OGC API - TJS - Part 1: Core

Open Geospatial Consortium

Submission Date: <yyyy-mm-dd>

Approval Date: <yyyy-mm-dd>

Publication Date: <yyyy-mm-dd>

External identifier of this OGC® document: http://www.opengis.net/doc/IS/xx/xx

Internal reference number of this OGC® document: YY-xxx

Version: 1.0.0-SNAPSHOT (Editor's draft)

Latest Published Draft: n/a

Category: OGC® Implementation Specification

Editors: Pekka Latvala

OGC API - TJS - Part 1: Core

Document type: OGC® Standard

Document subtype: Interface

Document stage: Draft

Document language: English

License Agreement

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

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

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

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

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

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

Table of Contents

i. Abstract

This document is the specification for the core module of the OGC API - TJS standard. The core module specifies a service interface that allows non-spatial attribute data to be joined with spatial datasets via common identifiers that are available in both datasets. The TJS core module supports also operations for viewing metadata on spatial datasets and their keys that are available on the server, operations for accessing, updating and deleting the created joins and operation for joining attribute data directly with inputted spatial data files.

CAUTION	This is a DRAFT version of the OGC API - TJS standard. This draft is not complete and there are open issues that are still under discussion.

ii. Keywords

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

ogcdoc, OGC document, standard, TJS, API, openapi

iii. Preface

This standard is the result of the work that was executed to renew the OGC implementation standard: OpenGIS® Georeferenced Table Joining Service (TJS) (document nr. 10-070r2), specified in 2010.

This document defines the core module of the OGC API - TJS 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 Inc. shall not be held responsible for identifying any or all such patent rights.

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

iv. Submitting organizations

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

- National Land Survey of Finland

v. Submitters

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

Name	**Affiliation**
Pekka Latvala *(editor)*	National Land Survey of Finland

1. Scope

This OpenGIS® standard defines the core module for the OGC API - TJS specification. The core module specifies a

RESTful service interface that contains 3 operation sets: *discovery operations*, *data joining operations* and *file joining operations*.

The operation set *discovery operations* contains operations for obtaining general information on the TJS implementation. It includes operations for accessing the API landing page, the API definition file and information on the service's conformance to the TJS standard.

The operation set *data joining operations* contains functionalities for retrieving metadata and key values on the spatial datasets that are available on the server, joining attribute data from csv files with these spatial datasets and accessing, updating and deleting the created joins.

The operation set *file joining operations* can be used for joining attribute data directly with inputted spatial data files. The core module contains file joining support between GeoJSON spatial data files and csv attribute data files.

The core module doesn't contain any functionalities for inserting, updating or deleting the spatial datasets on the server. These functionalities can be defined in potential extension modules. The support for other attribute data formats than csv in the *data joining operations* and *file joining operations*, together with the support for other spatial data formats than GeoJSON in the *file joining operations* can be also defined in potential extension modules.

---

2. Conformance

This standard defines 1 requirement class: "core".

Requirements for 1 standardization target types are considered:

- Web services

This standard defines three operations classes *discovery operations*, *data joining operations* and *file joining operations*.

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. References

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

- Internet Engineering Task Force (IETF). RFC 2616, **Hypertext Transfer Protocol — HTTP/1.1** [online]. Edited by R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach and T. Berners-Lee. 1999 [viewed 2020-04-06]. Available at: https://tools.ietf.org/html/rfc2616
- Internet Engineering Task Force (IETF). RFC 3986, **Uniform Resource Identifier (URI): Generic Syntax** [online]. Edited by T. Berners-Lee, R. Fielding and L. Masinter. 2005 [viewed 2020-04-06]. Available at: https://tools.ietf.org/html/rfc3986
- Internet Engineering Task Force (IETF). RFC 6266 **Use of the Content-Disposition Header Field in the Hypertext Transfer Protocol (HTTP)** [online]. Edited by J. Reschke. 2011 [viewed 2020-04-06]. Available at: https://tools.ietf.org/html/rfc6266

- Internet Engineering Task Force (IETF). RFC 7578 **Returning Values from Forms: multipart/form-data** [online]. Edited by L. Masinter. 2015 [viewed 2020-04-06]. Available at: https://tools.ietf.org/html/rfc7578
- Internet Engineering Task Force (IETF). RFC 7946 **The GeoJSON Format** [online]. Edited by H. Butler, M. Daly, A. Doyle, S. Gillies, S. Hagen, and T. Schaub. 2016 [viewed 2020-04-06]. Available at: https://tools.ietf.org/html/rfc7946
- Internet Engineering Task Force (IETF). RFC 8288 **Web Linking** [online]. Edited by M. Nottingham. 2017 [viewed 2020-04-06]. Available at: http://tools.ietf.org/rfc/rfc8288
- Open API Initiative (OAI): **The OpenAPI specification 3.0** [online]. 2020 [viewed 2020-04-06]. Available at: https://github.com/OAI/OpenAPI-Specification/blob/master/versions/

---

4. Terms and Definitions

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 and background

5.1. Identifiers

The normative provisions in this specification are denoted by the URI `http://www.opengis.net/spec/tjs/2.0`

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

5.2. Link relations

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

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

- **alternate**: Refers to a substitute for this context.
- **self**: Conveys an identifier for the link’s context.
- **service-desc**: Identifies service description for the context that is primarily intended for consumption by machines.
  - API definitions are considered service descriptions.
- **service-doc**: Identifies service documentation for the context that is primarily intended for human consumption.

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

- **conformance**: Refers to a resource that identifies the specifications that the link's context conforms to.
- **spatialdatasets**: Refers to a resource that is comprised of the metadata of the spatialdatasets that are available on the server.
- **keys**: Refers to a resource that is comprised of the metadata of the key fields that belong to a specific spatial dataset that is available on the server.
- **keyname**: Refers to a resource that is comprised of the metadata of the key values that belong to a specific key field of a specific spatial dataset that is available on the server.
- **key**: Refers to a resource that is comprised of a specific key value of a specific key field of a specific spatial dataset that is available on the server.
- **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.

5.2.1. Response Schema for the Link Object

```
Link:
  required:
  - href
  type: object
  properties:
    href:
      type: string
    rel:
      type: string
    type:
      type: string
    hreflang:
      type: string
    title:
      type: string
```

5.3. Exception messages

The exception messages have the following structure:

Name	Description
status	The HTTP status code of the response.
message	Details of the exception.
locator	Name of the request parameter that caused the exception

5.3.1. Response Schema for the Exception Messages

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

ExceptionMessage:
  required:
  - message
  - status
  type: object
  properties:
    status:
      type: string
    message:
      type: string
    locator:
      type: string
```

6. Requirements Class "Core"

6.1. Overview

The requirement class “core” contains 3 operation sets: *discovery operations*, *data joining operations* and *file joining operations*. The Table 1 contains an overview of the operations specified in the core module.

The operation set *discovery operations* contains functionalities for accessing the API landing page, the API definition file and the information on the service’s conformance to the specification. The TJS implementations SHALL support all operations in the operation set *discovery operations*.

The TJS implementations SHALL support at least one of the operation sets: *data joining operations* and *file joining operations*. If a server supports a particular operation set it SHALL implement all mandatory operations that belong to it.

The operation set *data joining operations* contains functionalities for accessing metadata and key values on the spatial datasets that are available on the server, joining csv files with the spatial datasets and accessing, updating and deleting the joins on the server.

The operation set *file joining operations* contains a functionality for joining attribute data from an inputted csv file to an inputted GeoJSON file.

Table 1. Overview of the operations in the OGC API - TJS core module

Path	HTTP method	Description
Discovery operations		
/	GET	API landing page
/api	GET	API definition document
/conformance	GET	API conformance declaration
Data joining operations		
/spatialdatasets	GET	Returns metadata on all spatial datasets available on the server
/spatialdatasets/{spatialdatasetid}	GET	Returns metadata on a specific spatial dataset
/spatialdatasets/{spatialdatasetid}/keys	GET	Returns a list of available key fields of a specific spatial dataset
/spatialdatasets/{spatialdatasetid}/keys/{keyname}	GET	Returns the key values of a specific key field of a specific spatial dataset
/spatialdatasets/{spatialdatasetid}/keys/{keyname}/{key}	GET	Returns a specific key value of a specific key field of a specific spatial dataset
/joindata/{spatialdatasetid}/csv	POST	Creates a new join by joining attribute data from a csv file with a specific spatial dataset
/joins	GET	Returns a list of all joins available on the

Path	HTTP method	Description
/joins/{joinid}	GET	Returns metadata on a specific join
/joins/{joinid}/csv	POST	Updates fully a specific join with a data from a csv file
/joins/{joinid}	DELETE	Deletes a specific join
File joining operations		
/joinfiles/geojson/csv	POST	Joins attribute data from a csv file with a GeoJSON file

server.

6.2. Operation Set: Discovery Operations

The operation set *discovery operations* contains operations that provide general information on the TJS implementation. The TJS implementations SHALL support all operations in this operation set.

6.2.1. API Landing Page

The HTTP GET operation at service root path `/` returns the API landing page document. The API landing page document contains links to the API definition document, conformance information and to the metadata on the spatial datasets and joins that are available on the server.

6.2.1.1. Request

Requirement 1	/req/core/root-op
A	The server SHALL support the HTTP GET operation at the path `/`.

6.2.1.2. Response

Requirement 2	/req/core/root-success
A	A successful execution of the operation SHALL be reported as a response with a HTTP status code `200`.
B	The server implementations SHALL support the JSON output format. Any other output formats MAY also be supported.
C	The response is an API landing page document that is based on the landing page schema. The response document SHALL contain links to the following resources: • `self` (link rel property value: 'self') • `/api` (link rel property value: 'service-desc' or 'service-doc') • `/conformance` (link rel property value: 'conformance') If the TJS implementation supports the *data joining operations* operation set, the landing page SHALL contain also links to the following resources: • `/spatialdatasets` (link rel property value: 'spatialdatasets')

	• `/joins` (link rel property value: 'joins') The links SHALL contain the property values 'href', 'rel', 'title' and 'type'.

6.2.1.3. Errors

Requirement 3	/req/core/root-error
A	If there is an error in the server during the processing of the request, it SHALL be reported as a response with a HTTP status code `500`. The response body SHALL contain an exception report message in the JSON format that is based on exception message schema. The value of the message property SHALL be 'Internal server error'.

6.2.1.4. Schema for the Landing Page

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

LandingPageResponseObject:
  required:
  - links
  type: object
  properties:
    title:
      type: string
    description:
      type: string
    links:
      type: array
      items:
        $ref: '#/components/schemas/Link'
```

