INTERNATIONAL STANDARD

ISO 19168-2

First edition
2022-09

Geographic information – Geospatial API for features —

Part 2: **Coordinate Reference Systems by Reference**

Information géographique — API géospatiale pour les entités —

Partie 2: Systèmes de coordonnées de référence par référence

Reference number
ISO 19168-2:2022(E)

Contents

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO documents should be noted (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html.

This document was prepared by the Open Geospatial Consortium (as OGC API — Features — Part 2: Coordinate Reference Systems by Reference) and drafted in accordance with its editorial rules. It was assigned to Technical Committee ISO/TC 211, *Geographic information/Geomatics*, and adopted under the “fast-track procedure”.

The main changes are as follows:

— addition of an Introduction;

— alignment of spellings with ISO spelling rules;

— renumbering and reordering of Clauses 2-4 in order to accommodate the fixed structure of ISO documents;

— set texts introduced in Clauses 2 and 3;

A list of all parts in the ISO 19168 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user’s national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

OGC API standards define modular API building blocks to spatially enable Web APIs in a consistent way. The OpenAPI specification is used to define the API building blocks.

The OGC API family of standards is organized by resource type. This document extends the fundamental API building blocks for interacting with features. The spatial data community uses the term 'feature' for things in the real world that are of interest.

For those not familiar with the term 'feature,' the explanations on Spatial Things, Features and Geometry in the W3C/OGC Spatial Data on the Web Best Practice document[6] provide more detail.

OGC API Features provides API building blocks to create, modify and query features on the Web. OGC API Features is comprised of multiple parts, each of them is a separate standard. This document extends the core capabilities specified in OGC API — Features — Part 1: Core (ISO 19168-1) with the ability to use coordinate reference system identifiers other than the defaults defined in the core.

By default, every API implementing this document will provide access to a single dataset. Rather than sharing the data as a complete dataset, the OGC API Features standards offer direct, fine-grained access to the data at the feature (object) level.

The API building blocks specified in this document are consistent with the architecture of the Web. In particular, the API design is guided by the IETF HTTP/HTTPS RFCs, the W3C Data on the Web Best Practices, the W3C/OGC Spatial Data on the Web Best Practices and the emerging OGC Web API Guidelines. A particular example is the use of the concepts of datasets and dataset distributions as defined in DCAT and used in schema.org.

A subset of the OGC API family of standards is expected to be published by ISO. For example, this document is published by ISO as ISO 19168-2. To reflect that only a subset of the OGC API standards will be published by ISO and to avoid using organization names in the titles of ISO standards, standards from the "OGC API" series are published by ISO as "Geospatial API," i.e. the title of this document in OGC is "OGC API — Features — Part 2: Coordinate Reference Systems by Reference" and the title in ISO is "Geographic Information — Geospatial API for Features — Part 2: Coordinate Reference Systems by Reference."

For simplicity, this document consistently uses:

— "OGC API" to refer to the family of standards for geospatial Web APIs that in ISO is published as "Geospatial API;"

— "OGC API — Features" to refer to the multipart standard for features that in ISO is published as ISO 19168 / "Geographic Information - Geospatial API for Features;"

— "OGC API — Features — Part 1: Core" to refer to the document that in ISO is published as ISO 19168-1 / "Geographic Information - Geospatial API for Features - Part 1: Core."

Geographic information – Geospatial API for features —

Part 2: Coordinate Reference Systems by Reference

1 Scope

This document specifies an extension to the Geospatial API for Features — Part 1: Core standard that defines the behaviour of a server that supports the ability to present geometry valued properties in a response document in one from a list of supported Coordinates Reference Systems (CRS).

Each supported CRS is specified by reference using a uniform resource identifier (URI).

This document specifies:

— how, for each offered feature collection, a server advertises the list of supported CRS identifiers;

— how the coordinates of geometry valued feature properties can be accessed in one of the supported CRSs;

— how features can be accessed from the server using a bounding box specified in one of the supported CRSs; and

— how a server can declare the CRS used to present feature resources.

2 Normative references

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

ISO 19168-1:2020, *Geographic information — Geospatial API for features — Part 1: Core*

3 Terms and definitions

For the purposes of this document, the terms and definition given in ISO19168-1 and the following apply.

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

NOTE The terms *feature* (3.4) and *feature collection* (3.5) and the acronym *CRS* (3.2) are duplicated here from ISO 19168-1.

3.1
coordinate
one of a sequence of numbers designating the position of a point

Note 1 to entry: In a spatial coordinate reference system, the coordinate numbers are qualified by units.

