deegree OGC API - Features Implementation

lat/lon Gesellschaft für raumbezogene Informationssysteme mbH

Version 2.0.2, 13.12.2025

Table of Contents

Chapter 1. Preambel

List of Authors: Jens Fitzke, Torsten Friebe, Lyn Goltz, Sabine Schmitz, Dirk Stenger.

lat/lon - Gesellschaft für raumbezogene Informationssysteme mbH
info@lat-lon.de, https://www.lat-lon.de/
Im Ellig 1, D-53343 Wachtberg, Germany

Chapter 2. Introduction

The deegree OGC API - Features implementation is compliant to the latest OGC standards called OGC APIs. deegree ogcapi provides a server implementation that has been developed based on deegree's existing workspace concept using the API of deegree webservices. Compared to deegree webservices where spatial data is provided via services such as WFS and WMS with deegree ogcapi spatial data can be served with a resource-oriented approach.

2.1. OGC API family

The OGC API family of standards are being developed to make it easy for anyone to provide geospatial data to the web. These standards build upon the legacy of the OGC Web Service standards (WMS, WFS, WCS, WPS, etc.), but define resource-centric APIs that take advantage of modern web development practices.

Read more at www.ogcapi.org about the new standards and their goals. The OGC E-learning tutorial provides more information about the new standards and explains the basic concepts.

2.2. Features

deegree ogcapi supports the following standards:

- OGC API - Features - Part 1: Core
- OGC API - Features - Part 2: CRS by Reference
- OGC API - Features - Part 3: Filtering (Partly) with support for CQL2 intersects function.

The deegree ogcapi provides implementations of representations for spatial data using encodings such as:

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

The Web API is provided as an Open API 3.0 document implementing the requirements class:

- OpenAPI Specification 3.0.

deegree ogcapi was built on top of the robust API and configuration concept of deegree webservices. Notable features:

- easy to install: simple deployment in every Java Servlet container
- easy to configure: uses the deegree workspace configuration concept
- connects to several data sources such as files, and databases
- serves a rich HTML output format including a map component based on OpenLayers

- links to metadata in different formats
- supports data types such as arrays and structured objects
- supports a bulk download for datasets as specified by requirements class "INSPIRE-bulk-download"

2.3. Quick Start

1. Install the deegree ogcapi webapp on your preferred Java Servlet container (the runtime environment). See section Installation for more information.
2. Create the necessary deegree configuration files stored within a deegree workspace with at least one datasource. Which files are required is described in section Configuration.
3. Install deegree ogcapi webapp and start the runtime environment. This is described in section Start the webapp.
4. Start a browser and open the OpenAPI specification. How to use the Web API document is explained in section OpenAPI document.
5. Browse the content using the HTML interface. How to access the data is described in section Using the HTML interface.

Chapter 3. Terms and Definitions

For simplicity, this document consistently uses:

- "OGC API" to refer to the family of standards for geospatial Web APIs specified by the OGC.
- "OAPI-F" to refer to the standard OGC API - Features, including Part: 1, 2 , and 3.
- "deegree ogcapi" to refer to the implementation of OGC API standards based on deegree core API.

Furthermore the following terms are used in this document:

3.1. dataset

Is a collection of data. In the context of deegree ogcapi a dataset is the set of feature types provided by a feature store. It is identified by it's *{datasetId}*.

3.2. datasource

Is a source containing data retrieved from a file or a database. In the context of deegree ogcapi a datasource can be a feature store implementation. Which feature store implementations are supported is explained in section Feature store configuration.

3.3. feature

Is an abstraction of real world phenomena. In the context of deegree ogcapi it is a spatial entity stored in a datasource. It is identified by it's *{featureId}*.

3.4. feature collection

Is a set of features from a dataset. It is identified by it's *{collectionId}*.

3.5. feature store

Is a ressource within the deegree workspace providing access to stored features. In the context of deegree ogcapi a feature store provides access to datasource such as SHAPE file or a database.

3.6. metadata

Is a description of the dataset. In the context of deegree ogcapi metadata can be linked per dataset.

3.7. WebAPI

An API using an architectural style that is based upon open standards and best practices such as OpenAPI Specification and it's implementation Swagger.

3.8. webapp

Is a deployable instance of a web application. In the context of deegree ogcapi it is a Web application ARchive (WAR) containing all resources that together constitute the deegree ogcapi application.

3.9. workspace

Is a directory structure containing configuration files. Most configuration files are in XML format. Read more in Configuration how to configure deegree ogcapi.

Chapter 4. Installation

There are many ways to install deegree ogcapi. This section will describe the various installation paths available.

4.1. Requirements

deegree ogcapi works on every Java Servlet container implementing the Java Servlet API 6.0 using an OpenJDK or Oracle JDK 17 or higher. The JRE and Java Servlet container constitute the runtime environment. The installation of these components must be done prior of this installation.

The server hosting the service should have at least 4 GB of free memory and more than 2 CPU cores available.

NOTE AdoptOpenJDK 17 with Apache Tomcat 10 is the recommended runtime environment.

4.2. Download

The deegree OGC API webapp is provided as a web application archive (WAR) file and release versions are available on the deegree OGC API GitHub page.

Choose either the ***deegree-ogcapi-webapp-postgres.war*** for PostgreSQL/PostGIS or the ***deegree-ogcapi-webapp-oracle.war*** for Oracle databases. Both webapps contain all required libraries and feature store implementations to access file-based feature stores (see Feature store configuration for more information which feature stores are supported). Download the WAR file and store it in the local file system of the server.

4.3. Deploy the webapp

Move the WAR file to the deployment folder of the runtime environment. For Apache Tomcat the folder is `$CATALINA_HOME/webapps`.

NOTE Within this document the context path *deegree-ogcapi* for the deegree ogcapi webapp is used. Rename the WAR file to *deegree-ogcapi.war* to follow this convention.

NOTE Deploying one single webapp per runtime instance is recommended. See FAQ for more information about deployment of deegree ogcapi.

4.4. Start the webapp

To start the webapp the runtime environment needs to be started. For Apache Tomcat use the start script `$CATALINA/bin/startup.sh`.

4.5. Stop the webapp

To stop the webapp the runtime environment needs to be stopped. For Apache Tomcat use the start script `$CATALINA/bin/shutdown.sh`.

4.6. Uninstallation

To uninstall the webapp:

1. Stop the runtime environment (if it is running).
2. Delete the webapp and the directory in which the deegree ogcapi webapp is installed.

4.7. Docker

Docker images with deegree OGC API are available on Docker Hub. This requires the installation of Docker on the server. Check the Docker documentation for requirements and installation instructions. The docker image provides all software components and no additional software installations are necessary.

Get the docker image with:

```
docker pull deegree/deegree-ogcapi:latest
```

To start a Docker container with the name *ogcapi* on port *8080* run the following command:

```
docker run --name ogcapi -d -p 8080:8080 deegree/deegree-ogcapi:latest
```

See the Docker CLI documentation for more information how to connect a container to a network, mount a volume into the container, or set environment variables.

4.8. Supported Browser

deegree ogcapi has been tested across a wide range of browsers, and operating systems.