6.2.2. API Definition

The HTTP GET operation at path `/api` returns the service's API definition document.

6.2.2.1. Request

Requirement 4	/req/core/api-definition-op
A	The URIs of all API definitions referenced from the landing page SHALL support the HTTP GET method.

6.2.2.2. Response

Requirement 5	/req/core/api-definition-success
A	A successful execution of the operation SHALL be reported as a response with a HTTP status code `200`. The server SHALL return an API definition document. The recommended format is OpenAPI version 3.0.

6.2.2.3. Errors

Requirement 6	/req/core/api-definition-error
A	If there is an error in the server during the processing of the request, it SHALL be

Requirement 6	/req/core/api-definition-error
	reported as a response with a HTTP status code `500`. The response body SHALL contain an exception report message in the JSON format that is based on exception message schema. The value of the message property SHALL be 'Internal server error'.

6.2.3. Declaration of Conformance Classes

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

The conformance class values, defined in the core module are:

- Core
  - http://www.opengis.net/spec/tjs/2.0/conf/core
- Data joining from CSV files
  - http://www.opengis.net/spec/tjs/2.0/conf/core/data-joining-csv
- Join updating from CSV files:
  - http://www.opengis.net/spec/tjs/2.0/conf/core/data-joining-csv-update
- Join deleting
  - http://www.opengis.net/spec/tjs/2.0/conf/core/data-joining-delete
- File joining (GeoJSON and CSV)
  - http://www.opengis.net/spec/tjs/2.0/conf/core/file-joining-geojson-csv

6.2.3.1. Request

Requirement 7	/req/core/conformance-op
A	The server SHALL support the HTTP GET operation at the path `/conformance`.

6.2.3.2. Response

Requirement 8	/req/core/conformance-success
A	A successful execution of the operation SHALL be reported as a response with a HTTP status code `200`.
B	The server implementations SHALL support the JSON output format. Any other output formats MAY also be supported.
C	The response contains a list of conformance classes that the service supports. The response document is based on the conformance schema.

6.2.3.3. Errors

Requirement 9	/req/core/conformance-error
A	If there is an error in the server during the processing of the request, it SHALL be reported as a response with a HTTP status code `500`. The response body SHALL contain an exception report message in the JSON

Requirement 9	format that is based on exception message schema. The value of the message property SHALL be 'Internal server error'. **/req/core/conformance-error**

6.2.3.4. Response schema for the Conformance

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

ConformanceResponseObject:
  required:
  - conformsTo
  type: object
  properties:
    conformsTo:
      type: array
    items:
      type: string
```

6.3. Operation Set: Data Joining Operations

The operation set *data joining operations* contains operations for:

- Retrieving metadata and key values of the spatial datasets that are available on the server
- Joining attribute data from CSV files with the spatial datasets that are available on the server
- Accessing, updating and deleting the created joins.

The data joins are executed through common keys that are shared between the spatial dataset and the attribute dataset.

6.3.1. Spatial Datasets

The HTTP GET operation at path `/spatialdatasets` returns metadata on all spatial datasets that are available on the server.

6.3.1.1. Request

Requirement 10	**/req/core/spatialdatasets-op**
A	If the server implements the *data joining operations* operation set it SHALL support the HTTP GET operation at path `/spatialdatasets`. The server implementations SHALL support the following query parameters: **Query parameters:**

Name	Type	Description
organization	String	Filters spatial datasets by organization
startDate	String	Filters spatial datasets by start date. Format: yyyy-mm-dd
endDate	String	Filters spatial datasets by end date. Format: yyyy-mm-dd

6.3.1.2. Response

<table>
<tr><th>Requirement 11</th><th>/req/core/spatialdatasets-success</th></tr>
<tr><td>A</td><td>A successful execution of the operation SHALL be reported as a response with a HTTP status code <code>200</code>.</td></tr>
<tr><td>B</td><td>The server implementations SHALL support the JSON output format. Any other output formats MAY also be supported.</td></tr>
<tr><td>C</td><td>The response document is based on the spatialdatasets schema.

The response document SHALL include the following links:

• A link to this document (link rel: 'self')
• Links to this document in other supported media types (link rel: 'alternate')

The links SHALL contain the parameters href, rel, title and type.

Description of properties of the spatialDatasetObject property:

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>date</td><td>Date that applies to the spatial dataset. Format: yyyy-mm-dd</td></tr>
<tr><td>description</td><td>Description of the spatial dataset</td></tr>
<tr><td>documentation</td><td>Link to the spatial dataset´s documentation</td></tr>
<tr><td>links</td><td>Links to this spatial dataset´s different representations. The links object SHALL include the properties href, rel and type. The value of the rel property SHALL be 'spatialdataset'</td></tr>
<tr><td>organization</td><td>Name of the organization that has produced the spatial dataset</td></tr>
<tr><td>spatialDatasetId</td><td>Unique identifier for the spatial dataset. The identifier is used in other operations for indicating the spatial dataset in question</td></tr>
<tr><td>spatialDatasetURI</td><td>URI that identifies uniquely the spatial dataset</td></tr>
<tr><td>title</td><td>Title of the spatial dataset</td></tr>
</table>
</td></tr>
</table>

6.3.1.3. Errors

Requirement 12	/req/core/spatialdatasets-error
A	If an incorrect request is made to the server, it SHALL be reported as a response with a HTTP status code `400`. The response body SHALL contain an exception report message in the JSON

Requirement 12	format that is based on exception message schema. The value of the message property SHALL be 'InvalidParameterValue'. The locator property SHALL contain the name of the request parameter that caused the exception.
	/req/core/spatialdatasets-error
B	If spatial datasets are not found on the server, it SHALL be reported as a response with a HTTP status code `404`. The response body SHALL contain an exception report message in the JSON format that is based on exception message schema. The value of the message property SHALL be 'Spatial datasets not found'.
C	If there is an error in the server during the processing of the request, it shall be reported as a response with a HTTP status code `500`. The response body SHALL contain an exception report message in the JSON format that is based on exception message schema. The value of the message property SHALL be 'Internal server error'.

6.3.1.4. Response Schema for the Spatial Datasets

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

SpatialDatasetsResponseObject:
  required:
  - links
  - spatialDatasets
  type: object
  properties:
    links:
      type: array
      items:
        $ref: '#/components/schemas/Link'
    spatialDatasets:
      type: array
      items:
        $ref: '#/components/schemas/SpatialDatasetsObject'

SpatialDatasetsObject:
  required:
  - date
  - description
  - links
  - organization
  - spatialDatasetId
  - spatialDatasetURI
  - title
  type: object
  properties:
    date:
      type: string
    description:
      type: string
    documentation:
      type: string
    links:
      type: array
        items:
        $ref: '#/components/schemas/Link'
    organization:
      type: string
    spatialDatasetId:
      type: integer
      format: int32
    spatialDatasetURI:
      type: string
    title:
      type: string
```

6.3.2. Spatial Dataset

The HTTP GET operation at path `/spatialdatasets/{spatialdatasetid}` returns metadata on a specific spatial dataset

available on the server.

6.3.2.1. Request

Requirement 13	**/req/core/spatialdatasets-spatialdatasetid-op**
A	If the server implements the *data joining operations* operation set it SHALL support the HTTP GET operation at path `/spatialdatasets/{spatialdatasetid}`.