[SOURCE: ISO 19111:2019, 3.1.5]

3.2
coordinate reference system
CRS
coordinate system (3.3) that is related to an object by a datum

Note 1 to entry: Geodetic and vertical datums are referred to as reference frames.

Note 2 to entry: For geodetic and vertical reference frames, the object will be the Earth. In planetary application, geodetic and vertical reference frames may be applied to other celestial bodies.

[SOURCE: ISO 19111:2019, 3.1.9]

3.3
coordinate system
set of mathematical rules for specifying how *coordinates* (3.1) are to be assigned to points

[SOURCE: ISO 19111:2019, 3.1.11]

3.4
feature
abstraction of real world phenomena

Note 1 to entry: The explanations on Spatial Things, Features and Geometry in the W3C/OGC Spatial Data on the Web Best Practice document[6] provide more detail.

[SOURCE: ISO 19101-1:2014, 4.1.11, modified – Note 1 to entry has been added.]

3.5
feature collection
collection
set of *features* (3.4) from a *dataset* (ISO 19168-1, 3.1.1)

3.6
spatial feature collection
spatial collection
feature collection (3.5) that includes one or more *features* (3.4) that have properties whose value is a geometry

[SOURCE: ISO 19168-1:2020, 3.1.4]

4 Conformance

This document defines one requirements class, Coordinate Reference Systems by Reference. The standardization target is "Web APIs".

The URI of the associated conformance class is http://www.opengis.net/spec/ogcapi-features-2/1.0/conf/crs.

Conformance with this standard shall be checked using all the relevant tests specified in Annex A 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.

5 Conventions and background

See ISO 19168-1:2020, Clauses 5 and 6.

6 Requirements Class Coordinate Reference Systems by Reference

6.1 Overview

Requirements Class	
http://www.opengis.net/spec/ogcapi-features-2/1.0/req/crs	
Target type	Web API
Dependency	OGC API - Features - Part 1: Core, Requirements Class 'core'

The OGC API — Features — Part 1: Core standard defines support for only two coordinate reference systems:

— WGS 84 longitude, latitude;

— WGS 84 longitude, latitude, ellipsoidal height.

This extension defines the behaviour of a server that supports additional coordinate reference systems.

Requirement 1	**/req/crs/crs-uri**
Each CRS supported by a server shall be referenceable by a uniform resource identifier (i.e. a URI).	

Recommendation 1	**/rec/crs/crs-format-model**
Servers that implement this extension should be able to recognize and generate CRS identifiers with the following format model: `http://www.opengis.net/def/crs/authority/version/code` In this format model, the token `{authority}` is a placeholder for a value that designates to authority responsible for the definition of this CRS. Typical values include "EPSG" and "OGC". The token `{version}` is a placeholder for the specific version of the CRS definition or `0` for un-versioned CRS definitions. The token `{code}` is a placeholder for the authority's code for the CRS.	

For more information, see 6.2 in OGC Name Type Specification, Part 1.

Note that while the EPSG register itself is versioned, the registered items are not versioned and the "version" is always "0" in URIs of the authority "EPSG".

6.2 Discovery

6.2.1 CRS identifier list

Requirement 2	**/req/crs/fc-md-crs-list**
A	The `crs` property in the collection object of a spatial feature collection shall contain the identifiers for the list of CRSs supported by the server for that collection.
B	This list shall include the default(s) defined in OGC API - Features - Part 1: Core.

The list has to include the default CRS — that is the CRS used unless something else is explicitly requested — is defined in ISO 19168-1, *Geographic information — Geospatial API for features — Part 1: Core* as:

— http://www.opengis.net/def/crs/OGC/1.3/CRS84 (for coordinates without height);

— http://www.opengis.net/def/crs/OGC/0/CRS84h (for coordinates with ellipsoidal height).

6.2.2 Storage CRS

The storage CRS for a spatial feature collection is the CRS identifier that may be used to retrieve features from that collection without the need to apply a CRS transformation.

Note that coordinates referenced to a dynamic coordinate reference system are ambiguous if the coordinate epoch is unknown. It is therefore recommended to also provide the coordinate epoch when the storage CRS is dynamic, such as an ITRF realization or WGS 84. For more information on dynamic coordinate reference systems and coordinate epoch, please see ISO 19111, *Geographic information — Referencing by coordina*tes (same as OGC Abstract Specification Topic 2: Referencing by coordinates).