The following browsers are supported:

- Mozilla Firefox (80.0+), Google Chrome (85.0+), Microsoft Edge (79.0+)

If you are **not** using one of the browser above, you should be able to access all resources provided by deegree ogcapi and use the HTML pages, but it might not display the site as designed, nor provide you with the best user experience.

Some browsers, particularly earlier versions, either do not or only partly support W3C standards and JavaScript. These browsers might not display the HTML pages properly. See the list of supported browsers by the Vue.js framework for more information.

Chapter 5. Configuration

This section describes how to configure the deegree ogcapi webapp.

IMPORTANT deegree ogcapi is under development, and new configuration options may be added in future releases. For the latest configuration options please consult the example workspace provided with the release version!

5.1. ogcapi workspace

The deegree workspace is the modular, resource-oriented and extensible configuration concept used by deegree. The deegree ogcapi workspace is an extension of the standard deegree workspace. A deegree ogcapi workspace can contain additional configuration files specific to deegree ogcapi.

The directory structure and related files are described by the following example. This example consists out of two datasets called *trees* and *streets* which data is stored within a PostgreSQL/PostGIS database. The following directory structure shows the files used in this example:

ogcapi-workspace/

```
├──── config.apikey                 ①
├──── webapps.properties            ②
└──── ogcapi-workspace              ③
    ├──── datasources
    │     └──── feature              ④
    │         ├──── streets.xml      ⑤
    │         └──── trees.xml        ⑥
    ├──── html                      ⑦
    │     ├──── htmlview.xml         ⑧
    │     ├──── streetsview.xml      ⑨
    │     ├──── treesview.xml        ⑩
    │     └──── default-style.css    ⑪
    ├──── jdbc
    │     └──── postgres-db.xml      ⑫
    ├──── ogcapi                    ⑬
    │     ├──── datasets.xml         ⑭
    │     ├──── streets.xml          ⑮
    │     └──── trees.xml            ⑯
    └──── services
        ├──── streets_metadata.xml  ⑰
        └──── trees_metadata.xml    ⑱
```

① config file with APIKEY required to access REST-API (optional)

② config file maps deegree ogcapi workspace to deegree ogcapi webapp (mandatory)

③ the workspace root directory (mandatory)

④ subdirectory must contain at least one feature store configuration

⑤ a feature store with id *streets*

⑥ a feature store with id *trees*

⑦ subdirectory with the configuration of the HTML encoding (optional)

⑧ global HTML encoding configuration (optional)

⑨ HTML encoding configuration for the dataset streets (optional)

⑩ HTML encoding configuration for the dataset trees (optional)

⑪ CSS file (optional)

⑫ configuration file required for database feature stores defining a JDBC connection (optional)

⑬ subdirectory must contain at least one dataset configuration (mandatory)

⑭ global dataset configuration (optional)

⑮ dataset configuration for the dataset streets (mandatory)

⑯ dataset configuration for the dataset trees (mandatory)

⑰ metadata configuration for dataset streets (optional)

⑱ metadata configuration for dataset trees (optional)

IMPORTANT The path to the deegree ogcapi workspace directory can be set by the environment variable `DEEGREE_WORKSPACE_ROOT`. The deegree ogcapi workspace may contain service configuration files for deegree webservices such as WFS and WMS but those services won't be available with deegree ogcapi webapp!

More information about deegree's workspace concept is available in the deegree webservices handbook. There you will find more information about the feature store configuration and the JDBC connection configuration.

5.2. Configuration files

The deegree ogcapi workspace adds the following directories to a standard deegree workspace:

- *ogcapi/*: subdirectory with **dataset configuration** files (required)
- *html/*: subdirectory with **HTML encoding configuration** files (optional)

A deegree ogcapi workspace uses the following directories of a standard deegree workspace:

- *datasources/feature/*: feature store configuration files
- *jdbc/*: JDBC connection configuration files
- *services/*: metadata configuration files
- *config.apikey*: security configuration file in the root directory of the workspace

The following chapters describe how to setup a deegree ogcapi workspace by the given example.

5.2.1. Datasets configuration

To provide general information about the datasets provider the following configuration file can be used:

ogcapi/datasets.xml

```
<Datasets xmlns="http://www.deegree.org/ogcapi/datasets"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://www.deegree.org/ogcapi/datasets
https://schemas.deegree.org/ogcapi/1.3/datasets.xsd">
  <Title>Datasets Title</Title>
  <Description>Datasets Description</Description> ①
  <Contact>
    <Name>Contact Name</Name>
    <Url>https://www.deegree.org</Url>
    <EMail>info@deegree.org</EMail>
  </Contact>
</Datasets>
```

① Supports CDATA section with HTML elements

NOTE The file *datasets.xml* shall be stored in the subdirectory *ogcapi/*. The file is optional.

This configuration file can contain the following elements:

Option	Cardinality	Value	Description
Title	0..1	String	Title
Description	0..1	String	Description
Contact	0..1	Complex	Contact configuration

The element `<Contact/>` has the following subelements:

Option	Cardinality	Value	Description
Name	0..1	String	Name of the dataset provider
Url	0..1	String	URL of the dataset provider
Email	0..1	String	Email of the dataset provider

NOTE The content of this file is returned under the resource */datasets*.

5.2.2. Dataset configuration

Each dataset is configured in a separate file. The following example shows a minimal configuration for a dataset called "streets". The filename defines the *{datasetId}*.

ogcapi/streets.xml

```
<deegreeOAF xmlns="http://www.deegree.org/ogcapi/features"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://www.deegree.org/ogcapi/features
https://schemas.deegree.org/ogcapi/1.3/features.xsd">

  <FeatureStoreId>streets</FeatureStoreId>  ①

  <QueryCRS>http://www.opengis.net/def/crs/OGC/1.3/CRS84</QueryCRS>  ②
  <QueryCRS>http://www.opengis.net/def/crs/EPSG/0/4326</QueryCRS>  ③
  <QueryCRS>http://www.opengis.net/def/crs/EPSG/0/25832</QueryCRS> ③

  <HtmlViewId>streetview</HtmlViewId>  ④

</deegreeOAF>
```

① identifier of the feature store configuration, links to file *datasources/feature/streets.xml*

② mandatory CRS, first CRS element must be `http://www.opengis.net/def/crs/OGC/1.3/CRS84` as specified in OGC API Features Core specification

③ additional CRS, to retrieve data in the given CRS the optional query parameter `{crs}` needs to be used, see section Using query parameters for more information

④ identifier of the HTML encoding configuration, links to file *html/streetsview.xml*

The next example shows a complete configuration for a dataset called "trees" with all options available.