6.3.2.2. Response

Requirement 14	**/req/core/spatialdatasets-spatialdatasetid-success**
A	A successful execution of the operation shall be reported as a response with a HTTP status code `200`.
B	The server implementations SHALL support the JSON output format. Any other output formats MAY also be supported.
C	The response document is based on the spatialdataset schema. The response document SHALL include the following links: • A link to this document (link rel: 'self') • Links to this document in other supported media types (link rel: 'alternate') The links SHALL contain the parameters href, rel, title and type. The response property: spatialDataset SHALL contain metadata on the requested spatial dataset. **Description of properties of the spatialDataset property:**

Name	**Description**
date	Date that applies to the spatial dataset. Format: yyyy-mm-dd
description	Description of the spatial dataset
documentation	Link to the spatial dataset´s documentation
keys	Links to the different representations of this spatial dataset´s key fields. The links SHALL include the properties href, rel and type. The value of the rel property SHALL be 'keys'.
organization	Name of the organization that has produced the spatial dataset
spatialDatasetId	Unique identifier for the spatial dataset
spatialDatasetURI	URI that identifies uniquely the spatial dataset

Name	Description
title	Title of the spatial dataset

6.3.2.3. Errors

Requirement 15	/req/core/spatialdatasets-spatialdatasetid-error
A	If an incorrect request is made to the server, it SHALL be reported as a response with a HTTP status code `400`. The response body SHALL contain an exception report message in the JSON format that is based on exception message schema. The value of the message property SHALL be 'InvalidParameterValue'. The locator property SHALL contain the name of the request parameter that caused the exception.
B	If the spatial dataset is not found on the server, it SHALL be reported as a response with a HTTP status code `404`. The response body SHALL contain an exception report message in the JSON format that is based on exception message schema. The value of the message property SHALL be 'Spatial dataset {spatialdatasetid} not found'.
C	If there is an error in the server during the processing of the request, it SHALL be reported as a response with a HTTP status code `500`. The response body SHALL contain an exception report message in the JSON format that is based on exception message schema. The value of the message property SHALL be 'Internal server error'.

6.3.2.4. Response Schema for the Spatial Dataset

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

SpatialDatasetResponseObject:
  required:
  - links
  type: object
  properties:
    links:
      type: array
      items:
        $ref: '#/components/schemas/Link'
    spatialDataset:
      $ref: '#/components/schemas/SpatialDatasetObject'

SpatialDatasetObject:
  required:
  - date
  - description
  - keys
  - organization
  - spatialDatasetId
  - spatialDatasetURI
  - title
  type: object
  properties:
    date:
      type: string
    description:
      type: string
    documentation:
      type: string
    keys:
      type: array
      items:
        $ref: '#/components/schemas/Link'
    organization:
      type: string
    spatialDatasetId:
      type: integer
      format: int32
    spatialDatasetURI:
      type: string
    title:
      type: string
```

6.3.3. Spatial Dataset Key Fields

The HTTP GET operation at path `/spatialdatasets/{spatialdatasetid}/keys` returns a list of key fields that belong to a specific spatial dataset.

6.3.3.1. Request

Requirement 16	**/req/core/spatialdatasets-spatialdatasetid-keys-op**
A	If the server implements the *data joining operations* operation set, it SHALL support the HTTP GET operation at the path `/spatialdatasets/{spatialdatasetid}/keys`.

6.3.3.2. Response

Requirement 17	**/req/core/spatialdatasets-spatialdatasetid-keys-success**
A	A successful execution of the operation SHALL be reported as a response with a HTTP status code `200`.
B	The server implementations SHALL support the JSON output format. Any other output formats MAY also be supported.
C	The response document is based on the spatialdataset keys schema.

The response document SHALL include the following links:

- A link to this document (link rel: 'self')
- Links to this document in other supported media types (link rel: 'alternate')

The links SHALL contain the parameters href, rel, title and type.

Description of properties in the response´s keys property:

Name	Description
isDefault	Indicates if the key is used as a default key field in the data joins with this spatial dataset. Only one object in the response SHALL have the value true
keyName	Name of the key field
links	Links to the different representations of this key value. The links SHALL have the properties href, rel and type. The value of the rel property SHALL be 'keyname'

6.3.3.3. Errors

Requirement 18	/req/core/spatialdataset-spatialdatasetid-keys-error
A	If an incorrect request is made to the server, it SHALL be reported as a response with a HTTP status code `400`. The response body SHALL contain an exception report message in the JSON format that is based on exception message schema. The value of the message property SHALL be 'InvalidParameterValue'. The locator property SHALL contain the name of the request parameter that caused the exception.
B	If the spatial dataset keys are not found on the server, it SHALL be reported as a response with a HTTP status code `404`. The response body SHALL contain an exception report message in the JSON format that is based on exception message schema. The value of the message property SHALL be 'Spatial dataset {spatialdatasetid} keys not found'.
B	If there is an error in the server during the processing of the request, it SHALL be reported as a response with a HTTP status code `500`. The response body SHALL contain an exception report message in the JSON format that is based on exception message schema. The value of the message property SHALL be 'Internal server error'.

6.3.3.4. Response Schema for the Spatial Dataset Key Fields

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

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

SpatialDatasetKeysObject:
  required:
  - isDefault
  - keyName
  - links
  type: object
  properties:
    keyName:
      type: string
    isDefault:
      type: boolean
    links:
      type: array
      items:
        $ref: '#/components/schemas/Link'
```

6.3.4. Spatial Dataset Key Field

The HTTP GET operation at path `/spatialdatasets/{spatialdatasetid}/keys/{keyname}` returns a list of key values from a specific key field of a specific spatial dataset.

6.3.4.1. Request

Requirement 19	**/req/core/spatialdatasets-spatialdatasetid-keys-keyname-op**
A	If the server implements the *data joining operations* operation set, it SHALL support the HTTP GET operation at the path `/spatialdatasets/{spatialdatasetid}/keys/{keyname}`.

6.3.4.2. Response

Requirement 20	**/req/core/spatialdatasets-spatialdatasetid-keys-keyname-success**
A	A successful execution of the operation SHALL be reported as a response with a HTTP status code `200`.
B	The server implementations SHALL support the JSON output format. Any other output formats MAY also be supported.
C	The response document is based on the spatialdataset key field schema. The response document SHALL include the following links: • A link to this document (link rel: 'self') • Links to this document in other supported media types (link rel: 'alternate') The links SHALL contain the parameters href, rel, title and type.

Description of properties in the response´s keys property:

Name	Description
key	Spatial dataset´s key field´s key value
links	Links to the different representations of this key value. The links SHALL have the properties href, rel and type. The value of the rel property SHALL be 'key'
title	Human-readable description of the key value

6.3.4.3. Errors

Requirement 21	/req/core/spatialdatasets-spatialdatasetid-keys-keyname-error
A	If an incorrect request is made to the server, it SHALL be reported as a response with a HTTP status code `400`. The response body SHALL contain an exception report message in the JSON format that is based on exception message schema. The value of the message property SHALL be 'InvalidParameterValue'. The locator property SHALL contain the name of the request parameter that caused the exception.
B	If the spatial dataset´s key field´s key values are not found on the server, it SHALL be reported as a response with a HTTP status code `404`. The response body SHALL contain an exception report message in the JSON format that is based on exception message schema. The value of the message property SHALL be 'Spatial dataset {spatialdatasetid} key field {keyname} not found'.
C	If there is an error in the server during the processing of the request, it SHALL be reported as a response with a HTTP status code `500`. The response body SHALL contain an exception report message in the JSON format that is based on exception message schema. The value of the message property SHALL be 'Internal server error'.