Requirement 3	**/req/crs/fc-md-storageCrs**
If all features in a spatial feature collection are stored using a particular CRS then the property `storageCrs` shall be specified in the collection object of the spatial feature collection to indicate the identifier for this storage CRS.	

Recommendation 2	**/rec/crs/fc-md-coordinateEpoch**
If the storage CRS of the spatial feature collection is a dynamic coordinate reference system, the property `storageCrsCoordinateEpoch` in the collection object of the spatial feature collection should provide the coordinate epoch of the coordinates.	

This document does not provide a mechanism to associate different coordinate epochs with feature geometries in a collection. If data with different coordinate epochs is merged in a collection, one option is to perform point motion operations (PMO) to convert all geometries to the same coordinate epoch. See ISO 19111, *Geographic information — Referencing by coordinates* (same as OGC Abstract Specification Topic 2: Referencing by coordinates), for more information.

Requirement 4	**/req/crs/fc-md-storageCrs-valid-value**
The value of the `storageCrs` property shall be one of the CRS identifiers from the list of supported CRS identifiers found in the collection object using the `crs` property.	

The following schema fragment extends the collection object to add the `storageCrs` and `storageCrsCoordinateEpoch` properties.

```
type: object
required:
  - id
  - links
properties:
  id:
    description: identifier of the collection used, for example, in URIs
    type: string
    example: address
  title:
    description: human readable title of the collection
    type: string
    example: address
  description:
    description: a description of the features in the collection
    type: string
    example: An address.
  links:
    type: array
    items:
      $ref: link.yaml
    example:
      - href: http://data.example.com/buildings
        rel: item
      - href: http://example.com/concepts/buildings.html
        rel: describedby
        type: text/html
  extent:
    $ref: extent.yaml
  itemType:
    description: indicator about the type of the items in the collection (the default
value is 'feature').
    type: string
    default: feature
  crs:
    description: the list of CRS identifiers supported by the service
    type: array
    items:
      type: string
    default:
      - http://www.opengis.net/def/crs/OGC/1.3/CRS84
    example:
      - http://www.opengis.net/def/crs/OGC/1.3/CRS84
      - http://www.opengis.net/def/crs/EPSG/0/4326
  storageCrs:
     description: the CRS identifier, from the list of supported CRS identifiers, that
may be used to retrieve features from a collection without the need to apply a CRS
transformation
     type: string
     format: uri
  storageCrsCoordinateEpoch:
     description: point in time at which coordinates in the spatial feature
collection are referenced to the dynamic coordinate reference system in
`storageCrs`, that may be used to retrieve features from a collection without
the need to apply a change of coordinate epoch. It is expressed as a decimal
year in the Gregorian calendar
     type: number
     example: '2017-03-25 in the Gregorian calendar is epoch 2017.23'
```

6.2.3 Global list of CRS identifiers

To prevent unnecessary duplication of lists of supported CRS identifiers in the collection object, a global list of supported CRS identifiers may be provided as part of the collections object.

This global list of CRS identifiers is not automatically inherited by each collection offered by the service. Rather the global list of CRS identifiers must be explicitly referenced in the `crs` property of the collection object using a JSON Pointer (RFC 6901).

Requirement 5	**/req/crs/fc-md-crs-list-global**
If the `crs` property in the collection object of a spatial feature collection includes a JSON Pointer to the global list of CRS identifiers (`#/crs`), then all CRS identifiers in the global list shall be valid for the referencing collection.	

Note that only a local JSON Pointer within the same document is supported.

The following schema fragment extends the collections object to add the `crs` property which contains the global list of CRS identifiers.

```
allOf:
- $ref:
'http://schemas.opengis.net/ogcapi/features/part1/1.0/openapi/schemas/collections.yaml'
- type: object
  properties:
    crs:
      description: a global list of CRS identifiers that are supported by spatial feature
collections offered by the service
      type: array
      items:
        type: string
        format: uri
```

The following example illustrates the use of a global list of CRS identifiers.

