatial data files in the GeoJSON format. The support for other data formats may be defined in potential extension modules.

2

CONFORMANCE

2 CONFORMANCE

This standard defines the requirement class “core” and encoding requirements classes “JSON”, “HTML” and “GeoJSON”

The standardization targets of all conformance classes are “Web APIs.”

The main requirements class is:

- Core.

The core requirements class specifies requirements that all Web APIs have to implement.

The JSON encoding format is mandatory to be supported for service responses.

The support for HTML encoding is recommended.

The GeoJSON encoding is mandatory to be supported for the joined data outputs.

The CSV format is mandatory to be supported for the join inputs.

The GeoJSON format input may be supported for the file joining functionality.

The Core does not mandate any encoding or format for the formal definition of the API. One option is the OpenAPI 3.0 specification and a requirements class has been specified in the OGC API — Common — Part 1.

The conformance class values defined in this core module are listed in Table 1.

Table 1 — Conformance classes defined in this module

CONFORMANCE CLASS	URI
Core	http://www.opengis.net/spec/ogcapi-joins-1/1.0/conf/core
Data joining	http://www.opengis.net/spec/ogcapi-joins-1/1.0/conf/core/data-joining
File joining	http://www.opengis.net/spec/ogcapi-joins-1/1.0/conf/core/file-joining
Join delete	http://www.opengis.net/spec/ogcapi-joins-1/1.0/conf/core/join-delete
HTML	http://www.opengis.net/spec/ogcapi-joins-1/1.0/conf/html
JSON	http://www.opengis.net/spec/ogcapi-joins-1/1.0/conf/json
Input	

CONFORMANCE CLASS	URI
File uploading with Query	http://www.opengis.net/spec/ogcapi-joins-1/1.0/conf/input/file-upload
File referencing with URL	http://www.opengis.net/spec/ogcapi-joins-1/1.0/conf/input/http-ref
CSV file input	http://www.opengis.net/spec/ogcapi-joins-1/1.0/conf/input/csv
GeoJSON file input	http://www.opengis.net/spec/ogcapi-joins-1/1.0/conf/input/geojson
Output	
GeoJSON output for joined data	http://www.opengis.net/spec/ogcapi-joins-1/1.0/conf/output/geojson
Direct GeoJSON output for joined data	http://www.opengis.net/spec/ogcapi-joins-1/1.0/conf/output/geojson-direct

In addition, the following conformance classes that are used from the specifications OGC API — Common — Part 1 and OGC API — Common — Part 2 are listed in Table 2.

Table 2 — Conformance classes

Conformance class	URI
OGC API — Common — Part 1	
Core	http://www.opengis.net/spec/ogcapi-common-1/1.0/conf/core
Landing Page	http://www.opengis.net/spec/ogcapi-common-1/1.0/conf/landing-page
OpenAPI 3.0	http://www.opengis.net/spec/ogcapi-common-1/1.0/conf/oas30
OGC API — Common — Part 2	
Collections	http://www.opengis.net/spec/ogcapi-common-2/1.0/conf/collections
Simple Query	http://www.opengis.net/spec/ogcapi-common-2/1.0/conf/simple-query

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.

3

NORMATIVE REFERENCES

3 NORMATIVE REFERENCES

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee: IETF RFC 2616, *Hypertext Transfer Protocol — HTTP/1.1*. Internet Engineering Task Force, Fremont, CA (1999). https://raw.githubusercontent.com/relaton/relaton-data-ietf/master/data/reference.RFC.2616.xml

E. Rescorla: IETF RFC 2818, *HTTP Over TLS*. Internet Engineering Task Force, Fremont, CA (2000). https://raw.githubusercontent.com/relaton/relaton-data-ietf/master/data/reference.RFC.2818.xml

G. Klyne, C. Newman: IETF RFC 3339, *Date and Time on the Internet: Timestamps*. Internet Engineering Task Force, Fremont, CA (2002). https://raw.githubusercontent.com/relaton/relaton-data-ietf/master/data/reference.RFC.3339.xml

T. Berners-Lee, R. Fielding, L. Masinter: IETF RFC 3986, *Uniform Resource Identifier (URI): Generic Syntax*. Internet Engineering Task Force, Fremont, CA (2005). https://raw.githubusercontent.com/relaton/relaton-data-ietf/master/data/reference.RFC.3986.xml

J. Reschke: IETF RFC 6266, *Use of the Content-Disposition Header Field in the Hypertext Transfer Protocol (HTTP)*. Internet Engineering Task Force, Fremont, CA (2011). https://raw.githubusercontent.com/relaton/relaton-data-ietf/master/data/reference.RFC.6266.xml

R. Fielding, J. Reschke: IETF RFC 7231, *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content*. Internet Engineering Task Force, Fremont, CA (2014). https://raw.githubusercontent.com/relaton/relaton-data-ietf/master/data/reference.RFC.7231.xml

L. Masinter: IETF RFC 7578, *Returning Values from Forms: multipart/form-data*. Internet Engineering Task Force, Fremont, CA (2015). https://raw.githubusercontent.com/relaton/relaton-data-ietf/master/data/reference.RFC.7578.xml

M. Nottingham, E. Wilde: IETF RFC 7807, *Problem Details for HTTP APIs*. Internet Engineering Task Force, Fremont, CA (2016). https://raw.githubusercontent.com/relaton/relaton-data-ietf/master/data/reference.RFC.7807.xml

H. Butler, M. Daly, A. Doyle, S. Gillies, S. Hagen, T. Schaub: IETF RFC 7946, *The GeoJSON Format*. Internet Engineering Task Force, Fremont, CA (2016). https://raw.githubusercontent.com/relaton/relaton-data-ietf/master/data/reference.RFC.7946.xml

T. Bray: IETF RFC 8259, *The JavaScript Object Notation (JSON) Data Interchange Format*. Internet Engineering Task Force, Fremont, CA (2017). https://raw.githubusercontent.com/relaton/relaton-data-ietf/master/data/reference.RFC.8259.xml

M. Nottingham: IETF RFC 8288, *Web Linking*. Internet Engineering Task Force, Fremont, CA (2017). https://raw.githubusercontent.com/relaton/relaton-data-ietf/master/data/reference.RFC.8288.xml

json-schema.org: **JSON Schema**, December 2020. Available at: https://json-schema.org/specification.html

Arliss Whiteside Jim Greenwood : OGC 06-121r9, *OGC Web Service Common Implementation Specification*. Open Geospatial Consortium (2010). https://portal.ogc.org/files/?artifact_id=38867

Heazel, C. (ed.): OGC: OGC 19-072, **OGC API — Common — Part 1: Core** (in development), 2021. Available at: http://docs.opengeospatial.org/DRAFTS/19-072.pdf

Heazel, C. (ed.): OGC: OGC 20-024, **OGC API — Common — Part 2: Geospatial Data** (in development), 2021. Available at: http://docs.opengeospatial.org/DRAFTS/20-024.pdf

Open API Initiative (OAI): **The OpenAPI specification 3.0**, 2020. Available at: https://github.com/OAI/OpenAPI-Specification/blob/master/versions/

Schema.org: http://schema.org/docs/schemas.html

W3C: **HTML5**, W3C Recommendation. Available at: http://www.w3.org/TR/html5/

4

TERMS AND DEFINITIONS

4 TERMS AND DEFINITIONS

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

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

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

This document uses the terms defined in Sub-clause 5.3 of OGC 06-121r9, 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. attribute dataset

Dataset that contains attribute information that can be joined with a spatial dataset through common identifiers.

4.2. spatial dataset

Dataset that contains geometry information.

5

CONVENTIONS

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/ogcapi-joins-1/1.0`.

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

5.2. Link relations

To express relationships between resources, RFC 8288 (Web Linking) is used.

The link relation types that are used in this document are listed in Table 3.

Table 3 — Link Relations

The following registered link relation types [IANA] are used in this document:

Link Relation	Purpose
alternate	Refers to a substitute for this context. Refers to a representation of the current resource that is encoded using another media type (the media type is specified in the `type` link attribute).
describedby	Refers to a resource providing information about the link's context. Links to external resources that further describe the subject resource.
license	Refers to a license associated with this context.
next	Indicates that the link's context is a part of a series, and that the next in the series is the link target.
prev	Indicates that the link's context is a part of a series, and that the previous in the series is the link target.
self	Conveys an identifier for the link's context.

	A link to another representation of this resource.
service-desc	Identifies service description for the context that is primarily intended for consumption by machines. API definitions are considered service descriptions.
service-doc	Identifies service documentation for the context that is primarily intended for human consumption.
service-meta	Identifies general metadata for the context that is primarily intended for consumption by machines.
In addition the following link relation types are used for which no applicable registered link relation type could be identified:	
Link Relation	**Purpose**
dataset	Refers to a resource that is comprised of the metadata of the specific collection that is available on the server.
http://www.opengis.net/def/rel/ogc/1.0/conformance	Refers to a resource that identifies the specifications that the link's context conforms to.
http://www.opengis.net/def/rel/ogc/1.0/data	Indicates that the link's context is a distribution of a dataset that is an API and refers to the root resource of the dataset in an API.
keys	Refers to a resource that is comprised of metadata of the key fields of the collection represented by the link's context.
key-values	Refers to a resource that is comprised of key values of the collection's key field represented by the link's context.
joins	Refers to a resource that is comprised of the metadata of the created joins that are available on the server.
join	Refers to a resource that is comprised of the metadata of the specific join that is available on the server.
output	Refers to an output of the join operation that contains the joined data.

5.2.1. Response Schema for the Link Object

The individual hyperlink elements that make up a "links" elements are defined in the hyperlink schema, originally defined in the section 6.3 of the OGC API — Common — Part 1 specification.

```
{
    "$schema": "http://json-schema.org/draft-07/schema#",
    "title": "Link Schema",
    "description": "Schema for external references",
    "type": "object",
    "required": [
        "href",
        "rel"
    ],
    "properties": {
        "href": {
```

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

Figure 1 — Hyperlink Schema

5.3. Use of HTTPS

For simplicity, this document in general only refers to the HTTP protocol. This is not meant to exclude the use of HTTPS and simply is a shorthand notation for “HTTP or HTTPS.” In fact, most servers are expected to use HTTPS, not HTTP.

OGC Web API standards do not prohibit the use of any valid HTTP option. However, implementers should be aware that optional capabilities that are not in common use could be an impediment to interoperability.

6

OVERVIEW

6 OVERVIEW

6.1. Encodings

This standard mandates the JSON encoding to be supported for service responses. In addition the support for HTML encoding is recommended.

The support for the GeoJSON format is mandatory for the joined data outputs.

7

REQUIREMENTS CLASS "CORE"

7 REQUIREMENTS CLASS "CORE"

REQUIREMENTS CLASS 1	
http://www.opengis.net/ogcapi-joins-1/1.0/req/core	
Obligation	requirement
Target type	Implementation Specification
Dependency	OGC 19-072 (OGC — API — Common: Part 1)
Dependency	OGC 20-024 (OGC — API — Common: Part 2)
Dependency	RFC 2616 (HTTP/1.1)
Dependency	RFC 2818 (HTTP over TLS)
Dependency	RFC 3339 (Date and Time on the Internet: Timestamps)
Dependency	RFC 7578 Returning Values from Forms: multipart/form-data
Dependency	RFC 8288 (Web Linking)

7.1. Overview

The core requirements class contains the following functionalities:

The `Landing page` (path `/`) provides an entry point to the API. It contains links to:

- The `API definition` (link relations `service-desc` and `service-doc`).
- The `Conformance declaration` (path `/conformance`, link relation `http://www.opengis.net/def/rel/ogc/1.0/conformance`).
- The `Collections` (path `/collections`, link relation `http://www.opengis.net/def/rel/ogc/1.0/data`).
- The `Joins` (path `/joins`, link relation `joins`).