6.3.4.4. Response Schema for the Spatial Dataset Key Field

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

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

SpatialDatasetKeysKeynameObject:
  required:
  - key
  - links
  type: object
  properties:
    key:
      type: string
    links:
      type: array
      items:
        $ref: '#/components/schemas/Link'
    title:
      type: string
```

6.3.5. Spatial Dataset Key Field Key

The HTTP GET operation at path `/spatialdatasets/{spatialdatasetid}/keys/{keyname}/{key}` returns a specific key value, from a specific key field from a specific spatial dataset.

6.3.5.1. Request

Requirement 22	**/req/core/spatialdatasets-spatialdatasetid-keys-keyname-key-op**
A	If the server implements the *data joining operations* operation set it SHALL support the HTTP GET operation at the path `/spatialdatasets/{spatialdatasetid}/keys/{keyname}/{key}`.

6.3.5.2. Response

Requirement 23	**/req/core/spatialdatasets-spatialdatasetid-keys-keyname-key-success**
A	A successful execution of the operation SHALL be reported as a response with a HTTP status code `200`.
B	The server implementations SHALL support the JSON output format. Any other output formats MAY also be supported.
C	The response document is based on the spatialdataset key field key schema. The response document SHALL include the following links: • A link to this document (link rel: 'self') • Links to this document in other supported media types (link rel: 'alternate') The links SHALL contain the parameters href, rel, title and type **Description of properties of the key response property:**

Name	Description
key	Spatial dataset´s key field´s key value
title	Human-readable description of the key value

6.3.5.3. Errors

Requirement 24	/req/core/spatialdatasets-spatialdatasetid-keys-keyname-key-error
A	If an incorrect request is made to the server, it SHALL be reported as a response with a HTTP status code `400`. The response body SHALL contain an exception report message in the JSON format that is based on exception message schema. The value of the message property SHALL be 'InvalidParameterValue'. The locator property SHALL contain the name of the request parameter that caused the exception.
B	If the spatial dataset key value is not found on the server, it SHALL be reported as a response with a HTTP status code `404`. The response body SHALL contain an exception report message in the JSON format that is based on exception message schema. The value of the message property SHALL be 'Spatial dataset {spatialdatasetid} key field {keyname} key {key} not found'.
C	If there is an error in the server during the processing of the request, it SHALL be reported as a response with a HTTP status code `500`. The response body SHALL contain an exception report message in the JSON format that is based on exception message schema. The value of the message property SHALL be 'Internal server error'.

6.3.5.4. Response Schema for the Spatial Dataset Key Field Key

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

SpatialDatasetKeysKeynameKeyResponseObject:
  required:
  - key
  - links
  type: object
  properties:
    links:
      type: array
      items:
        $ref: '#/components/schemas/Link'
    key:
      $ref: '#/components/schemas/SpatialDatasetKeysKeynameKeyObject'

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

6.3.6. Data Joining from a CSV File with a Spatial Dataset

The HTTP POST operation at path `/joindata/{spatialdatasetid}/csv` joins attribute data from a csv file with a spatial dataset that is available on the server. The join is executed by using shared key values in the two datasets.

6.3.6.1. Request

<table>
<tr><th>Requirement 25</th><th>/req/core/joindata-spatialdatasetid-csv-op</th></tr>
<tr><td>A</td><td>
If the server implements the data joining operations operation set, it SHALL support the HTTP POST operation at path <code>/joindata/{spatialdatasetid}/csv</code>.

The csv file can either be uploaded to the server with the csvFile parameter or provided through URL link with the csvFileURL parameter.

The request SHALL contain the header:

• Content-Type: multipart/form-data.

If the csv file is provided by upload, it SHALL contain the header

• Content-Disposition: form-data; filename="[csv file´s name]"; name="csvFile"

If the sld file is provided by upload, it SHALL contain the header:

• Content-Disposition: form-data; filename="[sld file´s name]" name="sldFile";

Request´s form data parameters:
<table>
<tr><th>Name</th><th>Description</th><th>Type and values</th><th>Required</th></tr>
<tr><td>csvFile</td><td>The csv file (uploaded file)</td><td>File</td><td>Optional[a]</td></tr>
<tr><td>csvFileURL</td><td>The csv file URL</td><td>URL</td><td>Optional[a]</td></tr>
<tr><td>csvFileKeyColumnNumber</td><td>The number of the key column in the csv file (counting starts from 1</td><td>Integer</td><td>Mandatory</td></tr>
<tr><td>csvFileAttributeColumnNumberList</td><td>The numbers of the attribute columns in the csv file that will be joined with the spatial dataset</td><td>Integers separated by commas</td><td>Mandatory</td></tr>
</table>
</td></tr>
</table>

Name	Description	Type and values	Required
	(counting starts from 1)		
csvFileDelimiter	The delimiter character used in the csv file	String	Mandatory
csvFileHeaderRowNumber	The row number of csv file´s header row in the csv file (counting starts from 1)	Integer	Optional, (omit, if header row is not available in the csv file)
csvFileDataStartRowNumber	The row number where the data values start in the csv file. (counting starts from 1)	Integer	Optional[b]
outputFormats	List of outputs that will be included to the response document	String, (comma separated)	Optional[c]
csvFileDuplicateKeyHandlingMethod	Method for handling duplicate key values in the csv file. Value is one of the following: *first*, *last*, *count*, *add*, *average*	String	Optional[d]
sldFile	The sld file for WMS output (uploaded file)	File	Optional[e]

Name	Description	Type and values	Required
sldFileURL	URL containing sld file that will be applied to WMS output	URL	Optional[e]
spatialDatasetKey	The key field of the spatial dataset that will be used in the join operation	String	Optional[f]

[a] One of the parameters: csvFile or csvFileURL is mandatory to be used with the operation. The csvFile parameter can be used for uploading a csv file to the server. The csvFileURL parameter can be used for providing the csv file through URL link. If both parameters are provided in the query, the server SHALL send an exception with message 'DuplicateAttributeFileInput'.

[b] Default value for the parameter is 2. If csvFileHeaderRowNumber parameter is missing, default value is 1.

[c] Comma separated list of the outputs that will be included to the response document. The supported output formats can be found from the API description document. If the parameter value is empty or missing, a default value 'geojson' is used.

[d] Possible values are: *first*, *last*, *count*, *add*, *average*. The value *first* uses data values from the first row where the key is encountered, the value *last* takes the values from the last row where the key is encountered, the value *count* tells how many rows there were for the key in the attribute dataset, the value *add* adds the data values together from all rows where the key is encountered (only for numerical values), the value *average* calculates an average from all rows where the key is encountered (only for numerical values).

[e] Styling file can be provided either by uploading it by using the sldFile parameter or through URL link by using the sldFileURL parameter. If both parameters are provided in the query, the server SHALL send an exception with message 'DuplicateStylingFileInput'.

[f] If spatialDatasetKey parameter is omitted, a default key field of the spatial dataset will be used in the join operation.

6.3.6.2. Response

Requirement 26	/req/core/joindata-spatialdatasetid-csv-success
A	A successful execution of the operation SHALL be reported as a response with a HTTP status code `201`.

B

The server implementations SHALL support the JSON output format. Any other output formats MAY also be supported.

The response document contains information on the execution of the data joining operation, including links to the created outputs for the joined data.

The TJS implementations SHALL support the GeoJSON output format for the joined data and MAY support any other output formats. Other recommended output formats to be supported are WFS and WMS.

The supported output formats SHALL be listed in the API description document.

C

The response document is based on the data joining from a csv file with a spatial dataset schema.

The response document SHALL include the following links:

- a link to this document (link rel: 'self')
- links to this document in other supported media types (link rel: 'alternate')

Description of properties in the response´s join property:

Name	Description
inputs	Join operation inputs
joinId	Unique identifier for the join
joinInformation	Information on the execution of the data join operation
outputs	Links to the created outputs
timestamp	Timestamp when the join has been executed

Description of properties in the inputs property:

Name	Description
attributeDataset	Name or URL of the joined csv file
spatialDataset	Link objects that contain links to spatialdataset´s different representations. Properties href, rel and type are mandatory.

Description of properties in the joinInformation property:

Name	Description

Name	Description
additionalAttributeKeys	List of additional keys in the csv file that were not available in the spatial dataset keys
duplicateAttributeKeyHandlingMethod	The method that was used for handling duplicate keys in csv file
duplicateAttributeKeys	List of duplicate keys in the csv file
matchedSpatialDatasetKeys	List of spatial dataset keys that were successfully matched with attribute data
numberOfAdditionalAttributeKeys	The number of additional attribute key values in the attribute dataset that were not available in the spatial dataset
numberOfDuplicateAttributeKeys	The number of attribute keys that had duplicate entries
numberOfMatchedSpatialDatasetKeys	The number of spatial dataset keys, to which attribute data was joined successfully
numberOfUnmatchedSpatialDatasetKeys	The number of spatial dataset keys, to which attribute data couldn´t be joined
unmatchedSpatialDatasetKeys	List of spatial dataset keys, to which attribute data couldn´t be joined

Description of properties in the outputs property:

Name	Description
format	Name of the output format
layerName	Name of the joined data layer (For WMS and WFS outputs)
link	Link to the output
styleName	Name of the joined data layer style

6.3.6.3. Errors

Requirement 27	/req/core/joindata-spatialdatasetid-csv-error
A	If an incorrect request is made to the server, it SHALL be reported as a response with a HTTP status code `400`. The response body SHALL contain an exception report message in the JSON format that is based on exception message schema. The possible values for the

Requirement 27	message property are 'InvalidParameterValue', 'MissingParameterValue', 'DuplicateAttributeFileInput' and 'DuplicateStylingFileInput'. The locator **/req/core/joindata-spatialdatasetid-csv-error**
	property SHALL contain the name of the request parameter that caused the exception.
B	If the spatialdataset is not found on the server, it SHALL be reported as a response with a HTTP status code `404`. The response body SHALL contain an exception report message in the JSON format that is based on exception message schema. The value of the message property SHALL be 'Spatial dataset {spatialdatasetid} not found'.
C	If there is an error in the server during the processing of the request, it shall be reported as a response with a HTTP status code `500`. The response body SHALL contain an exception report message in the JSON format that is based on exception message schema. The value of the message property SHALL be 'Internal server error'.

6.3.6.4. Response schema for the Data Joining from a CSV File with a Spatial Dataset

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

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

JoinDataObject:
  required:
  - inputs
  - joinId
  - outputs
  - timestamp
  type: object
  properties:
    joinId:
      type: integer
      format: int32
    timestamp:
      type: string
      format: date-time
    inputs:
      $ref: '#/components/schemas/JoinInputsObject'
    outputs:
      type: array
      items:
        $ref: '#/components/schemas/OutputObject'
    joinInformation:
      $ref: '#/components/schemas/JoinInformationObject'

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

OutputObject:
```

```
OutputObject:
  required:
  - format
  - link
  type: object
  properties:
    format:
      type: string
    layerName:
      type: string
    link:
      type: string
    styleName:
      type: string

JoinInformationObject:
  type: object
  properties:
    numberOfMatchedSpatialDatasetKeys:
      type: integer
      format: int32
    numberOfUnmatchedSpatialDatasetKeys:
      type: integer
      format: int32
    numberOfAdditionalAttributeKeys:
      type: integer
      format: int32
    matchedSpatialDatasetKeys:
      type: array
      items:
        type: string
    unmatchedSpatialDatasetKeys:
      type: array
      items:
        type: string
    additionalAttributeKeys:
      type: array
      items:
        type: string
    duplicateAttributeKeys:
      type: array
      items:
        type: string
    duplicateAttributeKeyHandlingMethod:
      type: string
    numberOfDuplicateAttributeKeys:
      type: integer
      format: int32
```

6.3.7. Joins

The HTTP GET operation at path `/joins` returns list of all joins that are available on the server.