EXAMPLE Collections object containing a global list of CRS identifiers.

```
{
  "links": [
    { "href": "http://data.example.org/collections.json",
      "rel": "self", "type": "application/json", "title": "this document" },
    { "href": "http://data.example.org/collections.html",
      "rel": "alternate", "type": "text/html", "title": "this document as
HTML" },
    { "href": "http://schemas.example.org/1.0/buildings.xsd",
      "rel": "describedby", "type": "application/xml", "title": "GML
application schema for Acme Corporation building data" },
    { "href": "http://download.example.org/buildings.gpkg",
      "rel": "enclosure", "type": "application/geopackage+sqlite3", "title":
"Bulk download (GeoPackage)", "length": 472546 }
  ],
  "crs": [
     "http://www.opengis.net/def/crs/OGC/1.3/CRS84",
     "http://www.opengis.net/def/crs/EPSG/0/4326",
     "http://www.opengis.net/def/crs/EPSG/0/3857",
     "http://www.opengis.net/def/crs/EPSG/0/3395"
  ],
  "collections": [
     {
       "id": "bonn_buildings",
       "title": "Bonn Buildings",
       "description": "Buildings in the city of Bonn.",
       "extent": {
         "spatial": {
           "bbox": [ [ 7.01, 50.63, 7.22, 50.78 ] ]
         },
         "temporal": {
           "interval": [ [ "2010-02-15T12:34:56Z", null ] ]
         }
       },
       "links": [
         { "href":
"http://data.example.org/collections/bonn_buildings/items",
           "rel": "items", "type": "application/geo+json",
           "title": "Bonn Buildings" },
         { "href": "https://creativecommons.org/publicdomain/zero/1.0/",
           "rel": "license", "type": "text/html",
           "title": "CC0-1.0" },
         { "href":
"https://creativecommons.org/publicdomain/zero/1.0/rdf",
           "rel": "license", "type": "application/rdf+xml",
           "title": "CC0-1.0" }
       ],
       "crs": [
          "#/crs",
          "http://www.opengis.net/def/crs/EPSG/0/4258",
          "http://www.opengis.net/def/crs/EPSG/0/25831",
          "http://www.opengis.net/def/crs/EPSG/0/25832"
       ]
     },
     {
       "id": "tor_buildings",
       "title": "Toronto Buildings",
       "description": "Buildings in the city of Toronto.",
       "extent": {
         "spatial": {
           "bbox": [ [ -79.62, 43.58, -79.12, 43.87 ] ]
         },
         "temporal": {
           "interval": [ [ "2010-02-15T12:34:56Z", null ] ]
         }
       },
       "links": [
         { "href":
"http://data.example.org/collections/tor_buildings/items",
           "rel": "items", "type": "application/geo+json",
           "title": "Toronto Buildings" },
         { "href": "https://creativecommons.org/publicdomain/zero/1.0/",
            "rel": "license", "type": "text/html",
            "title": "CC0-1.0" },
          { "href":
"https://creativecommons.org/publicdomain/zero/1.0/rdf",
            "rel": "license", "type": "application/rdf+xml",
            "title": "CC0-1.0" }
        ],
        "crs": [
           "#/crs"
         ]
       },
       {
         "id": "dc_buildings",
         "title": "Washington DC Buildings",
         "description": "Buildings in the city of Washington DC.",
         "extent": {
           "spatial": {
            "bbox": [ [ -77.12, 38.80, -76.89, 39.01 ] ]
           },
           "temporal": {
             "interval": [ [ "2010-02-15T12:34:56Z", null ] ]
           }
         },
         "links": [
           { "href":
"http://data.example.org/collections/dc_buildings/items",
            "rel": "items", "type": "application/geo+json",
            "title": "DC Buildings" },
          { "href": "https://creativecommons.org/publicdomain/zero/1.0/",
            "rel": "license", "type": "text/html",
            "title": "CC0-1.0" },
          { "href":
"https://creativecommons.org/publicdomain/zero/1.0/rdf",
            "rel": "license", "type": "application/rdf+xml",
            "title": "CC0-1.0" }
          ],
          "crs": [
            "http://www.opengis.net/def/crs/OGC/1.3/CRS84"
        ]
      }
    ]
}
```

In the above example, the `bonn_buildings` collection is offered in all the CRSs specified in the global list plus three other CRSs.

The `tor_buildings` collection is offered in the CRSs specified in the global list.

The `dc_buildings` collection is only offered in the default CRS (i.e. WGS 84 longitude, latitude).

6.3 Query

6.3.1 Parameter bbox-crs

The `bbox-crs` parameter may be used to assert the CRS used for the coordinate values of the `bbox` parameter.

Requirement 6	**/req/crs/fc-bbox-crs-definition**

Each GET request on a 'features' resource shall support a query parameter `bbox-crs` with the following characteristics:
`name: bbox-crs`
`in: query`
`required: false`
`schema:`
`  type: string`
`  format: uri`
`style: form`
`explode: false`	

Requirement 7	**/req/crs/fc-bbox-crs-valid-value**
A	If the value of the `bbox-crs` parameter is not one of the CRS identifiers from the list of supported CRS identifiers, then the server shall respond with the HTTP status code 400.
B	The list of supported CRS identifiers is found in the collection object using the `crs` property.