The `API definition` describes the capabilities of the server that can be used by clients to connect to the server or by development tools to support the implementation of servers and clients. Accessing the `API definition` using HTTP GET returns a description of the API. The API definition can be hosted on the API server(s) or a separate server.

The `Conformance declaration` states the conformance 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 conformance classes implemented by the server.

Accessing the `Collections` using HTTP GET returns a the list of datasets that are available on the server.

Each `Collection` element in the Collections list can be accessed further in order to get the metadata on each indivual Collection by making an HTTP GET request at path `/collections/{collectionId}`.

Accessing the `Collection's Key Fields` using HTTP GET at path `/collections/{collectionId}/keys` provides information on the key fields of a Collection.

Accessing the `Collection's Key Field` using HTTP GET at path `/collections/{collectionId}/keys/{keyFieldId}` provides the key values of the specific key field. The data joining is executed through these key values.

Accessing the `Joins` using HTTP GET returns a the list of Joins that are available on the server.

New `Joins` can be created by making a HTTP POST query to the path `/joins`. This is done by joining attribute data from inputted attribute data file with the Collection available on the server. Attribute data can be also joined directly with inputted spatial data files.

Each `Join` element in the Joins list can be accessed further in order to get the metadata on each indivual Join by making an HTTP GET request at path `/joins/{joinId}`.

Each `Join` can be deleted by making a HTTP DELETE request to path `/joins/{joinId}`.

The core module contains support for:

- CSV format for the inputted attribute data files.
- GeoJSON format for the inputted spatial data files.

The Table 4 contains an overview of the operations specified in the core module.

Table 4 — Overview of the resources definecd in the OGC API — Joins core module

PATH	HTTP METHOD	DESCRIPTION
/	GET	API landing page
/conformance	GET	API conformance declaration
/collections	GET	Returns metadata on the collections available on the server
/collections/{collectionId}	GET	Returns metadata on a specific collection available on the server

PATH	HTTP METHOD	DESCRIPTION
/collections/{collectionId}/keys	GET	Returns the key fields of a specific collection
/collections/{collectionId}/keys/{keyFieldId}	GET	Returns the key values of a specific key field of a specific collection
/joins	GET	Returns a list of the joins available on the server
/joins	POST	Creates a new join by joining attribute data from a inputted attribute data file with a specific collection or directly with an inputted spatial data file
/joins/{joinId}	GET	Returns metadata on a specific join
/joins/{joinId}	DELETE	Deletes a specific join

7.2. HTTP 1.1

REQUIREMENT 1	
/req/core/http	
Obligation	requirement
A	The server SHALL conform to HTTP 1.1.
B	If the API supports HTTPS, then the API SHALL also conform to HTTP over TLS.

7.3. HTTP Status Codes

Table 5 lists the main HTTP status codes that clients should be prepared to receive. This includes support for specific security schemes or URI redirection. In addition, other error situations may occur in the transport layer outside of the server.

Table 5 — Typical HTTP status codes

Status code	Description

200	A successful request.
201	The request was executed successfully and it resulted in one or more resources that were created.
204	A request was executed successfully and there is no additional content in the response payload body.
302	The target resource was found but resides temporarily under a different URI. A 302 response is not evidence that the operation has been successfully completed.
303	The server is redirecting the user agent to a different resource. A 303 response is not evidence that the operation has been successfully completed.
304	An entity tag was provided in the request and the resource has not changed since the previous request.
307	The target resource resides temporarily under a different URI and the user agent MUST NOT change the request method if it performs an automatic redirection to that URI.
308	Indicates that the target resource has been assigned a new permanent URI and any future references to this resource ought to use one of the enclosed URIs.
400	The server cannot or will not process the request due to an apparent client error. For example, a query parameter had an incorrect value.
401	The request requires user authentication. The response includes a WWW-Authenticate header field containing a challenge applicable to the requested resource.
403	The server understood the request, but is refusing to fulfill it. While status code 401 indicates missing or bad authentication, status code 403 indicates that authentication is not the issue, but the client is not authorized to perform the requested operation on the resource.
404	The requested resource does not exist on the server. For example, a path parameter had an incorrect value.
405	The request method is not supported. For example, a POST request was submitted, but the resource only supports GET requests.
406	Content negotiation failed. For example, the Accept header submitted in the request did not support any of the media types supported by the server for the requested resource.
500	An internal error occurred in the server.

The return status codes described in Table 5 do not cover all possible conditions. See IETF RFC 7231 for a complete list of HTTP status codes.

PERMISSION 1	
/per/core/additional-status-codes	
Obligation	permission
Servers MAY implement additional capabilities provided by the HTTP protocol. Therefore, they MAY return status codes in addition to those listed in Table 5.	

When the server encounters an error during the processing of the request , the server may wish to include information in addition to the status code in the response. Since Web API interactions are often machine-to-machine, a machine-readable report would be preferred. IETF RFC 7807 addresses this need by providing “Problem Details” response schemas for both JSON and XML.

RECOMMENDATION 1	
/rec/core/problem-details	
Obligation	recommendation
The server SHOULD include a “problem details” report in any error response in accordance with IETF RFC 7807.	

7.4. Support for Cross-Origin Requests

If the data is located on another host than the webpage (“same-origin policy”), access to data from a HTML page is by default prohibited for security reasons. A typical example is a web-application accessing feature data from multiple distributed datasets.

RECOMMENDATION 2	
/rec/core/cross-origin	
Obligation	recommendation
If the server is intended to be accessed from a browser, cross-origin requests SHOULD be supported. Note that support can also be added in a proxy layer on top of the server.	

Two common mechanisms to support cross-origin requests are:

- Cross-origin resource sharing (CORS)
- JSONP (JSON with padding)

7.5. API Landing Page

The HTTP GET operation at service root path `{root}/` returns the API landing page document. The API landing page provides a starting point for the use of the API and it contains links to:

- API definition document.
- Conformance information.

- Metadata on collections available on the server.
- Joins available on the server.

7.5.1. Request

REQUIREMENT 2	
/req/core/root-op	
Obligation	requirement
A	The server SHALL support the HTTP GET operation on the URI `{root}/`.
B	The response to the HTTP GET request issued in A SHALL satisfy requirement /req/core/root-success.

7.5.2. Response

REQUIREMENT 3	
/req/core/root-success	
Obligation	requirement
A	A successful execution of the operation SHALL be reported as a response with an HTTP status code `200`.
B	The content of that response SHALL be based upon Landing Page Schema. The response SHALL include links to following resources: • `/api` (link rel property value: `service-desc` or `service-doc`) • `/conformance` (link rel property value: `http://www.opengis.net/def/rel/ogc/1.0/conformance`) • `/collections` (link rel property value: `http://www.opengis.net/def/rel/ogc/1.0/data`) • `/joins` (link rel property value: `joins`)

7.5.2.1. Response Schema for the Landing Page

The landing page response is based on the following schema (from OGC — API — Common: Part 1 document).

```
{
"$schema": "http://json-schema.org/draft-07/schema#",
"title": "Landing Page Schema",
"description": "JSON schema for the OGC API - Common landing page",
"type": "object",
"required": [
    "links"
    ],
"properties": {
    "title": {
        "title": "The title of the API.",
        "description": "While a title is not required, implementers are
 strongly advised to include one.",
        "type": "string"
        },
    "description": {
        "description": "A textual description of the API",
        "type": "string"
        },
    "attribution" : {
        "type" : "string",
        "title" : "attribution for the API",
        "description" : "The `attribution` should be short and intended for
 presentation to a user, for example, in a corner of a map. Parts of the text
 can be links to other resources if additional information is needed. The
 string can include HTML markup."
        },
    "links": {
        "description": "Links to the resources exposed through this API.",
        "type": "array",
        "items": {"$href": "link.json"}
        }
    },
    "additionalProperties": true
}
```

Figure 2 — Landing Page Schema

7.5.3. Error Situations

See HTTP status codes for general guidance.

7.6. API Definition

Servers SHOULD provide an API Definition resource that describes the capabilities of the server. This resource can be used by client developers to understand the supported services, by software clients to connect to the server, and by development tools to support the implementation of servers and clients.

7.6.1. Request

REQUIREMENT 4	
/req/core/api-definition-op	
Obligation	requirement
A	The server SHALL support the HTTP GET operation on all links from the landing page that have the relation type `service-desc`.
B	The server SHALL support the HTTP GET operation on all links from the landing page that have the relation type `service-doc`.
C	The responses to all HTTP GET requests issued in `A` and `B` SHALL satisfy the requirement /req/core/api-definition-success.

RECOMMENDATION 3	
/rec/core/api-definition-op	
Obligation	recommendation
A	The server SHOULD support the HTTP GET operation on the URI `{root}/api`.
B	The response to the HTTP GET request issued in A SHOULD satisfy the requirement /req/core/api-definition-success.

7.6.2. Response

REQUIREMENT 5	
/req/core/api-definition-success	
Obligation	requirement
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 an API Definition document.
C	The API Definition document SHALL be consistent with the media type identified through HTTP content negotiation.
D	Note. The -f parameter MAY be used to satisfy this requirement.

RECOMMENDATION 4	
/rec/core/api-definition-oas	
Obligation	recommendation
If the API definition document uses the OpenAPI Specification 3.0, THEN The document SHOULD conform to the OpenAPI Specification 3.0 requirements class `http://www.opengis.net/spec/ogcapi-common-1/1.0/req/oas30` (defined in the OGC API — Common — Part 1 document).	

7.6.3. Error Situations

See HTTP status codes for general guidance.

7.7. Declaration of Conformance Classes

The HTTP GET operation at path `{root}/conformance` returns a list of conformance classes that the server supports.

7.7.1. Request

REQUIREMENT 6	
/req/core/conformance-op	
Obligation	requirement
A	The server SHALL support the HTTP GET operation at the path `{root}/conformance`.
B	The server SHALL support the HTTP GET operation on all links from the landing page that have the relation type `http://www.opengis.net/def/rel/ogc/1.0/conformance`.
C	The responses to all HTTP GET requests issued in A and B SHALL satisfy requirement /req/core/conformance-success.

7.7.2. Response

REQUIREMENT 7	
/req/core/conformance-success	
Obligation	requirement
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 based upon Conformance Declaration Schema.

7.7.2.1. Response Schema for the Conformance Declaration

The schema for the conformance response is based on the following schema (from OGC API — Common — Part 1 document).

```
{
"$schema": "http://json-schema.org/draft-07/schema#",
"title": "Conformance Declaration Schema",
"description": "This schema defines the resource returned from the /Conformance
 path",
"type": "object",
"required": [
    "conformsTo"
    ],
"properties": {
    "conformsTo": {
        "type": "array",
        "description": "conformsTo is an array of URIs.  Each URI should
 correspond to a defined OGC Conformance class.  Unrecognized URIs should be
 ignored",
        "items": {
            "type": "string",
            "example": "http://www.opengis.net/spec/ogcapi-common-1/1.0/conf/
core"
            }
        }
    }
}
```

Figure 3 — Conformance Declaration Schema

7.7.3. Error Situations

See HTTP status codes for general guidance.

7.8. Collections

The HTTP GET operation at path `{root}/collections` returns metadata on the collections that are available on the server.

7.8.1. Request

REQUIREMENT 8	
/req/core/collections-get-op	
Obligation	requirement
The server SHALL support the HTTP GET operation at path `{root}/collections`.	

7.8.2. Response

REQUIREMENT 9	
/req/core/collections-get-success	
Obligation	requirement
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 based upon the Collections schema.

7.8.2.1. Response Schema for the Collections

The response is based on the following schema (from OGC — API — Common: Part 2 document).

```
{
"$schema": "http://json-schema.org/draft-07/schema#",
"title": "Collections Schema",
"description": "This schema defines the resource returned from /collections
 path.",
"type": "object",
"required": [
    "links",
    "collections"
    ],
"properties": {
    "links": {
        "type": "array",
        "items": {"$href": "link.json"}
```

```
        },
    "timeStamp": {
        "type": "string",
        "format": "date-time"
        },
    "numberMatched": {
        "type": "integer",
        "min": "0"
        },
    "numberReturned": {
        "type": "integer",
        "min": "0"
        },
    "collections": {
        "type": "array",
        "items": {"$href": "collectionDesc.json"}
        }
    }
}
```

Figure 4 — Collections Schema

This schema is further constrained by the following requirements and recommendations.