6.3.7.1. Request

Requirement 28	**/req/core/joins-op**
A	If the server implements the *data joining operations* operation set it SHALL support the HTTP GET operation at the path `/joins`.

6.3.7.2. Response

Requirement 29	**/req/core/joins-success**
A	A successful execution of the operation shall be reported as a response with a HTTP status code `200`.
B	The server implementations SHALL support the JSON output format. Any other output formats MAY also be supported. The server SHALL return metadata on all joins that are available on the server.
C	The response document is based on the joins schema.

The response document SHALL include the following links:

- a link to this document (link rel: 'self')
- links to this document in other supported media types (link rel: 'alternate')

Description of elements in joins property:

Name	Description
joinId	Unique identifier for the join. The identifier is used in other operations for indicating the join in question
joinedLayerName	Name of the created layer for the join
joinTimestamp	Timestamp when the join has been executed
links	Links to the different representations of the join. The links SHALL have the properties href, rel and type. The value of the rel property SHALL be 'join'.

6.3.7.3. Errors

Requirement 30	/req/core/joins-error
A	If joins are not found on the server, it SHALL be reported as a response with a HTTP status code `404`. The response body SHALL contain an exception report message in the JSON format that is based on exception message schema. The value of the message property SHALL be 'Joins not found'.
B	If there is an error in the server during the processing of the request, it SHALL be reported as a response with a HTTP status code `500`. The response body SHALL contain an exception report message in the JSON format that is based on exception message schema. The value of the message property SHALL be 'Internal server error'.

6.3.7.4. 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'

JoinsObject:
  required:
  - joinId
  - joinTimestamp
  - links
  type: object
  properties:
    joinId:
      type: integer
      format: int32
    joinedLayerName:
      type: string
    joinTimestamp:
      type: string
      format: date-time
    links:
      type: array
      items:
        $ref: '#/components/schemas/Link'
```

6.3.8. Join

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

6.3.8.1. Request

Requirement 31	**/req/core/joins-joinid-op**
A	If the server implements the *data joining operations* operation set it SHALL support the HTTP GET operation at the path `/joins/{joinid}`.

6.3.8.2. Response

Requirement 32	**/req/core/joins-joinid-success**
A	A successful execution of the operation SHALL be reported as a response with a HTTP status code `200`.
B	The server implementations SHALL support the JSON output format. Any other output formats MAY also be supported.
C	The response document is based on the join schema. The response document SHALL include the following links: • a link to this document (link rel: 'self') • links to this document in other supported media types (link rel: 'alternate') **Description of properties in the response´s join property:**

Name	Description
inputs	Join operation inputs
joinId	Unique identifier for the join
joinInformation	Information on the execution of the data join operation
outputs	Links to the created outputs
timestamp	Timestamp when the join has been executed

Description of properties in the inputs property:

Name	Description
attributeDataset	Name or URL of the joined csv file
spatialDataset	Link objects that contain links to spatialdataset´s different representations. Properties href, rel and type are mandatory.

Description of properties in the joinInformation property:

Name	Description
additionalAttributeKeys	List of additional keys in the csv file that were not available in the spatial dataset keys
duplicateAttributeKeyHandlingMethod	The method that was used for handling duplicate keys in csv file
duplicateAttributeKeys	List of duplicate keys in the csv file
matchedSpatialDatasetKeys	List of spatial dataset keys that were successfully matched with attribute data
numberOfAdditionalAttributeKeys	The number of additional attribute key values in the attribute dataset that were not available in the spatial dataset
numberOfDuplicateAttributeKeys	The number of attribute keys that had duplicate entries
numberOfMatchedSpatialDatasetKeys	The number of spatial dataset keys, to which attribute data was joined successfully
numberOfUnmatchedSpatialDatasetKeys	The number of spatial dataset keys, to

Name	Description
unmatchedSpatialDatasetKeys	List of spatial dataset keys, to which attribute data couldn´t be joined

Description of properties in the outputs property:

Name	Description
format	Name of the output format
layerName	Name of the joined data layer (For WMS and WFS outputs)
link	link Link to the output
styleName	Name of the joined data layer style

6.3.8.3. Errors

Requirement 33	/req/core/joins-joinid-error
A	If spatial datasets are not found on the server, it SHALL be reported as a response with a HTTP status code `404`. The response body SHALL contain an exception report message in the JSON format that is based on exception message schema. The value of the message property SHALL be 'Join {joinid} not found'.
B	If there is an error in the server during the processing of the request, it SHALL be reported as a response with a HTTP status code `500`. The response body SHALL contain an exception report message in the JSON format that is based on exception message schema. The value of the message property SHALL be 'Internal server error'.

6.3.8.4. 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/JoinDataObject'

JoinDataObject:
  required:
```

```
  - inputs
  - joinId
  - outputs
  - timestamp
  type: object
  properties:
    joinId:
      type: integer
      format: int32
    timestamp:
      type: string
      format: date-time
    inputs:
      $ref: '#/components/schemas/JoinInputsObject'
    outputs:
      type: array
      items:
        $ref: '#/components/schemas/OutputObject'
    joinInformation:
      $ref: '#/components/schemas/JoinInformationObject'

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

OutputObject:
  required:
  - format
  - link
  type: object
  properties:
    format:
      type: string
    layerName:
      type: string
    link:
      type: string
    styleName:
      type: string

JoinInformationObject:
  type: object
  properties:
    numberOfMatchedSpatialDatasetKeys:
      type: integer
      format: int32
    numberOfUnmatchedSpatialDatasetKeys:
      type: integer
      format: int32
    numberOfAdditionalAttributeKeys:
      type: integer
      format: int32
    matchedSpatialDatasetKeys:
      type: array
      items:
        type: string
    unmatchedSpatialDatasetKeys:
      type: array
      items:
        type: string
    additionalAttributeKeys:
      type: array
      items:
        type: string
    duplicateAttributeKeys:
      type: array
      items:
        type: string
    duplicateAttributeKeyHandlingMethod:
      type: string
    numberOfDuplicateAttributeKeys:
      type: integer
      format: int32
```

6.3.9. Join Updating with CSV File Data

The HTTP POST operation at path `/joins/{joinid}/csv` updates fully the specific join with a data from a csv file.

6.3.9.1. Request

<table>
<tr><th>Requirement 34</th><th>/req/core/joins-joinid-csv-op</th></tr>
<tr><td>A</td><td>
If the server implements the data joining operations operation set it MAY support the HTTP POST operation at the path <code>/joins/{joinid}/csv</code>.

The csv file can either be uploaded to the server with the csvFile parameter or provided through URL link with the csvFileURL parameter.

The request SHALL contain the header:

• Content-Type: multipart/form-data.

If csv file is provided by upload, it SHALL contain the header

• Content-Disposition: form-data; filename="[csv file´s name]"; name="csvFile"

If sld file is provided by upload, it SHALL contain the header:

• Content-Disposition: form-data; filename="[sld file´s name]" name="sldFile";

Request´s form data parameters:
<table>
<tr><th>Name</th><th>Description</th><th>Type and values</th><th>Required</th></tr>
<tr><td>csvFile</td><td>The csv file (uploaded file)</td><td>File</td><td>Optional[a]</td></tr>
<tr><td>csvFileURL</td><td>The csv file URL</td><td>URL</td><td>Optional[a]</td></tr>
<tr><td>csvFileKeyColumnNumber</td><td>The number of the key column in the csv file (counting starts from 1</td><td>Integer</td><td>Mandatory</td></tr>
<tr><td>csvFileAttributeColumnNumberList</td><td>The numbers of the attribute columns in the csv file that will be joined with the spatial dataset (counting starts from</td><td>Integers separated by commas</td><td>Mandatory</td></tr>
</table>
</td></tr>
</table>

Name	Description[1)]	Type and values	Required
csvFileDelimiter	The delimiter character used in the csv file	String	Mandatory
csvFileHeaderRowNumber	The row number of csv file´s header row in the csv file (counting starts from 1)	Integer	Optional, (omit, if header row is not available in the csv file)
csvFileDataStartRowNumber	The row number where the data values start in the csv file (counting starts from 1)	Integer	Optional[b]
outputFormats	List of outputs that will be included to the response document	String, (comma separated)	Optional[c]
csvFileDuplicateKeyHandlingMethod	Method for handling duplicate key values in the csv file. Value is one of the following: *first*, *last*, *count*, *add*, *average*	String	Optional[d]
sldFile	The sld file for WMS output (uploaded file)	File	Optional[e]
sldFileURL	URL containing	URL	Optional[e]

Name	Description	Type and values	Required
	sld file that will be applied to WMS output		
spatialDatasetKey	The key field of the spatial dataset that will be used in the join operation	String	Optional[f]

[a] One of the parameters: csvFile or csvFileURL is mandatory to be used with the operation. The csvFile parameter can be used for uploading a csv file to the server. The csvFileURL parameter can be used for providing the csv file through URL link. If both parameters are provided in the query, the server SHALL send an exception with message 'DuplicateAttributeFileInput'.