ogcapi/trees.xml

```
<deegreeOAF xmlns="http://www.deegree.org/ogcapi/features"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://www.deegree.org/ogcapi/features
https://schemas.deegree.org/ogcapi/1.3/features.xsd">

  <FeatureStoreId>trees</FeatureStoreId>  ①

  <UseExistingGMLSchema>true</UseExistingGMLSchema> ②

  <QueryCRS>http://www.opengis.net/def/crs/OGC/1.3/CRS84</QueryCRS>  ③
  <QueryCRS>http://www.opengis.net/def/crs/EPSG/0/4326</QueryCRS>  ④

  <DateTimeProperties>
    <DateTimeProperty> ⑤
      <FeatureTypeName xmlns:app="http://www.deegree.org/app">
app:trees</FeatureTypeName>
      <PropertyName xmlns:app="http://www.deegree.org/app">app:seedyear</PropertyName>
    </DateTimeProperty>
  </DateTimeProperties>

  <GeometryProperties> ⑥
    <GeometryProperty skipExportAsWkt="true">
      <FeatureTypeName xmlns:app="http://www.deegree.org/app">
app:trees</FeatureTypeName>
      <PropertyName xmlns:app="http://deegree.org/app">app:geom1</PropertyName>
    </GeometryProperty>
  </GeometryProperties>

  <HtmlViewId>treesview</HtmlViewId>  ⑦

  <Metadata>
    <ProviderLicense> ⑧
      <Name>Provider license</Name>
      <Description>no limitations to public access</Description>
    </ProviderLicense>
    <DatasetLicense>  ⑨
      <Name>Dataset license</Name>
      <Url>https://www.govdata.de/dl-de/by-2-0</Url>
    </DatasetLicense>
    <DatasetCreator> ⑩
      <Name>Dataset Creator Name</Name>
      <Url>http://deegree.org</Url>
      <EMail>info@deegree.org</EMail>
    </DatasetCreator>
    <MetadataURL format="application/xml">http://example.metadata.org?service=CSW
&amp;request=GetRecordById&amp;version=2.0.2&amp;id=1234</MetadataURL> ⑪
    <MetadataURL format="text/html">
http://example.metadata.org/path_to_html/1234</MetadataURL> ⑫
  </Metadata>

  <ConfigureCollection id="TreeFeature"> ⑬
    <AddLink href="https://inspire.ec.europa.eu/featureconcept/" rel="tag" type=
"text/html" title="Feature concept for trees"/>
  </ConfigureCollection>

  <ConfigureCollections>
    <AddLink href="https://github.com/INSPIRE-MIF/" rel="describedby" type="text/html"
title="Encoding example"/> ⑭
    <AddLink href="https://schemas.deegree.org/trees.xsd" rel="describedby" type=
"application/xml" title="GML application schema for trees"/> ⑮
  </ConfigureCollections>

</deegreeOAF>
```

① identifier of the feature store configuration, links to file *datasources/feature/trees.xml*

② activates the resource to serve the GML schema, available for schema-driven SQLFeatureStore and MemoryFeatureStore. If not provided or set to `false` the schema file is generated by deegree.

③ mandatory CRS, first CRS must be http://www.opengis.net/def/crs/OGC/1.3/CRS84 as specified in OGC API Features Core specification

④ additional CRS, to retrieve data in the given CRS the optional query parameter `{crs}` needs to be used, see section Using query parameters for more information.

⑤ DateTime property defines a property *app:seedyear* of the feature type *app:trees* as a datetime property

⑥ Geometry property defines a geometry property *app:geom1* of the feature type *app:trees* as the property to export as geometry in GeoJSON. Only required if the featureType contains multiple geometries. If the attribute *skipExportAsWkt* is set to false, other geometries are exported as WKT in GeoJSON.

⑦ identifier of the HTML encoding configuration, links to file *html/treesview.xml*

⑧ provider license applicable to the service provider with description element

⑨ dataset license applicable to the dataset using link to license element

⑩ dataset provider contact details

⑪ metadata link in format `application/xml` for the dataset (optional)

⑫ metadata link in format `text/html` for the dataset (optional)

⑬ configure additional links for an individual collection. In the example, an additional link to the INSPIRE feature concept for the collection is provided (optional) (required by INSPIRE)

⑭ configure additional links for all collections. In the example, an additional link to the alternative encoding description for collections is provided (optional) (recommended by INSPIRE)

⑮ configure additional links for all collections. In the example, an additional link to the GML application schema is provided (optional) (used by tools such as QGIS and GDAL)

NOTE The dataset configuration file must be stored in the subdirectory *ogcapi/*. The file is mandatory.

This configuration file can contain the following elements:

Option	Cardinality	Value	Description
FeatureStoreId	0..n	String	Identifier of a feature store, see Feature store configuration which implementations are supported. This identifier also defines the *{collectionId}*
QueryCRS	0..n	String	The CRS codes supported, `CRS84` must be provided as the first element
DateTimeProperties	0..1	Complex	Configuration of date and time properties, see parameter datetime in the OGC API specification for more information
HtmlViewId	0..1	String	Identifier of the HTML encoding configuration, see HTML encoding configuration for more information

Option	Cardinality	Value	Description
Metadata	0..1	Complex	Configuration of the dataset metadata provided on the dataset's landing page
ConfigureCollection	0..1	Complex	Custom configuration for an individual collection
ConfigureCollections	0..1	Complex	Custom configuration for all collections

The element `<DateTimeProperties/>` can contain multiple elements of `<DateTimeProperty/>` which has the following subelements:

Option	Cardinality	Value	Description
FeatureTypeName	0..1	String	QName of the feature type
PropertyName	0..1	String	QName of the property

The element `<Metadata/>` has the following subelements:

Option	Cardinality	Value	Description
ProviderLicense	0..1	Complex	License of the dataset provider
DatasetLicense	0..1	Complex	License of the dataset
DatasetCreator	0..1	Complex	Contact details of the dataset creator
MetadataURL	0..n	URL	URL of the metadata record describing the dataset, use the attribute `format` to link HTML or XML representation. Use this link to a metadata record when you have a metadata record describing all containing feature collections. Otherwise use the element `<Dataset>` as described in the next chapter Metadata configuration.

The element `<ConfigureCollection/>` has the following subelement:

Option	Cardinality	Value	Description
AddLink	0..1	Complex	URL of additional link

The element `<ConfigureCollections/>` has the following subelement:

Option	Cardinality	Value	Description
AddLink	0..1	Complex	URL of additional link

The elements `<ProviderLicense/>` and `<DatasetLicense/>` can have either a `<Name/>` and `<Description/>` element or a `<Name/>` and `<URL/>` element. The `<URL/>` can have an optional attribute `format` specifying the media type such as `application/xml` (default is `text/html`). Same applies to the element <MetadataURL/>.

The <AddLink/> elements in <ConfigureCollection/> and <ConfigureCollections/> have href, rel, type and title parameters.

See the following section Metadata configuration for more configuration options for metadata.

NOTE The content of this file is returned under the resource */datasets/{datasetId}*. This resource per dataset is called landing page. Furthermore the content of this file is provided unter the resource */datasets/{datasetId}/api*.

5.2.3. Metadata configuration

The deegree service metadata configuration can be defined for each dataset. Use a file name ending with *{datasetId}_metadata.xml* to define the service metadata per dataset. Use the dataset identifier *{datasetId}* as a prefix. For example if you have a dataset configured in *streets.xml* the related metadata file has the file name *streets_metadata.xml*.