To support hypermedia navigation, the `links` property must be populated with sufficient hyperlinks to navigate through the whole dataset.

REQUIREMENT 10	
/req/core/collections-get-success-links	
Obligation	requirement
A	A `200`-response SHALL include the following links in the `links` property of the response: • A link to this response document (relation: `self`), • A link to the response document in every other media type supported by the service (relation: `alternate`).
B	All links SHALL include the `rel` and `type` link parameters.

Additional information may be available to assist in understanding and using this dataset. Links to those resources should be provided as well.

RECOMMENDATION 5	
/rec/core/collections-get-success-descriptions	
Obligation	recommendation
A	If external schemas or descriptions exist that provide additional information about the structure or semantics for the resource, a `200`-response SHOULD include links to each of those resources in the links property of the response (relation: `describedby`).

RECOMMENDATION 5	
B	The `type` link parameter SHOULD be provided for each link. This applies to resources that describe the whole dataset.

The `timeStamp` property of the Collections response indicates when the response was generated.

REQUIREMENT 11	
/req/core/collections-get-success-timestamp	
Obligation	requirement
If a property `timeStamp` is included in the response, the value SHALL be set to the time when the response was generated.	

The `collections` property of the Collections response provides a description of each individual collection hosted by the server. These descriptions are based on the Collection Schema.

REQUIREMENT 12	
/req/core/collections-get-success-items	
Obligation	requirement
A	For each dataset collection provided by the server, metadata describing that collection SHALL be provided in the `collections` property of the Collections response.
B	The content of that response SHALL comply with the requirements in the `http://www.opengis.net/spec/ogcapi-common-2/1.0/rm/collection` Requirements Class described in the OGC — API — Common: Part 2 document).

PERMISSION 2	
/per/core/collections-get-success-items	
Obligation	permission
To support servers with many collections, servers MAY limit the number of items in the property `collections`.	

7.8.3. Error Situations

See HTTP status codes for general guidance.

7.9. Collection

The HTTP GET operation at path `{root}/collections/{collectionId}` returns metadata on a specific collection available on the server.

7.9.1. Request

REQUIREMENT 13	
/req/core/collections-collectionid-get-op	
Obligation	requirement
A	The server SHALL support the HTTP GET operation at path `{root}/collections/{collectionId}`.
B	The parameter `collectionId` is each `id` property in the `collections` response (JSONPath: `$.collections[*].id`).

7.9.2. Response

REQUIREMENT 14	
/req/core/collections-collectionid-get-success	
Obligation	requirement
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 comply with the requirements in the http://www.opengis.net/spec/ogcapi-common-2/1.0/rm/collection Requirements Class defined in the OGC — API — Common: Part 2 document) with additions described in the Response Schema for the Collection section of this document.
C	The content of that response SHALL be consistent with the content for this collection in the `/collections` response. That is, the values for `id`, `title`, `description` and `extent` SHALL be identical.

7.9.2.1. Response Schema for the Collection

The Collection response is based on the following schema (from OGC — API — Common: Part 2 document).

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "Collection Resource Schema",
  "description": "This schema defines the resource returned from /collections/
{collectionId}.",
  "type": "object",
  "required": [
      "id",
      "links"
      ],
  "properties": {
      "id": {
          "description": "identifier of the collection used, for example, in
 URIs",
          "type": "string"
          },
      "title": {
          "description": "human readable title of the collection",
          "type": "string"
          },
      "description": {
          "description": "a description of the members of the collection",
          "type": "string"
          },
      "attribution" : {
          "type" : "string",
          "title" : "attribution for the collection",
          "description" : "The `attribution` should be short and intended for
 presentation to a user, for example, in a corner of a map. Parts of the text
 can be links to other resources if additional information is needed. The
 string can include HTML markup."
          },
      "links": {
          "type": "array",
          "items": {"$href": "link.json"}
          },
      "extent": {"$href": "extent.json"},
      "itemType": {
          "description": "An indicator about the type of the items in the
 collection.",
          "type": "string"
          },
      "crs": {
          "description": "the list of coordinate reference systems supported by
 the API; the first item is the default coordinate reference system",
          "type": "array",
          "items": {
              "type": "string"
              },
          "default": [
              "http://www.opengis.net/def/crs/OGC/1.3/CRS84"
              ],
          "example": [
              "http://www.opengis.net/def/crs/OGC/1.3/CRS84",
```

```
                "http://www.opengis.net/def/crs/EPSG/0/4326"
                ]
            }
        }
}
```

Figure 5 — Collection Resource Schema

The requirements of the response are originally defined in the http://www.opengis.net/spec/ogcapi-common-2/1.0/rm/collection Requirements Class described in the OGC — API — Common: Part 2 document.

7.9.2.1.1. Extent

REQUIREMENT 15	
/req/core/rc-md-extent	
Obligation	requirement
A	For each spatial collection resource, the `extent` property, if provided, SHALL define boundaries that encompass the spatial and temporal properties of all of the resources in this collection. The temporal extent may use null values to indicate an open time interval.
B	If a spatial resource has multiple properties with spatial or temporal information, it is the decision of the API implementation whether only a single spatial or temporal geometry property is used to determine the extent or all relevant geometries.

RECOMMENDATION 6	
/rec/core/rc-md-extent	
Obligation	recommendation
A	If an extent contains multiple spatial boundaries (multiple bbox, etc.), then the extent SHOULD include in the first bbox a boundary which represents the union of all of the other boundaries.
B	If an extent contains multiple temporal intervals, then the extent SHOULD include as the first interval an interval which represents the union of all of the other intervals.

RECOMMENDATION 7
/rec/core/rc-md-extent-single

RECOMMENDATION 7	
Obligation	recommendation
A	While the spatial and temporal extents support multiple bounding boxes (`bbox` array) and time intervals (`interval` array) for advanced use cases, implementations SHOULD provide only a single bounding box or time interval unless the use of multiple values is important for the use of the dataset and agents using the API are known to be support multiple bounding boxes or time intervals.

7.9.2.1.2. Item Type

The collections defined by this core module provide information on their key fields and key values.

RECOMMENDATION 8	
/rec/core/rc-md-items-type	
Obligation	recommendation
If the key field metadata and the key values of the key fields of the collection can be accessed by a client, then the `itemType` property SHOULD be included in the collection resource to indicate the type of the collection. The value of the `itemType` property SHOULD be `dataset`.	

7.9.2.1.3. Links

To support hypermedia navigation, the links property must be populated with sufficient hyperlinks to navigate through the whole dataset.

REQUIREMENT 16	
/req/core/rc-md-items-links	
Obligation	requirement
A	`200`-response SHALL include the following links in the `links` property of the response: • A link to this response document (relation: `self`) • A link to the response document in every other media type supported by the service (relation: `alternate`) • A link to key fields metadata of this collection (relation: `keys`)

REQUIREMENT 16	
B	All links SHALL include the `rel` and `type` properties.

Additional information may be available to assist in understanding and using this dataset. Links to those resources should be provided as well.

REQUIREMENT 17	
/req/core//rc-md-items-descriptions	
Obligation	requirement
A	If external schemas or descriptions exist that provide additional information about the structure or semantics of the collection, a `200`-response SHOULD include links to each of those resources in the `links` property of the response (relation: `describedby`).
B	The `type` link parameter SHOULD be provided for each link.

7.9.3. Error Situations

See HTTP status codes for general guidance.

If the parameter `collectionId` does not exist on the server, the status code of the response will be `404`. (see Table 5).

7.10. Collection's Key Fields

The HTTP GET operation at path `{root}/collections/{collectionId}/keys` returns a list of key fields of a specific collection.

7.10.1. Request

REQUIREMENT 18	
/req/core/collections-collectionid-keys-op	
Obligation	requirement
A	The server SHALL support the HTTP GET operation at the path `{root}/collections/{collectionId}/keys`.

REQUIREMENT 18	
B	The parameter `collectionId` is each id property in the collections response (JSONPath: `$.collections[*].id`).

7.10.2. Response

REQUIREMENT 19	
/req/core/collections-collectionid-keys-success	
Obligation	requirement
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 based on the Collection's Key Fields Schema.

7.10.2.1. Response Schema for the Collection's Key Fields

```
schema:
  $ref: '#/components/schemas/CollectionKeysResponseObject'

CollectionKeysResponseObject:
  required:
  - keys
  - links
  type: object
  properties:
    links:
      type: array
      items:
        $ref: '#/components/schemas/Link'
    keys:
      type: array
      items:
        $ref: '#/components/schemas/CollectionKeysObject'

CollectionKeysObject:
  required:
  - id
  - isDefault
  - links
  type: object
  properties:
    isDefault:
      type: boolean
    language:
      type: string
    id:
      type: string
    links:
      type: array
      items:
```