As usual, it is good practice to include a message about the reason for the error in the response.

Requirement 8	**/req/crs/fc-bbox-crs-valid-default-value**
If the `bbox-crs` parameter is not specified then the coordinate values of the `bbox` parameter shall be assumed to be in the default CRS specified in OGC API - Feature - Part 1: Core; that is http://www.opengis.net/def/crs/OGC/1.3/CRS84 for coordinates without height and http://www.opengis.net/def/crs/OGC/0/CRS84h for coordinates with ellipsoidal height.	

Requirement 9	**/req/crs/fc-bbox-crs-action**
If the `bbox-crs` parameter is specified, then the values of the `bbox` parameter shall be assumed to be in the specified CRS and the server shall perform the necessary internal transformations to properly fetch data from within the specified bounding box.	

The following fragment illustrates the use of the `bbox-crs` parameter (reserved characters have to be encoded):

EXAMPLE Specifying a bounding box in one of the supported coordinate reference systems.

```
...&bbox=32507317%2C5224265%2C33427450%2C5603836&bbox-crs=http%3A%2F%2Fwww.opengis.net%2Fd
ef%2Fcrs%2FEPSG%2F0%2F25832
```

6.3.2 Parameter crs

Requirement 10	**/req/crs/fc-crs-definition**

Each GET request on a 'features' or 'feature' resource shall support a query parameter named `crs` with the following characteristics:
`name: crs`
`in: query`
`required: false`
`schema:`
`  type: string`
`  format: uri`
`style: form`
`explode: false`	

Requirement 11	**/req/crs/fc-crs-valid-value**
A	If the value of the `crs` parameter is not one of the CRS identifiers from the list of supported CRS identifiers, then the server shall respond with the HTTP status code 400.
B	The list of supported CRS identifiers is found in the collection object using the `crs` property.

As usual, it is good practice to include a message about the reason for the error in the response.

Requirement 12	**/req/crs/fc-crs-default-value**
If the `crs` parameter is not specified the geometry coordinates shall be presented in the default CRS specified in OGC API - Feature - Part 1: Core; that is http://www.opengis.net/def/crs/OGC/1.3/CRS84 for coordinates without height and http://www.opengis.net/def/crs/OGC/0/CRS84h for coordinates with ellipsoidal height.	

Requirement 13	**/req/crs/fc-crs-action**
If the `crs` parameter is specified, then the coordinates of all geometry-valued properties in the response document shall be presented in the requested CRS.	

Permission 1	**/per/crs/fc-crs-action**
Notwithstanding the requirement /req/crs/fc-crs-action, if the requested feature representation is subject to any limitations for supporting coordinate reference systems, the Web API may return a response with a status code `400`.	

For example, OGC KML only supports the default CRS (WGS84 longitude and latitude, optionally with ellipsoidal height).

The following fragment illustrates the use of the `crs` parameter:

EXAMPLE Retrieving features from a collection in one of the supported CRSs.

```
.../collections/buildings/items?crs=http%3A%2F%2Fwww.opengis.net%2Fdef%2Fcrs%2FEPSG%2F0
%2F26703&...
```

6.3.3 Output format considerations

6.3.3.1 General

ISO 19168-1, *Geographic information — Geospatial API for features — Part 1: Core* defines three conformance classes related to the output formats:

— GML/XML;

— GeoJSON/JSON;

— HTML.

6.3.3.2 Collections and Collection resource

This document specifies extensions to the Collection resource (the global list of coordinate reference systems) and the Collection resource (the storage CRS including the associated coordinate epoch).

How these extensions are reflected in each encoding is not fully specified by this standard, except for JSON-based or YAML-based encodings where the extensions are fully specified by the OpenAPI schema components.

For HTML, the requirement http://www.opengis.net/spec/ogcapi-features-1/1.0/req/html/content applies and the additional information has to be included in the body of the HTML document.

For XML, the content model of the of the complex types `core:CollectionsType` and `core:CollectionType` would have to be extended with additional information. This document does not specify the details for such extensions due to a lack of demand.

6.3.3.3 Features and Feature resource

GML has full CRS support and no further conventions are imposed by this document.