The following excerpt of the *streets_metadata.xml* shows which options are available:

services/streets_metadata.xml

```
<deegreeServicesMetadata xmlns="http://www.deegree.org/services/metadata"
                         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                         xsi:schemaLocation="http://www.deegree.org/services/metadata
https://schemas.deegree.org/3.5/services/metadata/metadata.xsd">

  <ServiceIdentification> ①
    <Title>deegree OGC API - Features</Title>
    <Abstract>Streets of the city of Hamburg</Abstract> ②
  </ServiceIdentification>

  <DatasetMetadata>
    <MetadataUrlTemplate>http://example.metadata.org/services/csw?service=CSW
&amp;request=GetRecordById&amp;version=2.0.2&amp;id=${metadataSetId}</MetadataUrlTempl
ate> ③
    <MetadataUrlTemplate format="text/html"
>http://example.metadata.org/csw/htmlrepaesentation/${metadataSetId}</MetadataUrlTempl
ate> ④
    <Dataset> ⑤
      <Name xmlns:app="http://www.deegree.org/app">app:streets</Name> ⑥
      <Title>Streets</Title> ⑦
      <Abstract>Streets of the city of Hamburg</Abstract>
      <MetadataSetId>beefcafe-beef-cafe-beef-cafebeefcaf</MetadataSetId>
    </Dataset>
  </DatasetMetadata>

</deegreeServicesMetadata>
```

① information about the service, in the context of ogcapi it is used per dataset and is shown on the landing page

② supports CDATA section with HTML elements

③ service metadata link, in the context of ogcapi this link is used in the collection view link of the metadata

④ service metadata link in format `text/html`, in the context of ogcapi this link is used in the collection view linking to the HTML representation of the metadata.

⑤ Use this element when you have a metadata record for the defined feature collection, otherwise define the link to the metadata record on the dataset level as described in chapter Dataset configuration for the element `<MetadataURL/>`.

⑥ feature collection name which links to the feature type configured, here the {collectionId}.

⑦ title of the feature collection, used in HTML encoding instead of the {collectionId}

NOTE The file *streets_metadata.xml* must be stored in the subdirectory *services/*. The file is mandatory.

A detailed documentation of the deegree service metadata configuration is described in section "Metadata" of the deegree webservices handbook.

NOTE The content of this file is returned under the resources */datasets/{datasetId}*, */datasets/{datasetId}/collections* and */datasets/{datasetId}/collections/{collectionId}* providing information about metadata.

5.2.4. Feature store configuration

Currently, deegree ogcapi supports the following feature stores:

- `SQLFeatureStore` - retrieves data from a database supporting an extended mapping
- `SimpleSQLFeatureStore` - retrieves data from a database using a single table mapping
- `MemoryFeatureStore` - retrieves data from a file in GML file format
- `ShapeFeatureStore` - retrieves data from a file in SHAPE file format (storage CRS is required when using this FeatureStore)

The Storage CRS defined in the feature store configuration is used to return the `Content-Crs` HTTP header in each response.

The supported databases for `SQLFeatureStore` and `SimpleSQLFeatureStore` are:

- Oracle database
- PostgreSQL/PostGIS database.

A detailed documentation of the feature store configuration is described in section "Feature Stores" of the deegree webservices handbook.

NOTE The *{featureId}* is defined by the feature store configuration. Use the element `<FIDMapping/>` to define the mapping of this attribute.

Using schema- or table-driven FeatureStore configurations

When using a GML application schema in the feature store configuration the schema is used in the OpenAPI document for XML and JSON encoding to describe the data types (schema-driven mode). If no application schema is provided the data type description is derived from the feature store mapping (table-driven mode).

The following table shows the supported features depending on the feature store configuration.

	Table-driven	**Schema-driven**	**Description**
GML encoding	supported	supported	fully supports GML 3.2 and all deegree mappings
JSON encoding	supported	supported	derived from GML encoding without feature references and complex types
HTML encoding	limited	limited	derived from JSON encoding, only primitive properties and lists of primitives
GML schema	provided	provided	linked in OpenAPI document, available at `collections/{collectionId}/appschema` as XSD only
JSON schema	without GML properties	including GML properties	data type definition is provided in OpenAPI document

More information about table-driven and schema-driven mode is provided in section "Mapping GML application schemas" of the deegree webservices handbook.

5.2.5. HTML encoding configuration

To configure the HTML encoding a configuration file can be used. The following example contains the configuration for the dataset *trees*.

html/treesview.xml

```
<HtmlView xmlns="http://www.deegree.org/ogcapi/htmlview"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://www.deegree.org/ogcapi/htmlview
https://schemas.deegree.org/ogcapi/1.3/htmlview.xsd">

  <CssFile>../html/lgv.css</CssFile>  ①
  <LegalNoticeUrl>https://www.hamburg.de/legalNotice/</LegalNoticeUrl> ②
  <PrivacyPolicyUrl>https://www.hamburg.de/datenschutz/</PrivacyPolicyUrl> ③
  <DocumentationUrl>https://www.hamburg.de/</DocumentationUrl> ④
  <Map> ⑤
    <WMSUrl version="1.3.0">
https://geodienste.hamburg.de/HH_WMS_Cache_Stadtplan</WMSUrl> ⑥
    <WMSLayers>stadtplan</WMSLayers> ⑦
    <CrsProj4Definition code="EPSG:25832">+proj=utm +zone=32 +ellps=GRS80
+towgs84=0,0,0,0,0,0,0 +units=m +no_defs</CrsProj4Definition> ⑧
    <Source><![CDATA[© <a href="https://www.hamburg.de/"
target="_new">Datenquelle</a>]]></Source> ⑨
  </Map>

</HtmlView>
```

① CSS file used for all HTML views (optional)

② link to page containing the publishing, copyright, and legal information (optional)

③ link to page containing the privacy policy (optional)

④ link to page containing the documentation (optional)

⑤ configuration of the base map (optional)

⑥ URL of WMS used for the base map (required)

⑦ layer name served by the WMS used for the base map (required)

⑧ CRS configuration of the base map

⑨ adds information about the source of the base map to the map element. The text element can contain HTML packed in a CDATA section (optional)

NOTE The file *treesview.xml* must be stored in the subdirectory *html/*. To define a global configuration the file name must be *htmlview.xml*. The file is optional.

This configuration file can contain the following elements:

Option	Cardinality	Value	Description
CssFile	0..1	URI	relative path to a CSS file
LegalNoticeUrl	0..1	URL	URL to external page containing the legal notice used for link in footer "Legal Notice"

Option	Cardinality	Value	Description
PrivacyPolicyUrl	0..1	URL	URL to an external page containing the privacy policy for link in footer "Privacy Policy"
DocumentationUrl	0..1	URL	URL to an external page containing the documentation, if not set link in footer "Help" refers to this documentation
Map	0..1	Complex	Configuration for the base map

The element `<Map/>` has the following subelements:

Option	Cardinality	Value	Description
WMSUrl	1	URL	WMS service endpoint URL, default: http://sg.geodatenzentrum.de/wms_dtk250. Use the attribute `version` to specify the WMS version
WMSLayers	1	String	Name of the layer, default: dtk250
CrsProj4Definition	1	String	Use the attribute `code` to set the EPSG code, and the value element for the PROJ definition as provided by http://epsg.io.

Additional information about the option `CssFile`: The following elements can be configured using a CSS file: the background color of header and footer, images in header and footer, links to help, legal notice, and privacy policy.

NOTE The content of this file is returned under the resources */datasets/{datasetId}* for HTML encoding only.