[b] Default value for the parameter is 2. If csvFileHeaderRowNumber parameter is missing, default value is 1.

[c] Comma separated list of the outputs that will be included to the response document. The supported output formats can be found from the API description document. If the parameter value is empty or missing, a default value 'geojson' is used.

[d] Possible values are: *first*, *last*, *count*, *add*, *average*. The value *first* uses data values from the first row where the key is encountered, the value *last* takes the values from the last row where the key is encountered, the value *count* tells how many rows there were for the key in the attribute dataset, the value *add* adds the data values together from all rows where the key is encountered (only for numerical values), the value *average* calculates an average from all rows where the key is encountered (only for numerical values).

[e] Styling file can be provided either by uploading it by using the sldFile parameter or through URL link by using the sldFileURL parameter. If both parameters are provided in the query, the server SHALL send an exception with message 'DuplicateStylingFileInput'.

[f] If spatialDatasetKey parameter is omitted, a default key field of the spatial dataset will be used in the join operation.

6.3.9.2. Response

Requirement 35	/req/core/joins-joinid-csv-success
A	A successful execution of the operation SHALL be reported as a response with a HTTP status code `200`.
B	The server implementations SHALL support the JSON output format. Any other

output formats MAY also be supported.

The response document contains information on the execution of the data joining operation, including links to the created outputs for the joined data.

The TJS implementations SHALL support the GeoJSON output format for the joined data and MAY support any other output formats. Other recommended output formats to be supported are WFS and WMS.

The supported output formats SHALL be listed in the API description document.

C

The response document is based on the join updating with csv file data schema.

The response document SHALL include the following links:

- a link to this document (link rel: 'self')
- links to this document in other supported media types (link rel: 'alternate')

Description of properties in the response´s join property:

Name	Description
inputs	Join operation inputs
joinId	Unique identifier for the join
joinInformation	Information on the execution of the data join operation
outputs	Links to the created outputs
timestamp	Timestamp when the join has been executed

Description of properties in the inputs property:

Name	Description
attributeDataset	Name or URL of the joined csv file
spatialDataset	Link objects that contain links to spatialdataset´s different representations. Properties href, rel and type are mandatory.

Description of properties in the joinInformation property:

Name	Description
additionalAttributeKeys	List of additional keys in the csv file that were not available in the spatial dataset

Name	Description
duplicateAttributeKeyHandlingMethod	The method that was used for handling duplicate keys in csv file
duplicateAttributeKeys	List of duplicate keys in the csv file
matchedSpatialDatasetKeys	List of spatial dataset keys that were successfully matched with attribute data
numberOfAdditionalAttributeKeys	The number of additional attribute key values in the attribute dataset that were not available in the spatial dataset
numberOfDuplicateAttributeKeys	The number of attribute keys that had duplicate entries
numberOfMatchedSpatialDatasetKeys	The number of spatial dataset keys, to which attribute data was joined successfully
numberOfUnmatchedSpatialDatasetKeys	The number of spatial dataset keys, to which attribute data couldn´t be joined
unmatchedSpatialDatasetKeys	List of spatial dataset keys, to which attribute data couldn´t be joined

Description of properties in the outputs property:

Name	Description
format	Name of the output format
layerName	Name of the joined data layer (For WMS and WFS outputs)
link	link Link to the output
styleName	Name of the joined data layer style

6.3.9.3. Errors

Requirement 36	/req/core/joins-joinid-csv-error
A	If an incorrect request is made to the server, it SHALL be reported as a response with a HTTP status code `400`. The response body SHALL contain an exception report message in the JSON format that is based on exception message schema. The possible values for the message property are 'InvalidParameterValue', 'MissingParameterValue', 'DuplicateAttributeFileInput' and 'DuplicateStylingFileInput'. The locator property SHALL contain the name of the request parameter that caused the

Requirement 36	/req/core/joins-joinid-csv-error
B	If the join is not found on the server, it SHALL be reported as a response with a HTTP status code `404`. The response body SHALL contain an exception report message in the JSON format that is based on exception message schema. The value of the message property SHALL be 'Join {joinid} not found'.
C	If there is an error in the server during the processing of the request, it shall be reported as a response with a HTTP status code `500`. The response body SHALL contain an exception report message in the JSON format that is based on exception message schema. The value of the message property SHALL be 'Internal server error'.

exception.

6.3.9.4. Response schema for the Join Updating with CSV File Data

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

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

JoinDataObject:
  required:
  - inputs
  - joinId
  - outputs
  - timestamp
  type: object
  properties:
    joinId:
      type: integer
      format: int32
    timestamp:
      type: string
      format: date-time
    inputs:
      $ref: '#/components/schemas/JoinInputsObject'
    outputs:
      type: array
      items:
        $ref: '#/components/schemas/OutputObject'
    joinInformation:
      $ref: '#/components/schemas/JoinInformationObject'

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

OutputObject:
  required:
  - format
  - link
  type: object
  properties:
```

```
    format:
      type: string
    layerName:
      type: string
    link:
      type: string
    styleName:
      type: string