```
$ref: '#/components/schemas/Link'
```

Figure 6 — Collection's Key Fields Schema

7.10.2.1.1. isDefault

Information on the Collection's default key field for the data joins. Value `true` indicates the default key field.

REQUIREMENT 20	
/req/core/collection-collectionsid-keys-default-key	
Obligation	requirement
Exactly one object in the response's `keys` array SHALL have the `isDefault` property value `true`.	

7.10.2.1.2. language

Language in which the key field's key values are written (if applicable to the key field). The format of the language field follows the ISO 639-1 language code values.

7.10.2.1.3. id

Identifier of the key field.

7.10.2.1.4. links

To support hypermedia navigation, the `links` property must be populated with sufficient hyperlinks to navigate through the collection's key fields' key values.

Description of the `links` property of the `CollectionKeysResponseObject`:

REQUIREMENT 21	
/req/core/collections-collectionid-keys-links	
Obligation	requirement
A	`200`-response SHALL include the following links in the `links` property of the response: • A link to this response document (relation: `self`). • A link to the response document in every other media type supported by the service (relation: `alternate`).

REQUIREMENT 21	
B	All links SHALL include the `rel` and `type` properties.

Description of `links` property of the `CollectionKeysObject`:

REQUIREMENT 22	
/req/core/collections-collectionid-keys-items-links	
Obligation	requirement
A	`200`-response SHALL include a following link in the `links` property for each key field included in the response's `keys` array: • A link to the key field's key values response document (relation: `key-values`).
B	The links SHALL include the `rel` and `type` properties.

7.10.3. Error Situations

See HTTP status codes for general guidance.

If the parameter `collectionId` does not exist on the server, the status code of the response will be `404`. (see Table 5).

7.11. Collection's Key Field

The HTTP GET operation at path `{root}/collections/{collectionId}/keys/{keyFieldId}` returns a list of key values of a specific key field of a specific collection.

7.11.1. Request

REQUIREMENT 23	
/req/core/collections-collectionid-keys-keyfieldid-get-op	
Obligation	requirement
A	The server SHALL support the HTTP GET operation at the path `{root}/collections/{collectionId}/keys/{keyFieldId}`.
B	The parameter collectionId is each `id` property in the `collections` response (JSONPath: `$.collections[*].id`).

REQUIREMENT 23	
C	The parameter `keyFieldId` is each `id` property in the `collection's key fields` response (JSONPath: `$.keys[*].id`).

D	The server MAY support the query parameter `key` to filter the results by key value. The `key` parameter SHALL possess the following characteristics (using an OpenAPI Specification 3.0 fragment):
	`name: key`
	`in: query`
	`required: false`
	`schema:`
	`  type: string`

E	The server MAY support the query parameter `limit` to limit the number of key values that can be returned in a single response. The `limit` parameter SHALL possess the following characteristics (using an OpenAPI Specification 3.0 fragment):
	`name: limit`
	`in: query`
	`required: false`
	`schema:`
	`  type: integer`
	`  minimum: 1`
	`  maximum: 10000`
	`  default: 1000`
	Note The values for minimum, maximum and default are only examples and MAY be changed.

7.11.2. Response

REQUIREMENT 24	
/req/core/collections-collectionid-keys-keyfieldid-get-success	
Obligation	requirement
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 based on the Collection's Key Field Schema.

7.11.2.1. Parameter Limit

The number of returned key values depends on the server and the value of the `limit` parameter.

The client can request a limit to the number of key values returned in a response by using the `limit` parameter. The `limit` parameter indicates the maximum number of key values which should be included in a single response.

The server may have a default value for the limit and a maximum limit.

REQUIREMENT 25	
/req/core/collections-collectionid-keys-keyfieldid-get-success-limit-response	
Obligation	requirement
A	If the `limit` parameter is provided by the client and supported by the server, then the response SHALL not contain more resources than specified by the `limit` parameter.
B	If the service specifies a maximum value for the `limit` parameter, the response SHALL not contain more resources than this maximum value.

REQUIREMENT 26	
/req/core/collections-collectionid-keys-keyfieldid-get-success-limit-unsupported	
Obligation	requirement
If the `limit` parameter is provided by the client but it is not supported by the server, then the server SHALL process the request as if the parameter had not been provided.	

7.11.2.2. Paged response

If the number of items in the `keys` array of the response is less than or equal to the requested/default/maximum limit then the server will include a link to the next set of results.

PERMISSION 3	
/per/core/collections-collectionid-keys-keyfieldid-get-success-server-limit	
Obligation	permission
If a server is configured with a maximum response size, then the server MAY page responses which exceed that threshold.	

RECOMMENDATION 9	
/rec/core/collections-collectionid-keys-keyfieldid-get-success-server-limit	
Obligation	recommendation

RECOMMENDATION 9
Clients SHOULD be prepared to handle a paged response even if they have not specified a `limit` parameter in their query.

The effect of the `limit` parameter is to divide the response into a number of pages. Each page (except for the last) contains the specified number of entities. The response contains the first page. Additional pages can be accessed through hyperlink navigation.

RECOMMENDATION 10	
/rec/core/collections-collectionid-keys-keyfieldid-get-success-next-1	
Obligation	recommendation
A `200`-response SHOULD include a link to the next "page" (relation: `next`), if more resources have been selected than returned in the response.	

RECOMMENDATION 11	
/rec/core/collections-collectionid-keys-keyfieldid-get-success-next-2	
Obligation	recommendation
Dereferencing a `next` link SHOULD return additional resources from the set of selected resources that have not yet been returned.	

RECOMMENDATION 12	
/rec/core/collections-collectionid-keys-keyfieldid-get-success-next-3	
Obligation	recommendation
The number of resources in a response to a `next` link SHOULD follow the same rules as for the response to the original query and again include a `next` link, if there are more resources in the selection that have not yet been returned.	

Providing `prev` links supports navigating back and forth between pages, but depending on the implementation approach it may be too complex to implement.

PERMISSION 4	
/per/core/collections-collectionid-keys-keyfieldid-get-success-prev	
Obligation	permission
A response to a `next` link MAY include a `prev` link to the resource that included the `next` link.	

If the server response does not contain all of the key values that match the selection parameters, then the client must be informed of that fact.

REQUIREMENT 27	
/req/core/collections-collectionid-keys-keyfieldid-get-success-paged-response	
Obligation	requirement
If the number of key values in the `keys` element is less than the number that match the selection paarameters, then the `numberMatched` and `numberReturned` properties SHALL be included in the response.	

The `numberMatched` property of the response indicates the number of key values that are available in the server that match the selection parameters in the request.

REQUIREMENT 28	
/req/core/collections-collectionid-keys-keyfieldid-get-success-numberMatched	
Obligation	requirement
A	If a property `numberMatched` is included in the response, the value SHALL be identical to the number of hosted key values that meet the selection parameters provided by the client.
B	A server MAY omit this information in a response, if the information about the number of matching resources is not known or difficult to compute.

The number of key values included in a response may be a subset of the number matched. In that case, the `numberReturned` property of the response indicates the number of key values returned in this “page” of the response.

REQUIREMENT 29	
/req/core/collections-collectionid-keys-keyfieldid-get-success-numberReturned	
Obligation	requirement
A	If a property `numberReturned` is included in the response, the value SHALL be identical to the number of items in the `keys` array in the response document.
B	A server MAY omit this information in a response, if the information about the number of resources in the response is not known or difficult to compute.

7.11.2.3. Response Schema for the Collection's Key Field

```
schema:
  $ref: '#/components/schemas/CollectionKeysKeyFieldResponseObject'

CollectionKeysKeyFieldResponseObject:
  required:
  - keys
  - links
  type: object
  properties:
    links:
      type: array
      items:
        $ref: '#/components/schemas/Link'
    keys:
      type: array
      items:
        $ref: '#/components/schemas/KeyObject'
    numberMatched:
      type: integer
    numberReturned:
      type: integer

KeyObject:
  required:
  - key
  type: object
  properties:
    key:
      type: string
    title:
      type: string
```

Figure 9 — Collection's Key Field Schema

7.11.2.3.1. keys

Array of key objects.

7.11.2.3.2. numberMatched

The number of key values that are available in the server that match the selection parameters in the request.

7.11.2.3.3. numberReturned

The number of key values returned in the response.

7.11.2.3.4. key

Key value.

7.11.2.3.5. links

To support hypermedia navigation, the links property must be populated with sufficient hyperlinks.

REQUIREMENT 30	
/req/core/collection-collectionid-keys-keyfieldid-links	
Obligation	requirement
A	`200`-response SHALL include the following links in the links property of the response: • A link to this response document (relation: `self`). • A link to the response document in every other media type supported by the service (relation: `alternate`).
B	All links SHALL include the `rel` and `type` properties.

7.11.2.3.6. title

Human-readable description of the key value.

7.11.3. Error Situations

See HTTP status codes for general guidance.

If the parameter `collectionId` does not exist on the server, the status code of the response will be `404`. (see Table 5).

If the parameter `keyFieldId` does not exist on the server, the status code of the response will be `404`. (see Table 5).

7.12. Joins

The HTTP GET operation at path `{root}/joins` returns a list of joins that are available on the server.

7.12.1. Request

REQUIREMENT 31	
/req/core/joins-get-op	
Obligation	requirement
A	The server SHALL support the HTTP GET operation at the path `{root}/joins`.

B	The server MAY support the query parameter `limit` to limit the number of resources that can be returned in a single response. The `limit` parameter SHALL possess the following characteristics (using an OpenAPI Specification 3.0 fragment):
	`name: limit`
	`in: query`
	`required: false`
	`schema:`
	`  type: integer`
	`  minimum: 1`
	`  maximum: 1000`
	`  default: 10`
	NoteThe values for minimum, maximum and default are only examples and MAY be changed.

C	The server MAY support the query parameter `datetime` to filter the returned joins by their execution timestamp. The `datetime` parameter SHALL possess the following characteristics (using an OpenAPI Specification 3.0 fragment):
	`name: datetime`
	`in: query`
	`required: false`
	`schema:`
	`  type: string`
	Temporal geometries are either a date-time value or a time interval. The parameter value SHALL conform to the following syntax (using ABNF):
	`interval-closed = date-time "/" date-time`
	`interval-open-start = "../" date-time`
	`interval-open-end = date-time "/.."`
	`interval = interval-closed / interval-open-start / interval-open-end`
	`datetime = date-time / interval`
	The syntax of date-time is specified by RFC 3339, 5.6.

REQUIREMENT 31	
	Open ranges in time intervals at the start or end are supported using a double-dot (..) or an empty string for the start/end..

7.12.2. Response

REQUIREMENT 32	
/req/core/joins-get-success	
Obligation	requirement
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 based on the Joins Schema.

7.12.2.1. Parameter Limit

The number of returned joins depends on the server and the value of the `limit` parameter.

The client can request a limit to the number of joins returned in a response by using the `limit` parameter. The `limit` parameter indicates the maximum number of joins which should be included in a single response.

The server may have a default value for the limit and a maximum limit.

REQUIREMENT 33	
/req/core/joins-get-success-limit-response	
Obligation	requirement
A	If the `limit` parameter is provided by the client and supported by the server, then the response SHALL not contain more resources than specified by the `limit` parameter.
B	If the service specifies a maximum value for the `limit` parameter, the response SHALL not contain more resources than this maximum value.

REQUIREMENT 34	
/req/core/joins-get-success-datetime	
Obligation	requirement

REQUIREMENT 34
If the `datetime` parameter is provided by the client and supported by the server, then only joins that have a `timeStamp` property that intersects the temporal information in the `datetime` parameter SHALL be part of the result set.

7.12.2.2. Parameter datetime

REQUIREMENT 35	
/req/core/joins-get-success-limit-unsupported	
Obligation	requirement
If the `limit` parameter is provided by the client but it is not supported by the server, then the server SHALL process the request as if the parameter had not been provided.	

7.12.2.3. Paged response

If the number of items in the `joins` array of the response is less than or equal to the requested/default/maximum limit then the server will include a link to the next set of results.

PERMISSION 5	
/per/core/joins-get-success-server-limit	
Obligation	permission
If a server is configured with a maximum response size, then the server MAY page responses which exceed that threshold.	

RECOMMENDATION 13	
/rec/core/joins-get-success-server-limit	
Obligation	requirement
Clients SHOULD be prepared to handle a paged response even if they have not specified a `limit` parameter in their query.	

The effect of the `limit` parameter is to divide the response into a number of pages. Each page (except for the last) contains the specified number of entities. The response contains the first page. Additional pages can be accessed through hyperlink navigation.

RECOMMENDATION 14

/rec/core/joins-get-success-next-1

Obligation	requirement

A `200`-response SHOULD include a link to the next "page" (relation: `next`), if more resources have been selected than returned in the response.

RECOMMENDATION 15

/rec/core/joins-get-success-next-2

Obligation	requirement

Dereferencing a `next` link SHOULD return additional resources from the set of selected resources that have not yet been returned.

RECOMMENDATION 16

/rec/core/joins-get-success-next-3

Obligation	requirement

The number of resources in a response to a `next` link SHOULD follow the same rules as for the response to the original query and again include a `next` link, if there are more resources in the selection that have not yet been returned.

Providing `prev` links supports navigating back and forth between pages, but depending on the implementation approach it may be too complex to implement.

PERMISSION 6

/per/core/joins-get-success-prev

Obligation	permission

A response to a `next` link MAY include a `prev` link to the resource that included the `next` link.

If the server response does not contain all of the joins that match the selection parameters, then the client must be informed of that fact.

REQUIREMENT 36

/req/core/joins-get-success-paged-response

Obligation	requirement

REQUIREMENT 36
If the number of joins in the `joins` element is less than the number that match the selection paarameters, then the `numberMatched` and `numberReturned` properties SHALL be included in the response.

The `numberMatched` property of the response indicates the number of joins that are available in the server that match the selection parameters in the request.

REQUIREMENT 37	
/req/core/joins-get-success-numberMatched	
Obligation	requirement
A	If a property `numberMatched` is included in the response, the value SHALL be identical to the number of joins that meet the selection parameters provided by the client.
B	A server MAY omit this information in a response, if the information about the number of matching resources is not known or difficult to compute.

The number of joins included in a response may be a subset of the number matched. In that case, the `numberReturned` property of the response indicates the number of joins returned in this “page” of the response.

REQUIREMENT 38	
/req/core/joins-get-success-numberReturned	
Obligation	requirement
A	If a property `numberReturned` is included in the response, the value SHALL be identical to the number of items in the `joins` array in the response document.
B	A server MAY omit this information in a response, if the information about the number of resources in the response is not known or difficult to compute.

7.12.2.4. Response schema for the Joins

```
schema:
  $ref: '#/components/schemas/JoinsResponseObject'