5.3. deegree config REST-API

deegree ogcapi provides a REST-API for configuration purposes. As in deegree webservices a client can use the REST interface to manage the configuration. The following operations are supported:

```
[HTTP METHOD] [RESOURCE] - [DESCRIPTION]
GET /config/download[/path] - download currently running workspace or file in
workspace
GET /config/restart - restart currently running workspace
GET /config/restart[/path] - restarts all resources connected to the specified one
GET /config/update - update currently running workspace, rescan config files and
update resources
GET /config/update/bboxcache[?featureStoreId=] - recalculates the bounding boxes of
all feature stores of the currently running workspace, with the parameter
'featureStoreId' a comma separated list of feature stores to update can be passed
GET /config/list[/path] - list currently running workspace or directory in workspace
GET /config/validate[/path] - validate currently running workspace or file in
workspace
PUT /config/upload/path/file - upload file into current workspace
DELETE /config/delete[/path] - delete currently running workspace or file in workspace
```

The REST-API is enabled by default. To protect this interface from unauthorized use, it is automatically secured with a so-called API key. Each HTTP request requires that the API key contained in the file *config.apikey* is transferred.

A detailed documentation of the REST-API interface and how access is configured is described in section "deegree REST interface" of the deegree webservices handbook.

5.4. Allow access to OpenAPI document from all origins

In case you want to avoid any issues when using the OpenAPI document from other locations due to CORS, you can enable allowing all origins specifically for accessing the OpenAPI document. To enable this set the system property `deegree.oaf.openapi.cors.allow_all` to *true*.

5.5. Logging configuration

The deegree ogcapi is using the Logback Project for logging. By default, deegree ogcapi uses the deegree autoconfiguration submodule. This module can log to the console and to files, adjust log patterns, and define log levels for various components of deegree ogcapi and its dependencies. If more advanced or customized logging is required, you can supply a Logback XML configuration file, which disables autoconfiguration and enables even complex logging setups.

TIP Place your `logback.xml` file in the classpath of the deegree ogcapi application to provide your custom logging configuration.

Further details on configuring both the autoconfiguration module and a file-based Logback setup can be found in the documentation for deegree webservices.

Chapter 6. Usage

This section describes how to access the data in the different encodings and how to use the HTML encoding to browse the data.

The main entry point for deegree ogcapi is provided under the resource path */datasets*. Given that the deegree ogcapi webapp is deployed under the context path */deegree-ogcapi* the resulting example URL for a local server running on port 8080 is: http://localhost:8080/deegree-ogcapi/datasets

6.1. Datasets overview

This resource provides an overview of all datasets served by deegree ogcapi:

Resource	Path	HTTP method	Supported Encodings	Description
Datasets overview	`/`	GET	`text/html`, `application/json`	The resource provides an overview of all datasets available
Landing page per dataset	`/{datasetId}`	GET	`text/html`, `application/json`	The API landing page per dataset (see next section for further information)

NOTE The context for the listed resource is */datasets*. This resource is not defined by the OGC API - Features standard and is an implementation specific resource.

All resources are available in different encodings. To request a resource in a different encoding the client shall use either the HTTP `Accept` header or the query parameter `f` to retrieve the data in the requested media type. To retrieve the supported media types per resource use the OpenAPI document available under '/api'.

6.2. Landing page per dataset

Besides the resources listed above for each dataset all resources defined by the OGC API - Features standard are supported. The configuration examples described in this document would result to the following addresses for each dataset. The landing page for the dataset *trees* would be served under the address http://localhost:8080/deegree-ogcapi/datasets/trees and the landing page for the dataset *streets* would be served under the address http://localhost:8080/deegree-ogcapi/datasets/streets.

The following table show the resources available per dataset:

Resource	Path	HTTP method	Supported Encodings	Description
Landing page	`/`	GET	`text/html`, `application/json`	Landing page is the top-level resource, which serves as an entry point per dataset
OpenAPI	`/api`	GET	`text/html`, `application/json`, `application/yaml`	API specification document provides metadata about the API itself
Conformance declaration	`/conformance`	GET	`text/html`, `application/json`	Declaration of conformance classes presents information about the functionality that is implemented by the server
Feature collections	`/collections`	GET	`text/html`, `application/json`, `application/xml`	Feature collections overview
Feature collection	`/collections/{collectionId}`	GET	`text/html`, `application/json`, `application/xml`	Feature collection identified by {collectionId}
Features	`/collections/{collectionId}/items`	GET	`text/html`, `application/json`, `application/xml`	List of features
Feature	`/collections/{collectionId}/items/{featureId}`	GET	`text/html`, `application/json`, `application/xml`	Feature identified by {featureId}
Provider License	`/license/provider`	GET	`text/html`, `application/json`	Provider license
Dataset License	`/license/dataset`	GET	`text/html`, `application/json`	Dataset license

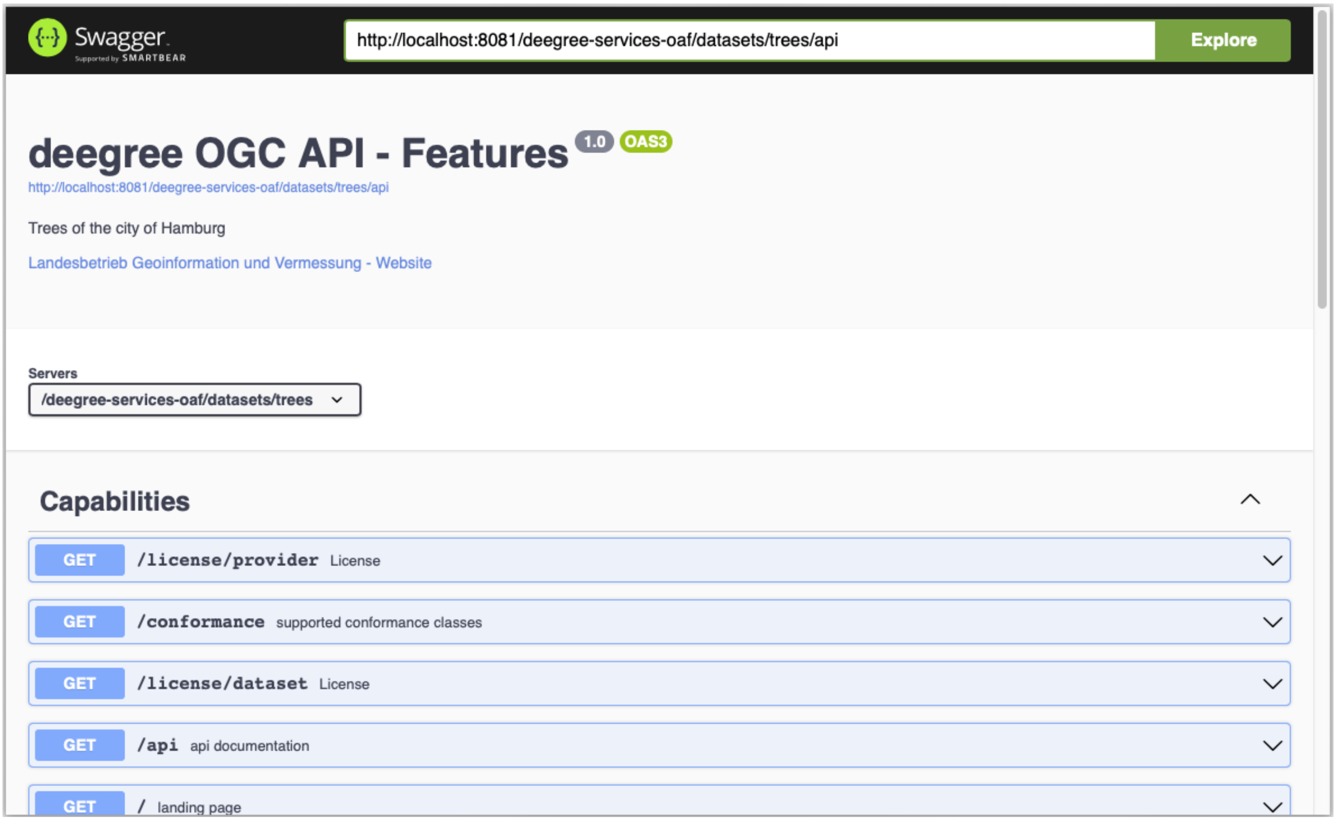
NOTE The context for the listed resource is *datasets/{datasetId}*. The OGC API - Features standard defines all resources from this base resource.