NOTE The CRS model in GML is based on ISO 19111:2007, but GML geometries reference CRSs by their URI identifier in the srsName attribute. These can resolve to a CRS that is defined based on the CRS model specified by ISO 19111:2019 (same as OGC Abstract Specification Topic 2: Referencing by coordinates), or a future edition.

HTML does not have any provisions for spatial geometries and coordinate reference systems. However, note that schema.org that is embedded in HTML only supports WGS 84 in the axis order latitude/longitude, so any coordinates in schema.org markup will have to be in that coordinate reference system, independent of the requested coordinate reference system.

GeoJSON normatively supports WGS 84 (without height: CRS84; with ellipsoidal height: CRS84h), but the "prior arrangement" provision allows other coordinate systems to be used.

Requirement 14	**/req/crs/geojson**
Servers that implement this extension plus the GeoJSON requirements class and clients that use this extension shall be subject to the prior arrangement provision in the second paragraph of Clause 4 of the GeoJSON standard.	

An explicit request by a client with a query parameter `crs` establishes a prior arrangement. It is the responsibility of the client that submits the request to handle the coordinates in the response correctly. In particular, clients should not make the GeoJSON document available to others unless they are aware of the prior arrangement.

This standard does not specify any standardized approach to encoding coordinate reference system information in a GeoJSON document.

The first paragraph of Clause 4 in GeoJSON also states: *"An OPTIONAL third-position element SHALL be the height in meters above or below the WGS 84 reference ellipsoid. In the absence of elevation values, applications sensitive to height or depth SHOULD interpret positions as being at local ground or sea level."*

If the requested coordinate reference system includes the vertical axis, the third-position element has to be interpreted according to that coordinate reference system, not as if it would be relative to the WGS 84 reference ellipsoid.

6.3.4 Coordinate reference system information independent of the feature encoding

Because of the inconsistent provision of CRS metadata in geospatial encodings and the continued confusion caused by the axis order of coordinates, this document defines a mechanism for a server to clearly and unambiguously assert the CRS being used in a response document independent of the requested output format.

Requirement 15	**/req/crs/ogc-crs-header**
An HTTP header named `Content-Crs` shall be used to assert the coordinate reference system used in the body of a response.	

Requirement 16	**/req/crs/ogc-crs-header-value**

The value of the `Content-Crs` header shall be a URI identifying the coordinate reference system used in the response document according to the following grammar for `CRS-header`.
`CRS-header = "Content-Crs" ":" CRS-value`
`CRS-value = "<" URI-reference ">"`	

NOTE The header is consistent with the draft "content negotiation by coordinate reference system" specification.

The following example illustrates the `Content-Crs` header in a response.

EXAMPLE HTTP header declaring the CRS used in the body of the response.

```
$ curl -i
"https://example.com/api/v1/collections/poi/items/1?crs=http%3A%2F%2Fwww.opengis.
net%2Fdef%2Fcrs%2FEPSG%2F0%2F3395"

  HTTP/1.1 200 OK
  Date: Sun, 24 May 2020 15:30:56 GMT
  Content-Type: application/geo+json
  Content-Language: en
  Content-Crs: <http://www.opengis.net/def/crs/EPSG/0/3395>
  Link:
<https://example.com/api/v1/collections/poi/items/1?crs=http%3A%2F%2Fwww.opengis.
net%2Fdef%2Fcrs%2FEPSG%2F0%2F3395&f=json>; rel="self"; title="This
document"; type="application/geo+json"
  Link:
<https://example.com/api/v1/collections/poi/items/1?crs=http%3A%2F%2Fwww.opengis.
net%2Fdef%2Fcrs%2FEPSG%2F0%2F3395&f=html>; rel="alternate"; title="This
document as HTML"; type="text/html"
  Link: <https://example.com/api/v1/collections/poi>; rel="collection"; title="The
collection the feature belongs to"
  Vary: Accept-Language,Accept-Encoding
  Content-Length: 1064

  ...
```

Annex A
(normative)

Abstract Test Suite

Conformance Class	
http://www.opengis.net/spec/ogcapi-features-2/1.0/conf/crs	
Target type	Web API
Requirements class	Requirements Class 'Coordinate Reference Systems by Reference'
Dependency	OGC API - Features - Part 1: Core, Conformance Class 'core'