JoinsResponseObject:
  required:
  - joins
  - links
  type: object
```

```
  properties:
    links:
      type: array
      items:
        $ref: '#/components/schemas/Link'
    joins:
      type: array
      items:
        $ref: '#/components/schemas/JoinsObject'
    numberMatched:
      type: integer
    numberReturned:
      type: integer
    timeStamp:
      type: string
      format: date-time

JoinsObject:
  required:
  - id
  - links
  - timeStamp
  type: object
  properties:
    id:
      type: string
    timeStamp:
      type: string
      format: date-time
    links:
      type: array
      items:
        $ref: '#/components/schemas/Link'
```

Figure 13 — Joins Schema

7.12.2.4.1. joins

The `joins` property of the response provides a description of each individual join hosted by the server.

REQUIREMENT 39	
/req/core/joins-get-success-items	
Obligation	requirement
For each join resource accessible through the server, metadata describing that join SHALL be provided in the property `joins`.	

7.12.2.4.2. links

To support hypermedia navigation, the `links` property must be populated with sufficient hyperlinks to navigate through the joins.

Property `links` in the `JoinsResponseObject`:

REQUIREMENT 40	
/req/core/joins-get-success-links	
Obligation	requirement
A	`200`-response SHALL include the following links in the `links` property of the response: • A link to this response document (relation: `self`) • A link to the response document in every other media type supported by the service (relation: `alternate`)
B	All links SHALL include the `rel` and `type` properties.

Property `links` in the `JoinsObject`:

REQUIREMENT 41	
/req/core/joins-get-success-items-links	
Obligation	requirement
A	For each item included in the `joins` array in the response, the `links` property of that item SHALL include a link for each supported encoding to the join resource (relation: `join`).
B	All links SHALL include the `rel` and `type` properties.

7.12.2.4.3. id

Identifier of the join.

7.12.2.4.4. numberReturned

The number of joins that are available in the server that match the selection parameters in the request.

7.12.2.4.5. numberMatched

The number of joins returned in the response.

7.12.2.4.6. timeStamp

Property `timeStamp` in the `JoinsResponseObject`:

REQUIREMENT 42	
/req/core/joins-get-success-timeStamp	
Obligation	requirement
If a property `timeStamp` is included in the response, the value SHALL be set to the time when the response was generated.	

Property `timeStamp` in the `JoinsObject`:

The time when the join was generated.

7.12.3. Error Situations

See HTTP status codes for general guidance.

7.13. Join Creation

The HTTP POST operation at path `{root}/joins` creates a new join.

The operation contains two joining modes:

a) Joining attribute data from an inputted attribute data file with a collection hosted on the server.

b) Joining attribute data from an inputted attribute data file directly with a spatial dataset file provided with the query.

The core module contains support for the CSV format for the attribute data files and for the GeoJSON format for the spatial data files.

The joins are executed via common key values that are available in both datasets.

If the attribute dataset contains additional key values that are not available in the collection or spatial dataset, they will be not included to the joined output. The extension modules may alter this behavior.

7.13.1. Request

REQUIREMENT 43	
/req/core/joins-post-op	
Obligation	requirement
A	The server SHALL support the HTTP POST operation at path `{root}/joins`.
B	The server SHALL support the `join-type` parameter value `hosted`.
C	The server MAY support the `join-type` parameter value `file`.
D	The request is made as a multipart/form-data request. The request SHALL contain the header: • Content-Type: multipart/form-data;
E	If the attribute dataset file is uploaded with the query, it SHALL contain the header • Content-Disposition: form-data; filename="[attribute dataset file's name]"; name="attribute-dataset-file";
F	If the spatial dataset file is uploaded with the query, it SHALL contain the header: • Content-Disposition: form-data; filename="[spatial dataset file's name]"; name="spatial-dataset-file";
G	The input data files SHALL be encoded with the `UTF-8` encoding.
H	The server SHALL support the GeoJSON output format for the joined data in the `join-type` parameter value `hosted`.
I	The server MAY support any other output formats for the joined data in the `join-type` parameter value `hosted`. The parameters of that request are listed in sections L and M.
J	The server MAY support a optional direct GeoJSON response for the joined data for the `join-type` parameter value `hosted`. In this case, the server returns the joined data directly to the client in the GeoJSON format instead of the join response document. The direct GeoJSON output can be requested with

REQUIREMENT 43

the *output-formats* parameter value: `http://www.opengis.net/spec/ogcapi-joins-1/1.0/conf/output/geojson-direct`.

K

If the server supports the `join-type` parameter value `file`, it SHALL support the GeoJSON output format for the joined data. The parameters of that request are listed in sections L and N.

L

The Common Form Data Parameters for the join-type Parameter Values 'hosted' and 'file':

NAME	DESCI	TYPE AND VALUI	REQUIRED
join-type	The type of join.	String, values: {'hosted', 'file'}	Mandatory
execution-type	The join execut type.	String, value: {'synch	Optional[a]
attribute-dataset-format	The format of the attribute dataset.	String	Mandatory[b]
attribute-dataset-file	The attribu datase file (uploa file).	File	Optional[c]
attribute-dataset-url	The attribute dataset URL.	URL	Optional[c]
attribute-dataset-key	The key field in the attribu datase that	String	Mandatory[d]

REQUIREMENT 43

NAME	DESCRIPTION	TYPE AND VALUE	REQUIRED
	contain the key values		
attribute-dataset-data-value-list	Fields in the attribute dataset that contain the attribute values that will be joined.	String, separated by commas	Mandatory[e]
csv-file-delimiter	The delimit charac used in the CSV file.	String	Optional[f]
csv-file-contains-header-row	Information on the existenc of header row in the CSV file.	Boolean, values: {true, false}	Optional[g]

[a] The core module supports only the value 'synchronous', other values may be defined in the extension modules.

[b] The core module contains support for the value: 'csv'.

[c] One of the parameters: *attribute-dataset-file* or *attribute-dataset-url* is mandatory to be used with the

REQUIREMENT 43

NAME	DESCRIPTION	TYPE AND VALUES	REQUIRED
	operation. The *attribute-dataset-file* parameter can be used for uploading an attribute dataset file to the server. The *attribute-dataset-url* parameter can be used for providing the attribute dataset file through URL link. If both parameters are provided in the query, the server SHALL send HTTP exception `400`.		
	[d] The key field in the attribute dataset that contains the key values. For CSV format, this value is the column number that contains the key values (counting starts from 0).		
	[e] For CSV format, the values are the column numbers that contain the attribute values that will be joined (counting starts from 0).		
	[f] The *csv-file-delimiter* parameter is mandatory to be used with the *attribute-dataset-format* parameter value: 'csv'. The parameter is not required for other formats that may be defined in the extension modules.		
	[g] The *csv-file-contains-header-row* parameter is optional to be used with the *attribute-dataset-format* parameter value: 'csv'. The parameter is not required for other formats that may be defined in the extension modules. Values: (*true*, *false*). If parameter is not provided in the request, default value *false* is used. If parameter has value *true*, CSV file's header is assumed to be on the first row of the CSV file and data values are assumed to start from the second row. If parameter has value *false*, the data values are assumed to start from CSV file's first row.		

M

Additional Form Data Parameters for join-type Parameter Value 'hosted':

NAME	DESCRIPTION	TYPE AND VALUES	REQUIRED
output-formats	List of output formats for the joined data	String, separated by commas	Optional[a]

REQUIREMENT 43

NAME	DESCRIPTION	TYPE AND VALUE	REQUIRED
	that will be included to the response document.		
include-join-metadata	Include the join Inform element to the respor docum	Boolea values {true, false}	Optional[b]
collection-id	The value of the `id` attribute of the collection available on the server, to which the attribute data will be joined.	String	Mandatory
collection-key	The value of the `id` attribu of the key field of the	String	Optional[c]

REQUIREMENT 43

NAME	DESCRIPTION	TYPE AND VALUE	REQUIRED
	collection that will be used in the join operation		

[a] Comma-separated list of the outputs that will be included to the response document. The output formats that the server supports SHALL be listed in the server's conformance declaration. If the parameter value is not provided in the request, a default value `http://www.opengis.net/spec/ogcapi-joins-1/1.0/conf/output/geojson` is used. If the server supports the direct geojson response for the join it SHALL support the value `http://www.opengis.net/spec/ogcapi-joins-1/1.0/conf/output/geojson-direct`.

[b] If parameter is not provided in the request, a default value *false* is used. The parameter is not used with the `output-formats` parameter value `http://www.opengis.net/spec/ogcapi-joins-1/1.0/conf/output/geojson-direct`

[c] If the `collection-key` parameter is not provided in the request, a default key field value of the collection will be used in the join operation.

N

Additional Form Data Parameters for join-type Parameter Value 'file':

NAME	DESCRIPTION	TYPE AND VALUE	REQUIRED
spatial-dataset-format	The format of the spatial dataset.	String	Mandatory[a]
spatial-dataset-file	The spatial dataset	File	Optional[b]

REQUIREMENT 43

NAME	DESCRIPTION	TYPE AND VALUE	REQUIRED
	file (upload file)		
spatial-dataset-url	A URL link to the spatial dataset file	URL	Optional[b]
spatial-dataset-key	The path to the key field in the spatial dataset file that contains key values Example 'feature.properties.key'	String	Mandatory

[a] The core module contains support for the format: `http://www.opengis.net/spec/ogcapi-joins-1/1.0/conf/output/geojson`.

[b] One of the parameters: *spatial-dataset-file* or *spatial-dataset-url* is mandatory to be used with the operation. The *spatial-dataset-file* parameter can be used for uploading the spatial dataset file to the server. The *spatial-dataset-url* parameter can be used for providing the spatial dataset file through URL link. If both parameters are provided in the query, the server SHALL send HTTP exception `400`.

7.13.2. Response

REQUIREMENT 44	
/req/core/joins-post-success	
Obligation	requirement
A	A successful execution of the operation for the `join-type` parameter value `hosted` with other `output-format` parameter values other than `http://www.opengis.net/spec/ogcapi-joins-1/1.0/conf/output/geojson-direct` SHALL be reported as a response with a HTTP status code `201`.
B	The content of the response in `A` is based on the Join Schema.
C	A successful execution of the operation for the `join-type` parameter value `hosted` for the output format `http://www.opengis.net/spec/ogcapi-joins-1/1.0/conf/output/geojson-direct` SHALL be reported as a response with a HTTP status code `200`.
D	The content of the response in `C` SHALL be the joined data in the GeoJSON format.
E	A successful execution of the operation for the `join-type` parameter value `file` SHALL be reported as a response with a HTTP status code `200`.
F	The content of the response in `E` SHALL be the joined data in the GeoJSON format.

REQUIREMENT 45	
/req/core/joins-post-attribute-data-file-csv-multiple-keys	
Obligation	requirement
If the attribute data file is in CSV format and it contains multiple rows with the same key value, the value is used in the join operation from the row where it is encountered first.	

REQUIREMENT 46	
/req/core/joins-joinid-post-success-attribute-data-file-csv-attribute-names	
Obligation	requirement

REQUIREMENT 46	
A	If the attribute data file is in CSV format and it contains the header row, the names for the joined attributes SHALL be the values of the joined columns from the header row.
B	If the attribute data file is in CSV format and it doesn't contain a header row, the server SHALL generate the names for the joined attributes.

7.13.2.1. Response schema for the Join Creation

The response shema for queries that create a join resource is expressed with the Join Shema.

For repsonses that produce a GeoJSON output, the response is a GeoJSON file that contains also the joined attributes.

7.13.3. Error Situations

See HTTP status codes for general guidance.

7.14. Join

The HTTP GET operation at path `{root}/joins/{joinId}` returns metadata on a specific join that is available on the server.

7.14.1. Request

REQUIREMENT 47	
/req/core/joins-joinid-get-op	
Obligation	requirement
A	The server SHALL support the HTTP GET operation at the path `/joins/{joinId}`.
B	The parameter `joinId` is each id property in the `joins` response (JSONPath: `$.joins[*].id`).

7.14.2. Response

REQUIREMENT 48	
/req/core/joins-joinid-get-success	
Obligation	requirement
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 based on the Join Schema.

7.14.2.1. Response Schema for the Join