6.3. OpenAPI document

The OpenAPI page is available under the context */api* or */openapi*. The use of */api* on the server is optional and the API definition may be hosted on completely separate server. If you have deployed the deegree ogcapi webapp under the context of *deegree-ogcapi* and you have a dataset configured with the name *streets* the resulting request path would be */deegree-ogcapi/datasets/streets/api*.

The interface is grouped in the sections:

- Capabilities - capabilities of the datasets
- Data - access to features
- Collections - access to feature collections
- Schemas - access to schemas


http://localhost:8081/deegree-services-oaf/datasets/trees/api

deegree OGC API - Features 1.0 OAS3

http://localhost:8081/deegree-services-oaf/datasets/trees/api

Trees of the city of Hamburg

Landesbetrieb Geoinformation und Vermessung - Website

Servers

/deegree-services-oaf/datasets/trees

Capabilities

GET /license/provider License

GET /conformance supported conformance classes

GET /license/dataset License

GET /api api documentation

GET / landing page

Figure 1: Swagger start page

NOTE For each resource in the data section a schema description is provided derived from the underlying feature store configuration. See section Feature store configuration for more information which are supported. Schema and example data may vary depending on the selected encoding and the underlying configuration.

For all resources listed on the first three sections the following description explains how to send a request and retrieve the response in a given encoding.

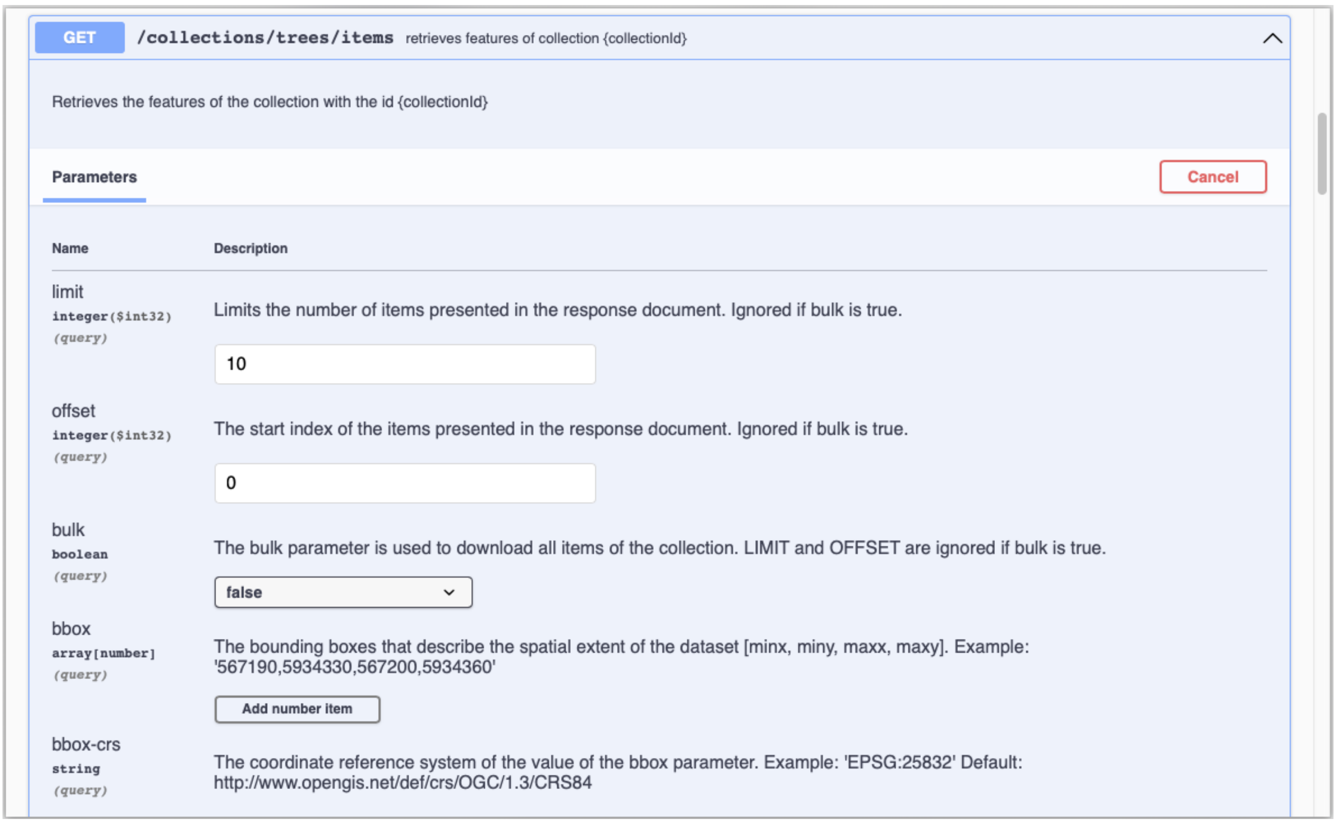
6.4. Making a request

Use the generic OpenAPI specification HTML page to make a request.

Example request: Get Landing page in `json` encoding.