A.1 Discovery

Abstract Test 1	**/conf/crs/crs-uri**
Test purpose	Verify that each CRS identifier is a valid value
Requirement	req/crs/crs-uri, /req/crs/fc-md-crs-list A, /req/crs/fc-md-storageCrs, /req/crs/fc-md-crs-list-global
Test method	For each string value in a `crs` or `storageCrs` property in the collections and collection objects in the paths /`collections` and /`collections`/`{collectionId}`, validate that the string conforms to the generic URI syntax as specified by RFC 3986, Clause 3. In addition, accept a single value of `#/crs` in each collection object at path /`collections`, if the collections object has a `crs` property. 1. For http-URIs (starting with `http:`) validate that the string conforms to the syntax specified by RFC 7230, 2.7.1. 2. For https-URIs (starting with `https:`) validate that the string conforms to the syntax specified by RFC 7230, 2.7.2. 3. For URNs (starting with `urn:`) validate that the string conforms to the syntax specified by RFC 8141, Clause 2. 4. For OGC URNs (starting with `urn:ogc:def:crs:`) and OGC http-URIs (starting with `http://www.opengis.net/def/crs/`) validate that the string conforms to the syntax specified by OGC Name Type Specification - definitions - part 1 – basic name.

Abstract Test 2	**/conf/crs/default-crs**
Test purpose	Verify that the list of supported CRSs includes the default CRS.
Requirement	/req/crs/fc-md-crs-list B
Test method	For each string value in a `crs` property in a collection object (for each path /`collections` and /`collections`/`{collectionId}`) validate that either `http://www.opengis.net/def/crs/OGC/1.3/CRS84` or `http://www.opengis.net/def/crs/OGC/1.3/CRS84h` is included in the array, if the collection has a spatial extent, i.e., is a spatial feature collection.

Abstract Test 3	**/conf/crs/storageCrs**
Test purpose	Verify that the storage CRS identifier is a valid value.
Requirement	/req/crs/fc-md-storageCrs-valid-value

Abstract Test 3	**/conf/crs/storageCrs**
Test method	For each collection object that includes a `storageCrs` property in the paths /`collections` and /`collections`/`{collectionId}`, validate that the string is also found in the `crs` property of the collection or, in case the `crs` property includes a value `#/crs`, in the global list of CRSs.

A.2 Query

A.2.1 Parameter crs

Abstract Test 4	**/conf/crs/crs-parameter**
Test purpose	Verify that the parameter `crs` has been implemented correctly.
Requirement	/req/crs/fc-crs-definition, /req/crs/fc-crs-valid-value B, /req/crs/ogc-crs-header, /req/crs/ogc-crs-header-value, /req/crs/geojson
Test method	For — each spatial feature collection `collectionId`, — every GML or GeoJSON feature representation supported by the Web API, and — every CRS supported for the collection (every CRS listed in the `crs` property of the collection plus those in the global CRS list, if `#/crs` is included in the `crs` property) send a request with CRS identifier in the parameter `crs` to — `/collections/{collectionId}/items` and — `/collections/{collectionId}/items/{featureId}` (with a valid `featureId` for the collection). Verify that — every response is a valid Features or Feature response, — has the status code `200` and — includes a `Content-Crs` http header with the value of the requested CRS identifier.

Abstract Test 5	**/conf/crs/crs-parameter-invalid**
Test purpose	Verify that invalid values in the parameter `crs` are reported.
Requirement	/req/crs/fc-crs-valid-value
Test method	For — each spatial feature collection `collectionId` send a request with an unsupported CRS identifier in the parameter `crs` to — `/collections/{collectionId}/items` and — `/collections/{collectionId}/items/{featureId}` (with a valid `featureId` for the collection). Verify that the response has status code `400`. Unsupported CRS identifiers are all strings not included in the `crs` property of the collection and also not included in the global CRS list, if `#/crs` is included in the `crs` property.

Abstract Test 6	**/conf/crs/crs-parameter-default**
Test purpose	Verify that the default value for parameter `crs` has been implemented correctly.
Requirement	/req/crs/fc-crs-default-value, /req/crs/ogc-crs-header, /req/crs/ogc-crs-header-value

Abstract Test 6	**/conf/crs/crs-parameter-default**
Test method	For each spatial feature collection, send a request without the `crs` parameter and verify that the response includes a `Content-Crs` http header with the value of the default CRS identifier of the collection.