```
schema:
  $ref: '#/components/schemas/JoinResponseObject'

JoinResponseObject:
  required:
  - join
  - links
  type: object
  properties:
    links:
      type: array
      items:
        $ref: '#/components/schemas/Link'
    join:
      $ref: '#/components/schemas/JoinObject'

JoinObject:
  required:
  - id
  - inputs
  - outputs
  - timeStamp
  type: object
  properties:
    id:
      type: string
    timeStamp:
      type: string
      format: date-time
    inputs:
      $ref: '#/components/schemas/JoinInputsObject'
    outputs:
      type: array
      items:
        $ref: '#/components/schemas/Link'
    joinInformation:
      $ref: '#/components/schemas/JoinInformationObject'

JoinInputsObject:
```

```
  required:
  - attributeDataset
  - collection
  type: object
  properties:
    attributeDataset:
      type: string
    collection:
      type: array
      items:
        $ref: '#/components/schemas/Link'

JoinInformationObject:
  type: object
  properties:
    numberOfMatchedCollectionKeys:
      type: integer
    numberOfUnmatchedCollectionKeys:
      type: integer
    numberOfAdditionalAttributeKeys:
      type: integer
    matchedCollectionKeys:
      type: array
      items:
        type: string
    unmatchedCollectionKeys:
      type: array
      items:
        type: string
    additionalAttributeKeys:
      type: array
      items:
        type: string
    duplicateAttributeKeys:
      type: array
      items:
        type: string
    numberOfDuplicateAttributeKeys:
      type: integer
```

Figure 14 — Join Schema

7.14.2.1.1. links

To support hypermedia navigation, the `links` property must be populated with sufficient hyperlinks.

REQUIREMENT 49	
/req/core/joins-joinid-links	
Obligation	requirement
A	A `200`-response SHALL include the following links in the `links` property of the response: • A link to this response document (relation: `self`)

REQUIREMENT 49	
	• links to this document in other supported media types (link rel: `alternate`)
B	All links SHALL include the `rel` and `type` properties.

7.14.2.1.2. join

The metadata on the join.

7.14.2.1.3. id

Unique identifier for the join resource.

7.14.2.1.4. timeStamp

The time when the join was generated.

7.14.2.1.5. inputs

Input datasets that were used in the join operation.

7.14.2.1.6. attributeDataset

Name or URL of the input attribute data file.

7.14.2.1.7. collection

A `link` object that contains information on the collection that was used in the join operation.

REQUIREMENT 50	
/req/core/joins-joinid-inputs-collection	
Obligation	requirement
A	A `200`-response SHALL include the following links in the `links` property of the `collection` property of the response: • A link to the `collection` resource that was used in the join operation in every supported media type (relation: `dataset`).

REQUIREMENT 50	
B	All links SHALL include the `rel` and `type` properties.

7.14.2.1.8. outputs

Links to the produced outputs that contain the joined data.

REQUIREMENT 51	
/req/core/joins-joinid-outputs	
Obligation	requirement
A	A `200`-response SHALL include the following links in the `links` property of the `outputs` property of the response: • A link to each produced output for the data join operation (relation: `output`)
B	All links SHALL include the `rel` and `type` properties.

7.14.2.1.9. joinInformation

Information on the execution of the data join operation

7.14.2.1.10. numberOfMatchedCollectionKeys

The number of collection's key values, to which attribute data was joined successfully.

7.14.2.1.11. numberOfUnmatchedCollectionKeys

The number of collection's key values, to which attribute data couldn´t be joined.

7.14.2.1.12. numberOfAdditionalAttributeKeys

The number of additional key values in the attribute dataset that were not available in the collection.

7.14.2.1.13. matchedCollectionKeys

List of collection's key values that were successfully joined with attribute data

7.14.2.1.14. unmatchedCollectionKeys

List of collection's key values, to which attribute data couldn´t be joined.

7.14.2.1.15. additionalAttributeKeys

List of key values in the attribute data file that were not available in the collection's key field.

7.14.2.1.16. duplicateAttributeKeys

List of duplicate keys in the attribute data file.

7.14.2.1.17. numberOfDuplicateAttributeKeys

The number of keys in the attribute data file that had duplicate entries.

7.14.3. Error Situations

See HTTP status codes for general guidance.

If the parameter `joinId` does not exist on the server, the status code of the response will be `404`. (see Table 5).

7.15. Join Delete

The HTTP DELETE operation at path `{root}/joins/{joinId}` deletes the specific join from the server.

7.15.1. Request

REQUIREMENT 52	
/req/core/joins-joinid-delete-op	
Obligation	requirement
A	The server MAY support the HTTP DELETE operation at the path `{root}/joins/{joinId}`.

REQUIREMENT 52	
B	The parameter `joinId` is the `id` property in the `joins` response (JSONPath: `$.joins.id`).

7.15.2. Response

REQUIREMENT 53	
/req/core/joins-joinid-delete-success	
Obligation	requirement
A	A successful execution of the operation SHALL be reported as a response with a HTTP status code `204`.
B	The body of the response body SHALL be empty.

7.15.2.1. Response schema for the Join Delete

The response of the operation is empty.

7.15.3. Error Situations

See HTTP status codes for general guidance.

If the parameter `joinId` does not exist on the server, the status code of the response will be `404`. (see Table 5).

8

REQUIREMENTS CLASSES FOR ENCODINGS

8 REQUIREMENTS CLASSES FOR ENCODINGS

8.1. Overview

This clause specifies three requirements classes for encodings to be used by the OGC API — Joins implementations.

- HTML
- JSON
- GeoJSON

The support for the JSON encoding class is mandatory and the support for HTML encoding class is recommended. The support for the GeoJSON format for the joined data output is mandatory.

8.2. Requirements 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 document Spatial Data on the Web Best Practices. This standard therefore recommends supporting HTML as an encoding.

REQUIREMENTS CLASS 2	
http://www.opengis.net/ogcapi-joins-1/1.0/req/html	
Obligation	requirement
Target type	Implementation Specification
Dependency	Requirements Class “Core”

REQUIREMENTS CLASS 2	
Dependency	<html5,HTML5>>
Dependency	Schema.org

REQUIREMENT 54	
/req/html/definition	
Obligation	requirement
Every 200-response of an operation of the server SHALL support the media type text/html.	

REQUIREMENT 55	
/req/html/content	
Obligation	requirement
Every 200-response of an operation of the server SHALL support the media type text/html.	

RECOMMENDATION 17	
/rec/html/schema-org	
Obligation	recommendation
A 200-response with the media type `text/html`, SHOULD include Schema.org annotations.	

8.3. Requirements Class "JSON"

JSON is a text syntax that facilitates structured data interchange between programming languages. It is commonly used for Web-based software-to-software interchanges. The support for JSON is mandatory for OGC API — Joins implementations.

REQUIREMENTS CLASS 3	
http://www.opengis.net/ogcapi-joins-1/1.0/req/json	
Obligation	requirement
Target type	Implementation Specification

REQUIREMENTS CLASS 3	
Dependency	Requirements Class “Core”
Dependency	<rfc8259, IETF RFC 8259: The JavaScript Object Notation (JSON) Data Interchange Format>>
Dependency	JSON Schema

REQUIREMENT 56	
/req/json/definition	
Obligation	requirement
Every 200-response of an operation of the server SHALL support the media type application/json.	

REQUIREMENT 57	
/req/json/content	
Obligation	requirement
A	Every 200-response with the media type application/json SHALL include, or link to, a payload encoded according to the JSON Interchange Format.
B	The schema of all responses with the media type application/json SHALL conform with the JSON Schema specified for that resource.

8.4. Requirements Class “GeoJSON”

REQUIREMENTS CLASS 4	
http://www.opengis.net/ogcapi-joins-1/1.0/req/geojson	
Obligation	requirement
Target type	Implementation Specification
Dependency	Requirements Class “Core”
Dependency	IETF RFC 8259: The JavaScript Object Notation (JSON) Data Interchange Format

REQUIREMENTS CLASS 4	
Dependency	The GeoJSON Format

REQUIREMENT 58	
/req/geojson/definition	
Obligation	requirement
The server SHALL support the GeoJSON format for the joined data outputs.	

9

MEDIA TYPES FOR ANY DATA ENCODING(S)

9 MEDIA TYPES FOR ANY DATA ENCODING(S)

JSON media types that would typically be supported by a server that supports JSON are:

- `application/geo+json` for joined data outputs, and
- `application/json` for all resources.

The typical HTML media type for all “web pages” in a server would be `text/html`.

9.1. Joined Data Outputs

The server implementations are required to support the media type `application/geo+json` for the joined data outputs.

The server implementations may support any other media types for the joined data outputs.

9.2. Problem Details Media Types

The media type that would typically be supported by a server that supports the Problem Details response defined in RFC 7807 is `application/problem+json`.

A

ANNEX A (NORMATIVE) ABSTRACT TEST SUITE (NORMATIVE)

A

ANNEX A (NORMATIVE) ABSTRACT TEST SUITE (NORMATIVE)

A.1. Conformance Class “Core”

Table A.1

Conformance Class	
http://www.opengis.net/spec/ogcapi-joins-1/1.0/conf/core	
Target type	Web API
Requirements class	Requirements Class “Core”
Dependency	http://www.opengis.net/spec/ogcapi-common-1/1.0/conf/core
Dependency	http://www.opengis.net/spec/ogcapi-common-1/1.0/conf/landing-page
Dependency	http://www.opengis.net/spec/ogcapi-common-1/1.0/conf/oas30
Dependency	http://www.opengis.net/spec/ogcapi-common-2/1.0/conf/collections
Dependency	http://www.opengis.net/spec/ogcapi-common-2/1.0/conf/simple-query

A.1.1. General Tests

A.1.1.1. HTTP

Table A.2

Abstract Test 1	/ats/core/http
Test Purpose	Validate that the resources advertised through the API can be accessed using the HTTP 1.1 protocol and, where appropriate, TLS.
Requirement	/req/core/http
Test Method	a) All compliance tests shall be configured to use the HTTP 1.1 protocol exclusively. b) For APIs that support HTTPS, all compliance tests shall be configured to use HTTP over TLS (RFC 2818) with their HTTP 1.1 protocol.

A.1.2. Landing Page {root}/

Table A.3

Abstract Test 2	/ats/core/root-op
Test Purpose	Validate that the landing page can be retrieved from the expected location.
Requirement	/req/core/root-op /req/core/root-success
Test Method	a) Issue an HTTP GET request to the URL `{root}/`. b) Validate that the document was returned with a status code `200`. c) Validate the contents of the returned document using test /ats/core/root-success.

Table A.4

Abstract Test 3	/ats/core/root-success
Test Purpose	Validate that a landing page complies with the required structure and contents.
Requirement	/req/core/root-success
Test Method	Validate the landing page for all supported media types using the landing page schema. a) Validate that the landing page includes a `service-desc` and/or `service-doc` link to an API Definition.

	b) Validate that the landing page includes a `http://www.opengis.net/def/rel/ogc/1.0/conformance` link to the conformance class declaration. c) Validate that the landing page includes a `http://www.opengis.net/def/rel/ogc/1.0/data` link to the metadata on collections. d) Validate that the landing page includes a `joins` link to the joins metadata.

A.1.3. API Definiton path {root}/api (link)

Table A.5

Abstract Test 4	**/ats/core/api-definition-op**
Test Purpose	Validate that the API definition document can be retrieved from the expected location.
Requirement	/req/core/api-definition-op /req/core/api-definition-success
Test Method	DO FOR EACH `service-desc` and `service-doc` link on the landing page: a) Issue a HTTP GET request to the link. b) Validate that a document was returned with a status code `200`. c) Validate the contents of the returned document using test /ats/core/api-definition-success DONE

Table A.6

Abstract Test 5	**/ats/core/api-definition-success**
Test Purpose	Validate that the API definition complies with the required structure and contents.
Requirement	/req/core/api-definition-success
Test Method	Validate the API definition document against an appropriate schema document.

A.1.4. Conformance {root}/conformance

Table A.7

Abstract Test 6	**/ats/core/conformance-op**
Test Purpose	Validate that a conformance declaration can be retrieved from the expected locations.
Requirement	/req/core/conformance-op /req/core/conformance-success
Test Method	DO FOR EACH `http://www.opengis.net/def/rel/ogc/1.0/conformance` link on the landing page: a) Issue an HTTP GET request for the link. b) Validate that a document was returned with a status code `200`. c) Validate the contents of the returned document using test /ats/core/conformance-success. DONE a) Issue an HTTP GET request to the path `{root}/conformance`. b) Validate that a document was returned with a status code `200`. c) Validate the contents of the returned document using test /ats/core/conformance-success.