1. Click on the button "GET" next to the resource /

2. Click on the button "Try it out"
3. Select the media type `application/json` from the selection list below "Responses"
4. Click on the button "Execute"

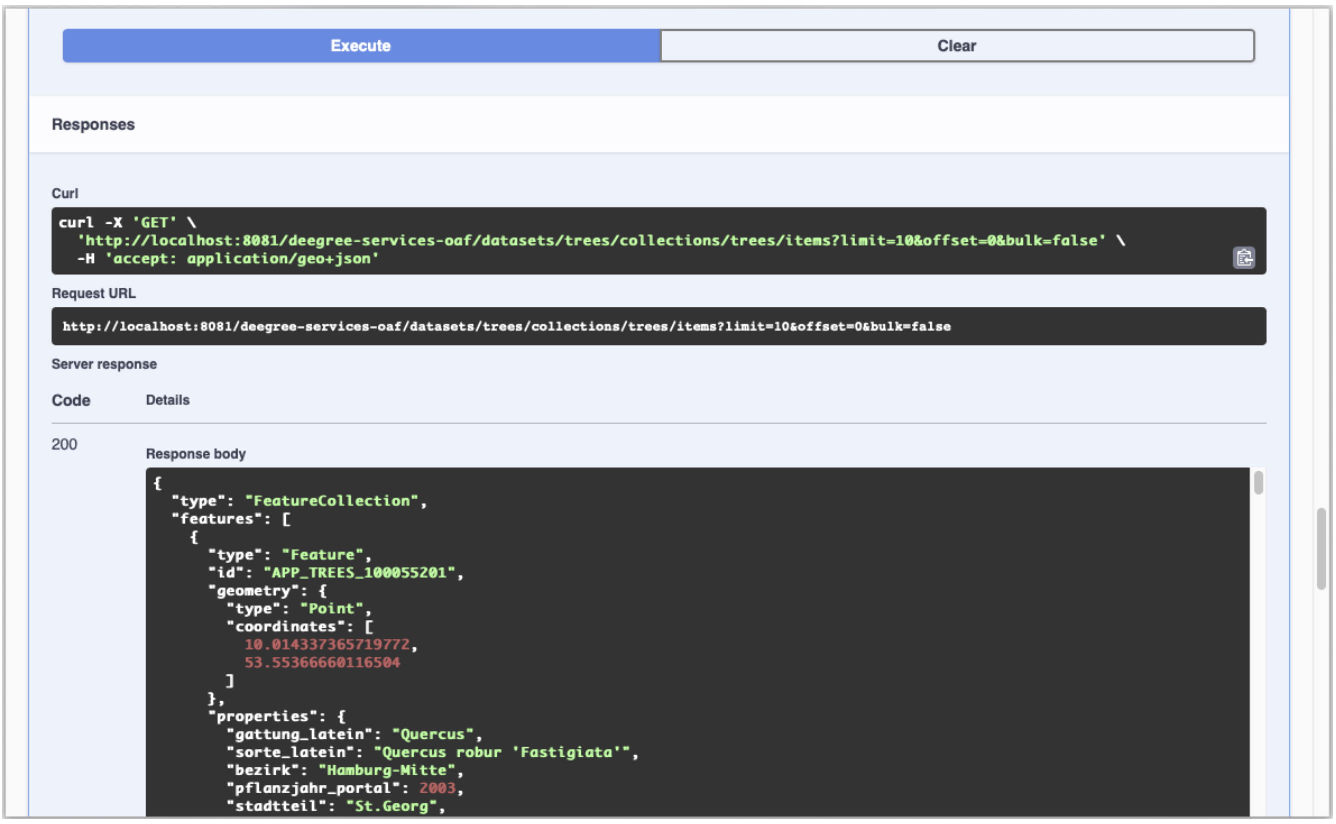

GET
/collections/trees/items retrieves features of collection {collectionId}
Parameters
Cancel
Name
Description
Add number item

Retrieves the features of the collection with the id {collectionId}

Name	Description
limit integer($int32) (query)	Limits the number of items presented in the response document. Ignored if bulk is true. 10
offset integer($int32) (query)	The start index of the items presented in the response document. Ignored if bulk is true. 0
bulk boolean (query)	The bulk parameter is used to download all items of the collection. LIMIT and OFFSET are ignored if bulk is true. false
bbox array[number] (query)	The bounding boxes that describe the spatial extent of the dataset [minx, miny, maxx, maxy]. Example: '567190,5934330,567200,5934360'
bbox-crs string (query)	The coordinate reference system of the value of the bbox parameter. Example: 'EPSG:25832' Default: http://www.opengis.net/def/crs/OGC/1.3/CRS84

Figure 2: Swagger send request

The page should display the server response in the selected encoding and the HTTP status code. In addition the HTTP response header information is displayed.



```
curl -X 'GET' \
  'http://localhost:8081/deegree-services-oaf/datasets/trees/collections/trees/items?limit=10&offset=0&bulk=false' \
  -H 'accept: application/geo+json'
```

```
http://localhost:8081/deegree-services-oaf/datasets/trees/collections/trees/items?limit=10&offset=0&bulk=false
```