Abstract Test 7	**/conf/crs/crs-parameter-transform**
Test purpose	Verify that the geometries are transformed.
Requirement	/req/crs/fc-crs-action
Test method	For every CRS identifier advertised by the Web API that is known to the test engine and for which the test engine can convert geometries between the CRS and the default CRS of the Web API ("known CRS") execute the following test. Skip the test for unknown CRSs. 1) For each spatial feature collection `collectionId`, send a request with the parameter crs to `/collections/{collectionId}/items` and `/collections/{collectionId}/items/{featureId}` (with a valid `featureId` for the collection) for every known CRS listed. In addition, send the same request, but without the `crs` parameter. 2) Convert the response for the known CRS to the default CRS and verify that the responses match. Due to the use of different coordinate conversions in the test engine and by the API, there will not be an exact match and the test engine will have to allow for reasonable differences when assessing whether the geometries match.

A.2.2 Parameter bbox-crs

Abstract Test 8	**/conf/crs/bbox-crs-parameter**
Test purpose	Verify that the parameter `bbox-crs` has been implemented correctly
Requirement	/req/crs/fc-bbox-crs-definition, /req/crs/fc-bbox-crs-action
Test method	For every CRS identifier advertised by the Web API that is known to the test engine and for which the test engine can convert geometries between the CRS and the default CRS of the Web API ("known CRS") execute the following test. Skip the test for unknown CRSs. 1) For each spatial feature collection `collectionId` and every GML or GeoJSON feature representation supported by the Web API, send a request with the parameters `bbox` and `bbox-crs` to `/collections/{collectionId}/items` for every known CRS. Use a `bbox` value in the spatial extent of the collection, converted to the known CRS. Send the same request, but with no `bbox-crs` parameter and a `bbox` value in the default CRS. Do not include a `crs` parameter in the requests. Verify that the responses include the same features.

Abstract Test 9	**/conf/crs/bbox-crs-parameter-invalid**
Test Purpose	Verify that the parameter `bbox-crs` has been implemented correctly
Requirement	/req/crs/fc-bbox-crs-valid-value
Test Method	For each spatial feature collection `collectionId` , send a request with the parameters `bbox` and `bbox-crs` to `/collections/{collectionId}/items` with a value for `bbox-crs` that is not included in the list of CRSs and verify that the response has status code `400`.

Abstract Test 10	**/conf/crs/bbox-crs-parameter-default**
Test Purpose	Verify that the parameter `bbox-crs` has been implemented correctly
Requirement	/req/crs/fc-bbox-crs-default-value

Abstract Test 10	**/conf/crs/bbox-crs-parameter-default**
Test Method	For each spatial feature collection `collectionId` and every GML or GeoJSON feature representation supported by the Web API, send a request with the parameters `bbox` and `bbox-crs` to `/collections/{collectionId}/items` for the default CRS of the collection. Use a `bbox` value in the spatial extent of the collection. Send the same request, but with no `bbox-crs` parameter. Do not include a `crs` parameter in the requests. Verify that the responses include the same features.

Bibliography

[1] OpenAPI Initiative (OAI). *OpenAPI Specification 3.0* [online]. 2020 [viewed 2020-03-16]. The latest patch version at the time of publication of this standard was 3.0.3, available at https://spec.openapis.org/oas/v3.0.3

[2] Open Geospatial Consortium (OGC). OGC 08-038r7: *Revision to Axis Order Policy and Recommendations* [online]. Edited byReed C., 2017 [viewed 2020-05-24]. Available at https://portal.opengeospatial.org/files/?artifact_id=76024

[3] Open Geospatial Consortium (OGC). OGC 10-100r3: *Geography Markup Language (GML) Simple Features Profile* [online]. Edited byvan den Brink L., Portele C., Vretanos P., 2012 [viewed 2020-03-16]. Available at http://portal.opengeospatial.org/files/?artifact_id=42729

[4] Internet Engineering Task Force (IETF) RFC 7946: *The GeoJSON Format* [online]. Edited by H. Butler, M. Daly, A. Doyle, S. Gillies, S. Hagen, T. Schaub. 2016 [viewed 2020-03-16]. Available at https://tools.ietf.org/rfc/rfc7946.txt

[5] WHATWG *HTML*, Living Standard [online, viewed 2020-03-16]. Available at https://html.spec.whatwg.org/

[6] W3C, Open Geospatial Consortium (OGC). *Spatial Data on the Web Best Practices* [online]. Edited by J. Tandy, L. van den Brink, P. Barnaghi. Available at https://www.w3.org/TR/sdw-bp

[7] ISO 19111:2019, *Geographic information — Referencing by coordinates*

[8] ISO 19111:2007, *Geographic information — Spatial referencing by coordinates*

ICS 35.240.70

Price based on 16 pages