Table A.8

Abstract Test 7	**/ats/core/conformance-success**
Test Purpose	Validate that the conformance declaration response complies with the required structure and contents.
Requirement	/req/core/conformance-success
Test Method	a) Validate the response document against conformance schema. b) Validate that the document lists all OGC API conformance classes that the API implements

A.1.5. Collections {root}/collections

Table A.9

Abstract Test 8	**/ats/core/collections-get-op**
Test Purpose	Validate that the information about collections can be retrieved from the expected location.

Requirement	/req/core/collections-get-op /req/core/collections-get-success
Test Method	a) Issue an HTTP GET request wihout query parameters to the URL `{root}/collections`. b) Validate that a document was returned with a status code `200`. c) Validate the contents of the returned document using test /ats/core/collections-get-success

Table A.10

Abstract Test 9	**/ats/core/collections-get-success**
Test Purpose	Validate that the Collections content complies with the required structure and contents.
Requirement	/req/core/collections-get-success /req/core/collections-get-success-links /req/core/collections-get-success-timestamp /req/core/collections-get-success-items
Test Method	a) Validate that the response document complies with /req/core/collections-get-success-links b) Validate that the response document complies with /req/core/collections-get-success-timestamp c) Validate that the response document complies with /req/core/collections-get-success-items d) Validate the Collections resource for all supported media types using the schema Collections schema.

Table A.11

Abstract Test 10	**/ats/core/collections-get-success-links**
Test Purpose	Validate that the required links are included in the Collections document.
Requirement	/req/core/collections-get-success-links
Test Method	Verify that the response document includes: a) A link to this response document (relation: `self`). b) A link to the response document in every other media type supported by the server (relation: `alternate`). Verify that all links include the `rel` and `type` link parameters.

Table A.12

Abstract Test 11	/ats/core/collections-get-success-timestamp
Test Purpose	If a `timeStamp` property is included in the Collections document, then validate that the value of that property equates to the time that the response was generated.
Requirement	/req/core/collections-get-success-timestamp
Test Method	IF the Collections document contains a `timeStamp` property, THEN a) Get the current date and time. b) Verify that the value of that property is within two minutes of the current date and time.
Note	The two minute threshold was chosen to allow for test script processing time and clock synchronization issues.

Table A.13

Abstract Test 12	/ats/core/collections-get-success-items
Test Purpose	Validate that each collection accessible through the API is described in the Collections document.
Requirement	/req/core/collections-get-success-items
Test Method	a) Verify that the Collections document includes a `collections` property. b) Verify that the `collections` property is an array. c) Verify that there is an entry in the `collections` property for each resource collection accessible through the API. d) Verify that each entry in the `collections` array is valid according to /ats/core/collections-collectionid-get-success-content.

A.1.6. Collection {root}/collections/{collectionId}

Table A.14

Abstract Test 13	/ats/core/collections-collectionid-get-op
Test Purpose	Validate that a collection information can be retrieved from the expected location.
Requirement	/req/core/collections-collectionid-get-op /req/core/collections-collectionid-get-success

Test Method	a) For every Collection describe in the Collections content, issue a HTTP GET request to the URL `{root}/collections/{collectionId}` where {collectionId} is the `id` property of a collection. b) Validate that a Collection was returned with a status code `200`. c) Validate the contents of the returned document using test /ats/core/collections-collectionid-get-success.

Table A.15

Abstract Test 14	**/ats/core/collections-collectionid-get-success**
Test Purpose	Validate that the Collection content complies with the required structure and contents.
Requirement	/req/core/collections-collectionid-get-success
Test Method	a) Validate the structure and content of the response document using /ats/core/collections-collectionid-get-success-content. b) Verify that the content of the response is consistent with the content for this Resource Collection in the `/collections` response. That is, the values for `id`, `title`, `description` and `extent` are identical.

Table A.16

Abstract Test 15	**/ats/core/collections-collectionid-get-success-content**
Test Purpose	Validate that a Collection document complies with the required structure and values.
Requirement	/req/core/collections-collectionid-get-success /req/core/rc-md-items-links /req/core/rc-md-extent
Test Method	FOR a each Collection document, validate: a) That the Collection document includes an `id` property. b) That the Collection document complies with /ats/core/rc-md-items-links. c) That any extent properties included in the Collection document comply with ats/core/rc-md-extent. d) Validate the content of the Collection document for all supported media types using the Collection schema and test /ats/json/content.

Table A.17

Abstract Test 16	**/ats/core/collections-collectionid-rc-md-extent**
Test Purpose	Validate the `extent` property if it is present
Requirement	/req/core/rc-md-extent
Test Method	IF the extent property is present, THEN: a) Verify that the `extent` provides bounding boxes that include all spatial geometries in this resource. b) Verify that the `extent` provides time intervals that include all temporal geometries in this resource.
Note	A temporal extent of `null` indicates an open time interval.

Table A.18

Abstract Test 17	**/ats/core/collections-collectionid-rc-md-items-links**
Test Purpose	Validate that a Collection document includes all required links.
Requirement	/req/core/rc-md-items-links
Test Method	a) Verify that the Collection document includes a `links` property. b) Verify that the `links` property includes an item which referes back to the Collection document (relation: `self`). c) Verify that the `links` property includes an item for each supported encoding of this Collection document and that each of these items includes an `href` to an appropriate resource (relation: `alternate`). d) Verify that all links include the `rel` and `type` link parameters.

A.1.7. Collection Key Fields {root}/collections/{collectionId}/keys

Table A.19

Abstract Test 18	**/ats/core/collections-collectionid-keys-op**
Test Purpose	Validate that the information on Collection's key fields' can be retrieved from the expected location.
Requirement	/req/core/collections-collectionid-keys-op /req/core/collections-collectionid-keys-success

Test Method	a) For a Collection (path `{root}/collections/{collectionId}`), issue an HTTP GET request to the URL `{root}/collections/{collectionId}/keys` where {collectionId} is the `id` property of a Collection. b) Validate that a document was returned with a status code `200`. c) Validate the contents of the returned document using test /ats/core/collections-collectionid-keys-success

Table A.20

Abstract Test 19	**/ats/core/collections-collectionid-keys-success**
Test Purpose	Validate that the response complies with the required structure and contents.
Requirement	/req/core/collections-collectionid-keys-success
Test Method	a) Validate that the response document complies with Collection Key Fields schema. b) Validate that the response document complies with /req/core/collection-collectionsid-keys-default-key c) Validate that the response document complies with /req/core/collection-collectionsid-keys-links d) Validate that the response document complies with /req/core/collections-collectionid-keys-items-links

Table A.21

Abstract Test 20	**/ats/core/collections-collectionid-keys-default-key**
Test Purpose	Validate that the response `isDefault` value of the response.
Requirement	/req/core/collection-collectionsid-keys-default-key
Test Method	Validate that the response's `keys` array contains exatly one object that has the `isDefault` property value `true`.

Table A.22

Abstract Test 21	**/ats/core/collections-collectionid-keys-links**
Test Purpose	Validate that the required links are included in the response document.
Requirement	/req/core/collections-collectionid-keys-links

Test Method	Verify that the response document includes: a) A link to this response document (relation: `self`), b) A link to the response document in every other media type supported by the server (relation: `alternate`). Verify that all links include the `rel` and `type` link parameters.

Table A.23

Abstract Test 22	**/ats/core/collections-collectionid-keys-items-links**
Test Purpose	Validate that a response document includes all required links.
Requirement	/req/core/collections-collectionid-keys-items-links
Test Method	Verify that the each item in the response's `keys` property includes contains: a) A `links` property that contains links to the key values of the key field in all media types supported by the server (relation: `key-values`), b) Verify that all links include the `rel` and `type` link parameters.

A.1.8. Collection Key Field {root}/collections/{collectionId}/keys/{keyFieldId}

Table A.24

Abstract Test 23	**/ats/core/collections-collectionid-keys-keyfield-op**
Test Purpose	Validate that the key values of a Collection's key field can be retrieved from the expected location.
Requirement	/req/core/collections-collectionid-keys-keyfieldid-op.
Test Method	a) For every key field of a Collection described in the Collection's key fields response `{root}/collections/{collectionId}/keys/`, issue an HTTP GET request to the URL `{root}/collections/{collectionId}/keys/{keyFieldId}` where {collectionId} is the `id` property for a Collection and {keyFieldId} is the `id` property of a collection's key field. b) Validate that a document was returned with a status code `200`. c) Validate the contents of the returned document using test: /ats/core/collections-collectionid-keys-keyfieldid-success.

Table A.25

Abstract Test 24	**/ats/core/collections-collection-keys-keyfield-success**
Test Purpose	Validate that the Collection key field's response complies with the required structure and contents.
Requirement	/req/core/collections-collectionid-keys-keyfieldid-get-success
Test Method	a) Validate that the response document complies with Collection Key Field schema Collection Key Field schema. b) Validate that the response document complies with /req/core/collections-collectionid-keys-keyfieldid-get-success-limit-response. c) Validate that the response document complies with /req/core/collection-collectionid-keys-keyfieldid-links.

A.1.9. Joins {root}/joins

Table A.26

Abstract Test 25	**/ats/core/joins-get-op**
Test Purpose	Validate that the information about Joins can be retrieved from the expected location.
Requirement	/req/core/joins-get-op /req/core/joins-get-success
Test Method	a) Issue an HTTP GET request to the URL `{root}/joins`. b) Validate that a document was returned with a status code `200`. c) Validate the contents of the returned document using test /ats/core/joins-get-success

Table A.27

Abstract Test 26	**/ats/core/joins-get-success**
Test Purpose	Validate that the joins content complies with the required structure and contents.
Requirement	/req/core/joins-get-success /req/core/joins-get-success-items /req/core/collections-get-success-timeStamp //req/core/joins-get-success-link /req/core/joins-get-success-items-links

Test Method	Validate that the response document complies with Joins Schema.

A.1.10. Join Creation {root}/joins

Table A.28

Abstract Test 27	**/ats/core/joins-post-op**
Test Purpose	a) Validate that attribute data can be joined from an inputted attribute dataset file with a Collection available on the server from the expected location. b) If the server supports the file joining functionality, validate that the data can be joined from an inputted attribute dataset file directly with an other inputted file from the expected location,
Requirement	/req/core/joins-post-op /req/core/joins-post-success
Test Method	Test method for (1): Execute the abstract test: /ats/data-joining/query Test method for (2): Execute the abstract test: /ats/file-joining/query

A.1.11. Join {root}/joins/{joinId}

Table A.29

Abstract Test 28	**/ats/core/joins-joinid-get-op**
Test Purpose	Validate that the information about a join can be retrieved from the expected location.
Requirement	/req/core/joins-joinid-get-op /req/core/joins-joinid-get-success