JoinInformationObject:
  type: object
  properties:
    numberOfMatchedSpatialDatasetKeys:
      type: integer
      format: int32
    numberOfUnmatchedSpatialDatasetKeys:
      type: integer
      format: int32
    numberOfAdditionalAttributeKeys:
      type: integer
      format: int32
    matchedSpatialDatasetKeys:
      type: array
      items:
        type: string
    unmatchedSpatialDatasetKeys:
      type: array
      items:
        type: string
    additionalAttributeKeys:
      type: array
      items:
        type: string
    duplicateAttributeKeys:
      type: array
      items:
        type: string
    duplicateAttributeKeyHandlingMethod:
      type: string
    numberOfDuplicateAttributeKeys:
      type: integer
      format: int32
```

6.3.10. Join Deleting

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

6.3.10.1. Request

Requirement 37	**/req/core/joins-joinid-delete-op**
A	If the server implements the *data joining operations* operation set, it MAY support the HTTP DELETE operation at the path `/joins/{joinid}`.

6.3.10.2. Response

Requirement 38	**/req/core/joins-joinid-delete-success**
A	A successful execution of the operation shall be reported as a response with a HTTP status code `204`. The response body SHALL be empty.

6.3.10.3. Errors

Requirement 39	**/req/core/joins-joinid-delete-error**
A	If the join is not found on the server, it shall be reported as a response with a HTTP status code `404`. The response body SHALL contain an exception report message in the JSON

Requirement 39	/req/core/joins-joinid-delete-error
	format that is based on exception message schema. The value of the message property SHALL be 'Join {joinid} not found'.
B	If there is an error in the server during the processing of the request, it shall be reported as a response with a HTTP status code `500`. The response body SHALL contain an exception report message in the JSON format that is based on exception message schema. The value of the message property SHALL be 'Internal server error'.

6.3.10.4. Response schema for the Join Deleting

The response of the operation is empty.

6.4. Operation Set: File Joining Operations

The operation set *file joining operations* contains an operation for joining attribute data from an inputted csv file with an inputted GeoJSON file. The response of the operation is the GeoJSON file that contains also the joined attributes.

6.4.1. Data Joining from a CSV File with a GeoJSON file

The HTTP POST operation at path `/joinfiles/geojson/csv` joins a data from a csv file with a GeoJSON file.

6.4.1.1. Request

Requirement 40	/req/core/joinfiles-geojson-csv-op
A	If the server supports the *file joining operations* operation set it SHALL support the HTTP POST operation at the path `/joinfiles/geojson/csv`. The GeoJSON file can either be uploaded to the server with the geojsonFile parameter or provided through URL link with the geojsonFileURL parameter. The csv file can either be uploaded to the server with the csvFile parameter or provided through URL link with the csvFileURL parameter. The request SHALL contain the header: • Content-Type: multipart/form-data If geojson file is provided by upload, it SHALL contain the header: • Content-Disposition: form-data; filename="[geojson file´s name]"; name="geojsonFile" If csv file is provided by upload, it SHALL contain the header: • Content-Disposition: form-data; filename="[csv file´s name]"; name="csvFile" **Request´s form data parameters:**

Name	Description	Type and values	Required
geojsonFile	The GeoJSON file (uploaded file)	File	Optional[a]
geojsonFileURL	A URL link to the GeoJSON file	URL type, not empty	Optional[a]
geojsonFileKeyFieldPath	The path to the key field name in the GeoJSON file that contains the key values. Example: 'features.properties.kunta'	String	Mandatory
csvFile	The csv file (uploaded file)	File	Optional[b]
csvFileURL	A URL link to the csv file	URL type, not empty	Optional[b]
csvFileKeyColumnNumber	The column number in the csv file that contains key values. (Counting starts from 1)	Integer	Mandatory
csvFileAttributeColumnNumberList	The numbers of the columns in the csv file that will be joined with a GeoJSON file. When multiple columns are joined the values shall be comma-separated. (Counting starts from 1)	Integer (multiple values are comma-separated)	Mandatory
csvFileDelimiter	The delimiter character used in the csv file	String	Mandatory
csvFileHeaderRowNumber	TThe row number of csv file´s header row in the csv file (Counting starts from 1)	Integer	Optional, (omit, if header row is not available in the csv file)
csvFileDataStartRowNumber	The row number where the data values start in the csv file (Counting starts from 1)	Integer	Optional[c]
csvFileDuplicateKeyHandlingMethod	Method for handling duplicate key values in	String	Optional[d]

Name	Description	Type and values	Required
	the csv file. Value is one of the following: *first*, *last*, *count*, *add*, *average*		

[a] One of the parameters: geojsonFile or geojsonFileURL is mandatory to be used with the operation. The geojsonFile parameter can be used for uploading a geojson file to the server. The geojsonFileURL parameter can be used for providing the geojson file through URL link. If both parameters are provided in the query, the server SHALL send an exception with message 'DuplicateGeojsonFileInput'.

[b] One of the parameters: csvFile or csvFileURL is mandatory to be used with the operation. The csvFile parameter can be used for uploading a csv file to the server. The csvFileURL parameter can be used for providing the csv file through URL link. If both parameters are provided in the query, the server SHALL send an exception with message 'DuplicateAttributeFileInput'.

[c] Default value for the parameter is 2. If csvFileHeaderRowNumber parameter is missing, default value is 1.

[d] Possible values are: *first*, *last*, *count*, *add*, *average*. The value *first* uses data values from the first row where the key is encountered, the value *last* takes the values from the last row where the key is encountered, the value *count* tells how many rows there were for the key in the attribute dataset, the value *add* adds the data values together from all rows where the key is encountered (only for numerical values), the value *average* calculates an average from all rows where the key is encountered (only for numerical values).

6.4.1.2. Response

Requirement 41	/req/core/joinfiles-geojson-csv-success
A	A successful execution of the operation SHALL be reported as a response with a HTTP status code `200`. The server SHALL return the GeoJSON file that includes the joined fields from the csv file.

6.4.1.3. Errors

Requirement 42	/req/core/joinfiles-geojson-csv-error
A	If an incorrect request is made to the server, it SHALL be reported as a response with a HTTP status code `400`. The response body SHALL contain an exception report message in the JSON format that is based on exception message schema. The Possible values for the message property are 'InvalidParameterValue', 'MissingParameterValue', 'DuplicateGeojsonFileInput' and 'DuplicateAttributeFileInput'
B	If there is an error in the server during the processing of the request, it shall be reported as a response with a HTTP status code `500`. The response body SHALL contain an exception report message in the JSON format that is based on exception message schema. The value of the message

Requirement 42	property SHALL be 'Internal server error'. **/req/core/joinfiles-geojson-csv-error**

6.4.1.4. Response Schema for the Spatial Datasets

The response of the operation SHALL be the GeoJSON file that contains also the joined attributes from the csv file.

---

7. Media Types for any data encoding(s)

7.1. Operation sets: discovery operations and data joining operations

Requirement 43	**/req/core/discovery-operations-and-data-joining-operations-outputs**
A	The server implementations SHALL support the JSON output format for all operations in the operation sets: *discovery operations* and *data joining operations*. Any other output formats MAY also be supported.

7.2. Outputs for the joined data in operation set: data joining operations

Requirement 44	**/req/core/data-joining-operations-joined-data-outputs**
A	The server implementations SHALL support the GeoJSON format for the joined data in the operation set *data joining operations*. Any other formats for the joined data MAY also be supported. Other recommended output formats to be supported are WFS and WMS.

7.3. Operation set: file joining operations

Requirement 45	**/req/core/file-joining-operations-output**
A	The server implementations SHALL support the GeoJSON output format for the operation defined in the *file joining operations* operation set.

Annex A: Abstract Test Suite (Normative)

A.1. Conformance Class "Core"

Conformance Class	
http://www.opengis.net/spec/tjs/2.0/conf/core	
Target type	Web API
Requirements class	Requirements Class "Core"

A.1.1. Landing Page {root}/

Abstract Test 1	**/ats/core/root-op**
Test Purpose	Validate that the landing page can be retrieved from the expected location.
Requirement	/req/core/root-op
Test Method	1. Issue an HTTP GET request on URL `{root}/`. 2. Validate that the document was returned with a status code `200`. 3. Validate the contents of the returned document using test /ats/core/root-success

Abstract Test 2	**/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. • Validate that the landing page includes a 'service-desc' and/or 'service-doc' link to an API Definition • Validate that the landing page includes a 'conformance' link to the conformance class declaration For servers that implement the *data joining operations* operation set: • Validate that the landing page includes a 'spatialdatasets' link to the spatialdatasets metadata • Validate that the landing page includes a "joins' link to the joins

A.1.2. API Definiton path {root}/api

Abstract Test 3	**/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
Test Method	1. Issue a HTTP GET request to the URL `{root}/api`. 2. Validate that a document was returned with a status code `200`. 3. Validate the contents of the returned document using test /ats/core/api-definition-success

Abstract Test 4	**/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.1.3. Conformance {root}/conformance

Abstract Test 5	**/ats/core/conformance-op**
Test Purpose	Validate that a Conformance declaration can be retrieved from the expected location.
Requirement	/req/core/conformance-op
Test Method	1. Issue an HTTP GET request to the URL `{root}/conformance`. 2. Validate that a document was returned with a status code `200`. 3. Validate the contents of the returned document using test /ats/core/conformance-success

Abstract Test 6	**/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	1. Validate the response document against conformance schema. 2. Validate that the document includes the conformance class 'http://www.opengis.net/spec/tjs/2.0/conf/core' 3. Validate that the document lists all other OGC API conformance classes the server implements

A.2. Conformance class: Core / Data Joining CSV

Conformance Class

http://www.opengis.net/spec/tjs/2.0/conf/core/data-joining-csv	
Target type	Web API
Requirements class	Requirements Class "Core"

A.2.1. Spatial Datasets {root}/spatialdatasets

Abstract Test 7	**/ats/core/spatialdatasets-op**
Test Purpose	Validate that the information about spatial datasets can be retrieved from the expected location.
Requirement	/req/core/spatialdatasets-op
Test Method	1. Issue an HTTP GET request to the URL `{root}/spatialdatasets`. 2. Validate that a document was returned with a status code `200`. 3. Validate the contents of the returned document using test /ats/core/spatialdatasets-success

Abstract Test 8	**/ats/core/spatialdatasets-success**
Test Purpose	Validate that the spatial datasets content complies with the required structure and contents.
Requirement	/req/core/spatialdatasets-success
Test Method	1. Validate that the response document complies with spatialdatasets schema.

A.2.2. Spatial Dataset {root}/spatialdatasets/{spatialdatasetid}

Abstract Test 9	**/ats/core/spatialdatasets-spatialdatasetid-op**
Test Purpose	Validate that a spatial dataset information can be retrieved from the expected location.
Requirement	/req/core/spatialdatasets-spatialdatasetid-op
Test Method	1. For a list of all spatial datasets (path `{root}/spatialdatasets`), issue an HTTP GET request to the URL `{root}/spatialdatasets/{spatialdatasetid}` where {spatialdatasetid} is the spatialDatasetId property of a spatial dataset. 2. Validate that a document was returned with a status code `200`. 3. Validate the contents of the returned document using test /ats/core/spatialdatasets-spatialdatasetid-success

Abstract Test 10	**/ats/core/spatialdatasets-spatialdatasetid-success**
Test Purpose	Validate that the spatial dataset complies with the required structure and contents.

Requirement	/req/core/spatialdatasets-spatialdatasetid-success
Test Method	1. Validate that the response document complies with spatialdataset schema.

A.2.3. Spatial Dataset keys {root}/spatialdatasets/{spatialdatasetid}/keys

Abstract Test 11	**/ats/core/spatialdatasets-spatialdatasetid-keys-op**
Test Purpose	Validate that the information on spatial dataset key fields can be retrieved from the expected location.
Requirement	/req/core/spatialdatasets-spatialdatasetid-keys-op
Test Method	1. For a spatial dataset (path `{root}/spatialdatasets/{spatialdatasetid}`), issue an HTTP GET request to the URL `{root}/spatialdatasets/{spatialdatasetid}/keys` where {spatialdatasetid} is the spatialDatasetId property of a spatial dataset. 2. Validate that a document was returned with a status code `200`. 3. Validate the contents of the returned document using test /ats/core/spatialdatasets-spatialdatasetid-keys-success

Abstract Test 12	**/ats/core/spatialdatasets-spatialdatasetid-keys-success**
Test Purpose	Validate that the spatial datasets keys content complies with the required structure and contents.
Requirement	/req/core/spatialdatasets-spatialdatasetid-keys-success
Test Method	1. Validate that the response document complies with spatialdataset key fields schema.

A.2.4. Spatial Dataset key field {root}/spatialdatasets/{spatialdatasetid}/keys/{keyname}

Abstract Test 13	**/ats/core/spatialdatasets-spatialdatasetid-keys-keyname-op**
Test Purpose	Validate that the information on keys that belong to the spatial dataset´s key field can be retrieved from the expected location.
Requirement	/req/core/spatialdatasets-spatialdatasetid-keys-keyname-op
Test Method	1. For a spatial dataset (path `{root}/spatialdatasets/{spatialdatasetid}`), issue an HTTP GET request to the URL `{root}/spatialdatasets/{spatialdatasetid}/keys/{keyname}` where {spatialdatasetid} is the spatialDatasetId property of a spatial dataset and {keyname} is the keyName property of the spatial dataset key field. 2. Validate that a document was returned with a status code `200`. 3. Validate the contents of the returned document using test /ats/core/spatialdatasets-spatialdatasetid-keys-keyname-success

Abstract Test 14	**/ats/core/spatialdatasets-spatialdatasetid-keys-keyname-success**

Test Purpose	Validate that the spatial dataset key field´s contents comply with the required structure and contents.
Requirement	/req/core/spatialdatasets-spatialdatasetid-keys-keyname-success
Test Method	Validate that the response document complies with spatial dataset key field schema.

A.2.5. Spatial Dataset key field key {root}/spatialdatasets/{spatialdatasetid}/keys/{keyname}/{key}

Abstract Test 15	**/ats/core/spatialdatasets-spatialdatasetid-keys-keyname-key-op**
Test Purpose	Validate that the spatial dataset´s key field´s key information can be retrieved from the expected location.
Requirement	/req/core/spatialdatasets-spatialdatasetid-keys-keyname-key-op
Test Method	1. For a spatial dataset (path `{root}/spatialdatasets/{spatialdatasetid}`), issue an HTTP GET request to the URL `{root}/spatialdatasets/{spatialdatasetid}/keys/{keyname}/{key}` where {spatialdatasetid} is the spatialDatasetId property of a spatial dataset, {keyname} is the keyName property of the spatial dataset key field and {key} is the name of the key property of the spatial dataset key value. 2. Validate that a document was returned with a status code `200`. 3. Validate the contents of the returned document using test /ats/core/spatialdatasets-spatialdatasetid-keys-keyname-key-success

Abstract Test 16	**/ats/core/spatialdatasets-spatialdatasetid-keys-keyname-key-success**
Test Purpose	Validate that the spatial dataset´s key content complies with the required structure and contents.
Requirement	/req/core/spatialdatasets-spatialdatasetid-keys-keyname-key-success
Test Method	Validate that the response document complies with the spatial dataset key field key schema.

A.2.6. Spatial Dataset joining with CSV data {root}/joindata/{spatialdatasetid}/csv

Abstract Test 17	**/ats/core/joindata-spatialdatasetid-csv-op**
Test Purpose	Validate that the data can be joined from a csv file with a specific spatial dataset from expected location.
Requirement	/req/core/joindata-spatialdatasetid-csv-op
Test Method	1. Issue an HTTP POST request to the URL `{root}/joindata/{spatialdatasetid}/csv` where {spatialdatasetid} is the spatialDatasetId property of a spatial dataset (from query `{root}/spatialdatasets`).

	2. Validate that a document was returned with a status code `201`. 3. Validate the contents of the returned document using test /ats/core/joindata-spatialdatasetid-csv-success

Abstract Test 18	**/ats/core/joindata-spatialdatasetid-csv-success**
Test Purpose	Validate that the data join response document complies with the required structure and contents.
Requirement	/req/core/joindata-spatialdatasetid-csv-success
Test Method	1. Validate that the response document complies with the join data csv schema. 2. Validate that the response document contains the joined data in all the requested output formats that are supported by the service implementation.

A.2.7. Joins {root}/joins

Abstract Test 19	**/ats/core/joins-op**
Test Purpose	Validate that the information about joins can be retrieved from the expected location.
Requirement	/req/core/joins-op
Test Method	1. Issue an HTTP GET request to the URL `{root}/joins`. 2. Validate that a document was returned with a status code `200`. 3. Validate the contents of the returned document using test /ats/core/joins-success

Abstract Test 20	**/ats/core/joins-success**
Test Purpose	Validate that the joins content complies with the required structure and contents.
Requirement	/req/core/joins-success
Test Method	Validate that the response document complies with joins schema.

A.2.8. Join {root}/joins/{joinid}

Abstract Test 21	**/ats/core/joins-joinid-op**
Test Purpose	Validate that the information about a join can be retrieved from the expected location.
Requirement	/req/core/joins-joinid-op
Test Method	1. For a list of all joins (path `{root}/joins`), issue an HTTP GET request to the URL `{root}/joins/{joinid}` where {joinid} is the joinId property of a join. 2. Validate that a document was returned with a status code `200`. 3. Validate the contents of the returned document using test /ats/core/joins-

	joinid-success

Abstract Test 22	/ats/core/joins-joinid-success
Test Purpose	Validate that the join content complies with the required structure and contents.
Requirement	/req/core/joins-joinid-success
Test Method	Validate that the response document complies with join schema.

A.3. Conformance class: Core / Data Joining CSV Update

Conformance Class	
http://www.opengis.net/spec/tjs/2.0/conf/core/data-joining-csv-update	
Target type	Web API
Requirements class	Requirements Class "Core"

A.3.1. Join Update CSV {root}/joins/{joinid}/csv

Abstract Test 23	/ats/core/joins-joinid-csv-op
Test Purpose	Validate that the join can be updated fully with a CSV data from expected location.
Requirement	/req/core/joins-joinid-csv-op
Test Method	1. Issue an HTTP POST request to the URL `{root}/joins/{joinid}/csv` where {joinid} is the joinId property of a join (from query `{root}/joins/{joinid}`). 2. Validate that a document was returned with a status code `200`. 3. Validate the contents of the returned document using test /ats/core/joins-joinid-csv-success

Abstract Test 24	/ats/core/joins-joinid-csv-success
Test Purpose	Validate that the data can be joined from a csv file with a specific spatial dataset.
Requirement	/req/core/joins-joinid-csv-success
Test Method	1. Validate that the response document complies with join update schema. 2. Validate that the response document contains the joined data in all the requested output formats that are supported by the service implementation.

A.4. Conformance class: Core / Data Joining Delete

Conformance Class	
http://www.opengis.net/spec/tjs/2.0/conf/core/data-joining-delete	

Target type	Web API
Requirements class	Requirements Class "Core"

A.4.1. Join Delete {root}/joins/{joinid}

Abstract Test 25	**/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	1. Issue an HTTP DELETE request to the URL `{root}/joins/{joinid}` where {joinid} is the joinId property of a join (from query `{root}/joins{joinid}`). 2. Validate that a document was returned with a status code `204`. 3. Validate the contents of the returned document using test /ats/core/joins-joinid-delete-success

Abstract Test 26	**/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	1. Validate that the join has been deleted from the server by issuing an HTTP GET request to the URL `{root}/joins/{joinid}` where {joind} is the same joinId property of the join that was used in the delete request. 2. Validate that the server sent a response code `404`.

A.5. Conformance class: Core / File Joining GeoJSON CSV

Conformance Class	
http://www.opengis.net/spec/tjs/2.0/req/core/file-joining-geojson-csv	
Target type	Web API
Requirements class	Requirements Class "Core"

A.5.1. File Joining GeoJSON CSV {root}//joinfiles/geojson/csv

Abstract Test 27	**/ats/core/joinfiles-geojson-csv-op**
Test Purpose	Validate that data from csv file can be joined with GeoJSON file from the expected location.
Requirement	/req/core/joinfiles-geojson-csv-op
Test Method	1. Issue an HTTP POST request to the URL `{root}/joinfiles/geojson/csv`. 2. Validate that a document was returned with a status code `200`. 3. Validate the contents of the returned document using test /ats/core/joinfiles-

	geojson-csv-success

Abstract Test 28	**/conf/core/joinfiles-geojson-csv-success**
Test Purpose	Validate that the GeoJSON file contains the data that was joined from the CSV file.
Requirement	/req/core/joinfiles-geojson-csv-success
Test Method	Validate that the GeoJSON document contains the attributes that were joined from the CSV file

Annex B: Revision History

Date	Release	Editor	Primary clauses modified	Description
2020-04-09	1.0.0-SNAPSHOT	P. Latvala	all	Document into asciidoc-based format

Annex C: Bibliography

- Internet Assigned Numbers Authority (IANA). **Link Relation Types** [online, viewed 2020-04-09], Available at https://www.iana.org/assignments/link-relations/link-relations.xml

Last updated 2020-04-09 13:48:04 +0300