Test Method	a) For a list of joins (path `{root}/joins`), issue an HTTP GET request to the URL `{root}/joins/{joinId}` where {join Id} is the `id` property of a join. b) Validate that a document was returned with a status code `200`. c) Validate the contents of the returned document using test /ats/core/joins-joinid-get-success.

Table A.30

Abstract Test 29	/ats/core/joins-joinid-get-success
Test Purpose	Validate that the Join content complies with the required structure and contents.
Requirement	/req/core/joins-joinid-get-success
Test Method	Validate that the response document complies with Join schema.

A.2. Conformance Class “Data Joining”

Table A.31

Conformance Class	
http://www.opengis.net/spec/ogcapi-joins-1/1.0/conf/core/data-joining	
Target type	Web API
Requirements class	Requirements Class “Core”

Table A.32

Abstract Test 30	/ats/data-joining/query
Test Purpose	a) Validate that attribute data can be joined from an attribute dataset file with a Collection available on the server from the expected location. b) If the server supports the direct GeoJSON output functionality, validate that the attribute data can be joined from the expected location from an attribute dataset file with a Collection available on the server with a direct Geo JSON output.
Requirement	/req/core/joins-post-op /req/core/joins-post-success
Test Method	Test method for (1): 1. Issue an HTTP POST request to the URL `{root}/joins` where: - The request contains the header: `Content-Type: multipart/form-data;`

	- The form data parameter `join-type` has the value `hosted`. - The form data parameter `collection-id` has the same value as `id` property of a collection (from query `{root}/collections`). - The request contains either the uploaded attribute dataset file or it is referenced via URL. - Other query parameters are set correctly. 2. Validate that a document was returned with a status code `201`. 3. Validate the contents of the returned document using test /ats/data-joining/success Test method for (2): Execute the abstract test: /ats/geojson-direct/query

Table A.33

Abstract Test 31	**/ats/data-joining/success**
Test Purpose	Validate that the data join response complies with the required structure and contents.
Requirement	/req/core/joins-post-success
Test Method	a) Validate that the response document complies with the Join Schema. b) Validate that the response document contains the joined data in all the requested output formats that are supported by the server.

A.3. Conformance Class “File Joining”

Table A.34

Conformance Class	
http://www.opengis.net/spec/ogcapi-joins-1/1.0/conf/http://www.opengis.net/spec/ogcapi-joins-1/1.0/conf/core/file-joining	
Target type	Web API
Requirements class	Requirements Class “Core”

Table A.35

Abstract Test 32	/ats/file-joining/query
Test Purpose	Validate that the data can be joined from an inputted attribute dataset file directly with a inputted file from the expected location.
Requirement	/req/core/joins-post-op /req/core/joins-post-success
Test Method	a) Issue an HTTP POST request to the URL `{root}/joins` where • The request contains the header: `Content-Type: multipart/form-data;` • The form data parameter `join-type` has the value `file`. • The request contains either the uploaded attribute dataset file or it is referenced via URL. • The request contains either the uploaded other file to which the attributes will be joined or it is referenced via URL. • Other query parameters are set correctly. b) Validate that a document was returned with a status code `200`. c) Validate the contents of the returned document using test /ats/file_joining/response

Table A.36

Abstract Test 33	/ats/file_joining/success
Test Purpose	Verify the response of the file joining functionality
Requirement	/req/core/joins-post-op /req/core/joins-post-success
Test Method	Validate that the response is a valid document in the correct output format and that it contains the attributes that were joined from the attribute dataset file.

A.4. Conformance Class “File Uploading with Query”

Table A.37

Conformance Class

http://www.opengis.net/spec/ogcapi-joins-1/1.0/conf/input/file-upload	
Target type	Web API
Requirements class	Requirements Class “Core”

Table A.38

Abstract Test 34	**/ats/file-upload/file-upload**
Test Purpose	a) Validate that the input attribute file can be uploaded with the query to the server in the data joining functionality. b) Validate that the input attribute file and the other file can be uploaded with the query to the server in the file data joining functionality.
Requirement	/req/core/joins-post-op /req/core/joins-post-success
Test Method	Test method for (1): Execute the abstract test: /ats/data-joining/ query where: • The request contains the form data parameter `attribute-dataset-file` that contains the uploaded file. • The request contains the header `Content-Disposition: form-data; filename="[attribute dataset file's name]"; name="attribute-dataset-file";` Test method for (2): Execute the abstract test: /ats/file-joining/ query where: • The request contains the form data parameter `attribute-dataset-file` that contains the uploaded file. • The request contains the header `Content-Disposition: form-data; filename="[attribute dataset file's name]"; name="attribute-dataset-file;` • The request contains the form data parameter `spatial-dataset-file` that contains the uploaded file. • The request contains the header `Content-Disposition: form-data; filename="[spatial dataset file's name]"; name="spatial-dataset-file;`

A.5. Conformance Class “File referencing with URL”

Table A.39

Conformance Class	
http://www.opengis.net/spec/ogcapi-joins-1/1.0/conf/input/http-ref	
Target type	Web API
Requirements class	Requirements Class “Core”

Table A.40

Abstract Test 35	/ats/file-referencing/file-referencing
Test Purpose	a) Validate that the input attribute file can be referenced via URL. b) Validate that the input attribute file and the other file can be referenced via URL
Requirement	/req/core/joins-post-op /req/core/joins-post-success
Test Method	Test method for (1): Execute the abstract test: /ats/data-joining/ query where: • The request contains the form data parameter `attribute-dataset-url` that contains the URL of the attribute dataset file. Test method for (2): Execute the abstract test: /ats/file-joining/ query where: • The request contains the form data parameter `attribute-dataset-url` that contains the URL of the attribute dataset file. • The request contains the form data parameter `spatial-dataset-url` that contains the URL of the other dataset file.

A.6. Conformance Class “Join Delete”

Table A.41

Conformance Class
http://www.opengis.net/spec/ogcapi-joins-1/1.0/conf/core/join-delete

Target type	Web API
Requirements class	Requirements Class “Core”

A.6.1. Join Delete {root}/joins/{joinId}

Table A.42

Abstract Test 36	/ats/core/joins-joinid-delete-op
Test Purpose	Validate that the join can be deleted from the expected location.
Requirement	/req/core/joins-joinid-delete-op
Test Method	a) Issue an HTTP DELETE request to the URL `{root}/joins/{joinId}` where {joinId} is the `id` property of a join (from query `{root}/joins/{joinId}`). b) Validate that a document was returned with a status code `204`. c) Validate the contents of the returned document using test /ats/core/joins-joinid-delete-success

Table A.43

Abstract Test 37	/ats/core/joins-joinid-delete-success
Test Purpose	Validate that the join was deleted from the server.
Requirement	/req/core/joins-joinid-delete-success
Test Method	a) Validate that the join has been deleted from the server by issuing an HTTP GET request to the URL `{root}/joins/{joinId}` where {joinId} is the same `id` property of the join that was used in the delete request. b) Validate that the server sent a response code `404`.

A.7. Conformance Class “HTML”

Table A.44

Conformance Class	
http://www.opengis.net/spec/ogcapi-joins-1/1.0/conf/html	
Target type	Web API
Requirements class	Requirements Class “HTML”

A.7.1. HTML Definition

Table A.45

Abstract Test 38	/ats/html/definition
Test Purpose	Verify support for HTML
Requirement	/req/html/definition
Test Method	Verify that every `200`-response of every operation of the API where HTML was requested is of media type `text/html`.

A.7.2. HTML Content

Table A.46

Abstract Test 39	/ats/html/content
Test Purpose	Verify the content of an HTML document given an input document and schema.
Requirement	/req/html/content
Test Method	a) Validate that the document is an HTML 5 document. b) Manually inspect the document against the schema.

A.8. Conformance Class “JSON”

Table A.47

Conformance Class	
http://www.opengis.net/spec/ogcapi-joins-1/1.0/conf/json	
Target type	Web API
Requirements class	Requirements Class “JSON”

A.8.1. JSON Definition

Table A.48

Abstract Test 40	/ats/json/definition
Test Purpose	Verify support for JSON.
Requirement	/req/json/definition /req/json/content
Test Method	DO FOR EACH resource and operation defined in the Core Conformance Class: 1. Execute the operation specifing `application/json` as the media type. 2. Validate that a document was returned with a status code `200`. 3. Validate the contents of the returned document using test / conf/json/content. DONE.

A.8.2. JSON Content

Table A.49

Abstract Test 41	/ats/json/content
Test Purpose	Verify the content of a JSON document given an input document and schema.
Requirement	/req/json/content
Test Method	a) Validate that the document is a JSON (IETF RFC 8259) document.

	b) Validate the document against the schema using a JSON Schema validator.

A.9. Conformance Class “GeoJSON Output for Joined Data”

Table A.50

Conformance Class	
http://www.opengis.net/spec/ogcapi-joins-1/1.0/conf/output/geojson	
Target type	Web API
Requirements class	Requirements Class “GeoJSON output for joined data”

A.9.1. GeoJSON Definition

Table A.51

Abstract Test 42	**/ats/geojson/definition**
Test Purpose	Verify support for GeoJSON ouput for joined data.
Requirement	/req/geojson/definition
Test Method	a) Execute the operation Join Creation, defined in the Core Requirements Class with the `join-type` parameter value `hosted` and `outputFormats` parameter value `http://www.opengis.net/spec/ogcapi-joins-1/1.0/conf/output/geojson`. b) Validate that a document was returned with a status code `201`. c) Validate that the contents of the returned document using test /conf/geojson/content.

A.9.2. GeoJSON Content

Table A.52

Abstract Test 43	/ats/geojson/content
Test Purpose	Verify the content of a GeoJSON document given an input document and schema.
Requirement	/req/json/content
Test Method	a) Execute the link that contains the joined data output in the GeoJSON format. b) Validate that the response is a GeoJSON document. c) Validate the document against the schema using an JSON Schema validator.

A.10. Conformance Class “Direct GeoJSON output for joined data”

Table A.53

Conformance Class	
http://www.opengis.net/spec/ogcapi-joins-1/1.0/conf/output/geojson-direct	
Target type	Web API
Requirements class	Requirements Class “Direct GeoJSON output for joined data”

A.10.1. Direct GeoJSON Definition

Table A.54

Abstract Test 44	/ats/geojson-direct/query
Test Purpose	Validate that the data join response complies with the required structure and contents.
Requirement	/req/core/joins-post-op /req/core/joins-post-success

Test Method	a) Issue an HTTP POST request to the URL `{root}/joins` where • The request contains the header: `Content-Type: multipart/form-data;` • The form data parameter `join-type` has value `hosted` • The request contains either the uploaded attribute dataset file or it is referenced via URL. • The form data parameter `collection-id` has the same value as `id` property of a collection (from query `{root}/collections`). • The form data parameter `outputFormats` has value `http://www.opengis.net/spec/ogcapi-joins-1/1.0/conf/output/geojson-direct`. • Other query parameters are set correctly. b) Validate that a document was returned with a status code `200`. c) Validate that the content of the returned document using test /ats/geojson-direct/success.

A.10.2. Direct GeoJSON Content

Table A.55

Abstract Test 45	**/ats/geojson-direct/success**
Test Purpose	Validate that the data join response complies with the required structure and contents.
Requirement	/req/core/joins-post-success
Test Method	Validate that the response is a valid GeoJSON document and that it contains the attributes that were joined from the attribute dataset file.

B

ANNEX B (NORMATIVE) REVISION HISTORY

B

ANNEX B (NORMATIVE) REVISION HISTORY

Table B.1

DATE	RELEASE	EDITOR	PRIMARY CLAUSES MODIFIED	DESCRIPTION
2021-12-17	1.0.0-SNAPSHOT	P. Latvala	all	Changed document into metanorma template, edited the whole document, removed join update operation
2021-11-30	1.0.0-SNAPSHOT	P. Latvala	all	Updated chapters 2 and 7
2021-11-24	1.0.0-SNAPSHOT	P. Latvala	all	Initial version

BIBLIOGRAPHY

BIBLIOGRAPHY

1. IANA: Link Relation Types, https://www.iana.org/assignments/link-relations/link-relations.xml

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