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "id": "APP_TREES_100055201",
      "geometry": {
        "type": "Point",
        "coordinates": [
          10.014337365719772,
          53.55366660116504
        ]
      },
      "properties": {
        "gattung_latein": "Quercus",
        "sorte_latein": "Quercus robur 'Fastigiata'",
        "bezirk": "Hamburg-Mitte",
        "pflanzjahr_portal": 2003,
        "stadtteil": "St.Georg",
```

Figure 3: Swagger response

6.5. Accessing a response

To access a response in the requested encoding directly use either the given command line tool `curl` with the given options as displayed in the OpenAPI page when sending a request described in section Making a request. Or use a browser and additional plugins to send the HTTP request directly. Use the HTTP `Accept` header or the query parameter *f* to define the expected response format.

Media type	Query parameter	`Accept` header	Description
`application/json`	`?f=json`	`application/json`, `application/geo+json`	JSON/GeoJSON encoding
`application/xml`	`?f=xml`	`application/xml`, `application/gml+xml`	XML/GML encoding
`text/html`	-	`text/html`	HTML encoding

6.6. Accessing data in JSON/GeoJSON format

To retrieve a resource in `application/json` encoding use the request parameter *f=json*. To retrieve the landing page of the dataset streets in `application/json` encoding use the following request *datasets/streets/?f=json*. Example URL: http://localhost:8080/deegree-ogcapi/datasets/streets/?f=json

See section Using query parameters for more information about other supported query parameters.

6.7. Accessing data in XML/GML format

To retrieve a resource in `application/xml` encoding use the request parameter *f=xml*. To retrieve the landing page of the dataset streets in `application/xml` encoding use the following request *datasets/streets/?f=xml*. Example URL: http://localhost:8080/deegree-ogcapi/datasets/streets/?f=xml

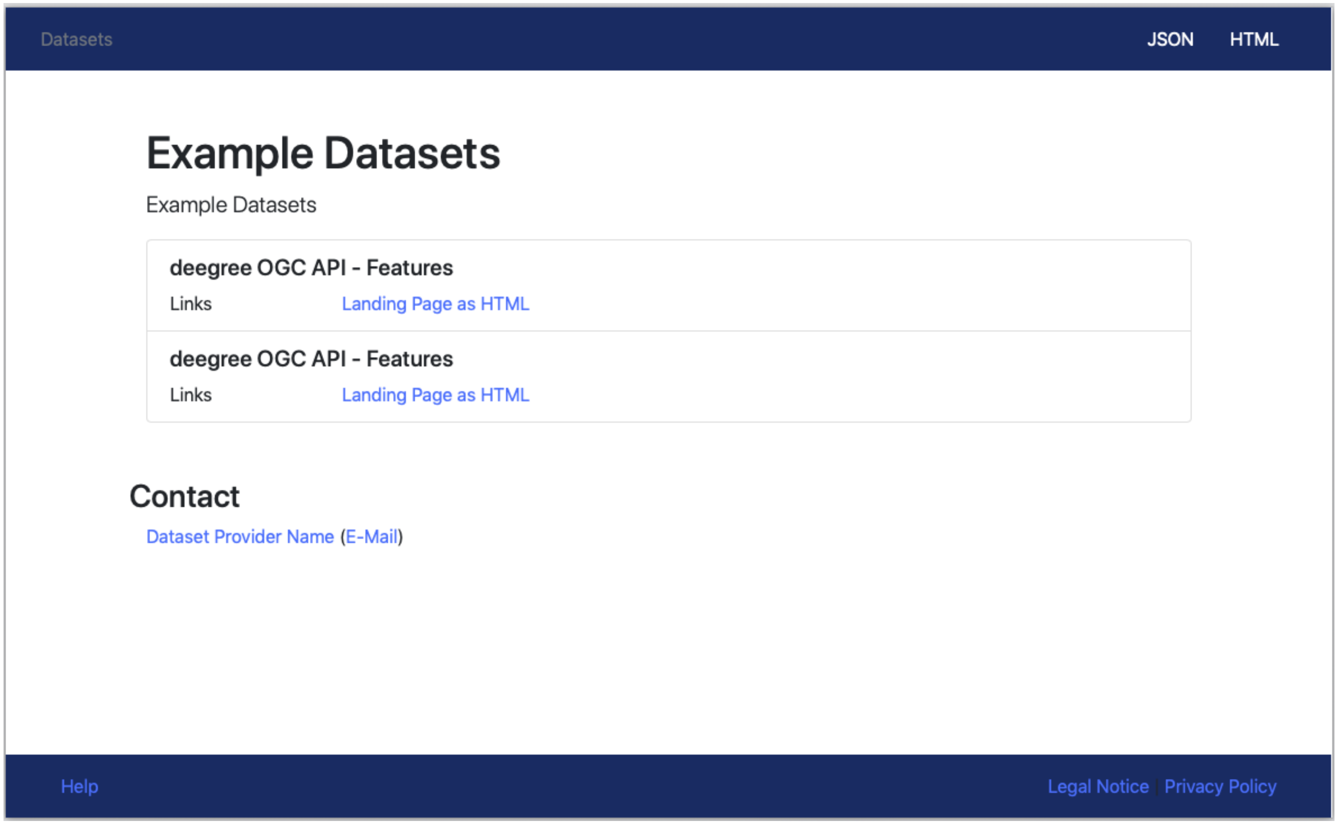
See section Using query parameters for more information about supported other query parameters.

The corresponding GML schema file is provided under the resource of each Feature collection *datasets/{datasetId}/collections/{collectionId}/appschema*. Example URL: https://localhost:8080/deegree-ogcapi/datasets/streets/collections/streets/appschema

6.8. Using the HTML interface

The HTML interface provides easy access to the spatial data using a browser (check the list of Supported Browser). It requires no additional client or browser plugin to browse the data. The browser sends by default the HTTP header `Accept` with the value `text/html` and therefore each resource is returned in HTML encoding.

To browse the data open the browser of your choice and start at the datasets overview available at *datasets/*. Example URL: http://localhost:8080/deegree-ogcapi/datasets



Example Datasets

Example Datasets

deegree OGC API - Features

Links Landing Page as HTML

deegree OGC API - Features

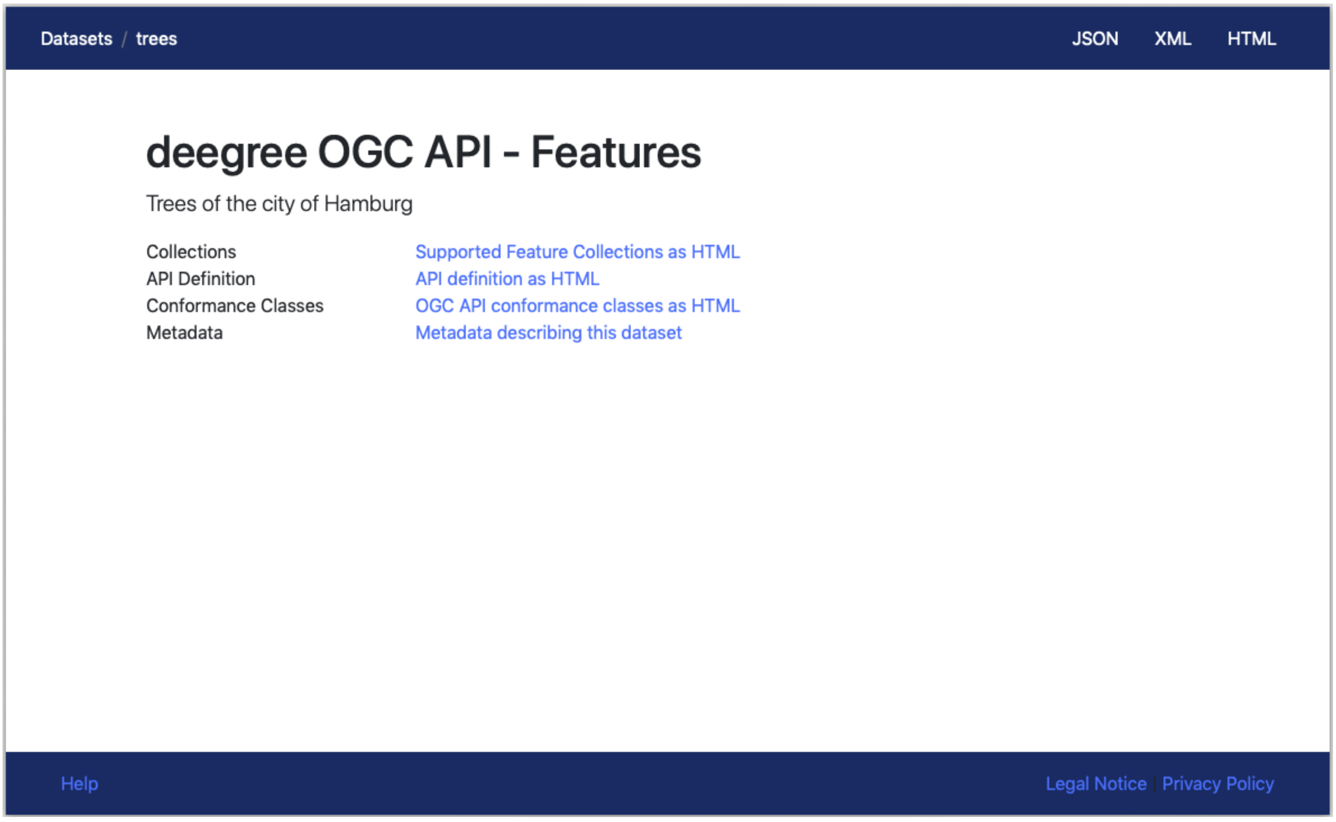
Links Landing Page as HTML

Contact

Dataset Provider Name (E-Mail)

Figure 4: Datasets overview in HTML encoding

Navigate to the landing page of the dataset *trees* by clicking on the link "Landing page as HTML".


Datasets / trees JSON XML HTML

deegree OGC API - Features

Trees of the city of Hamburg

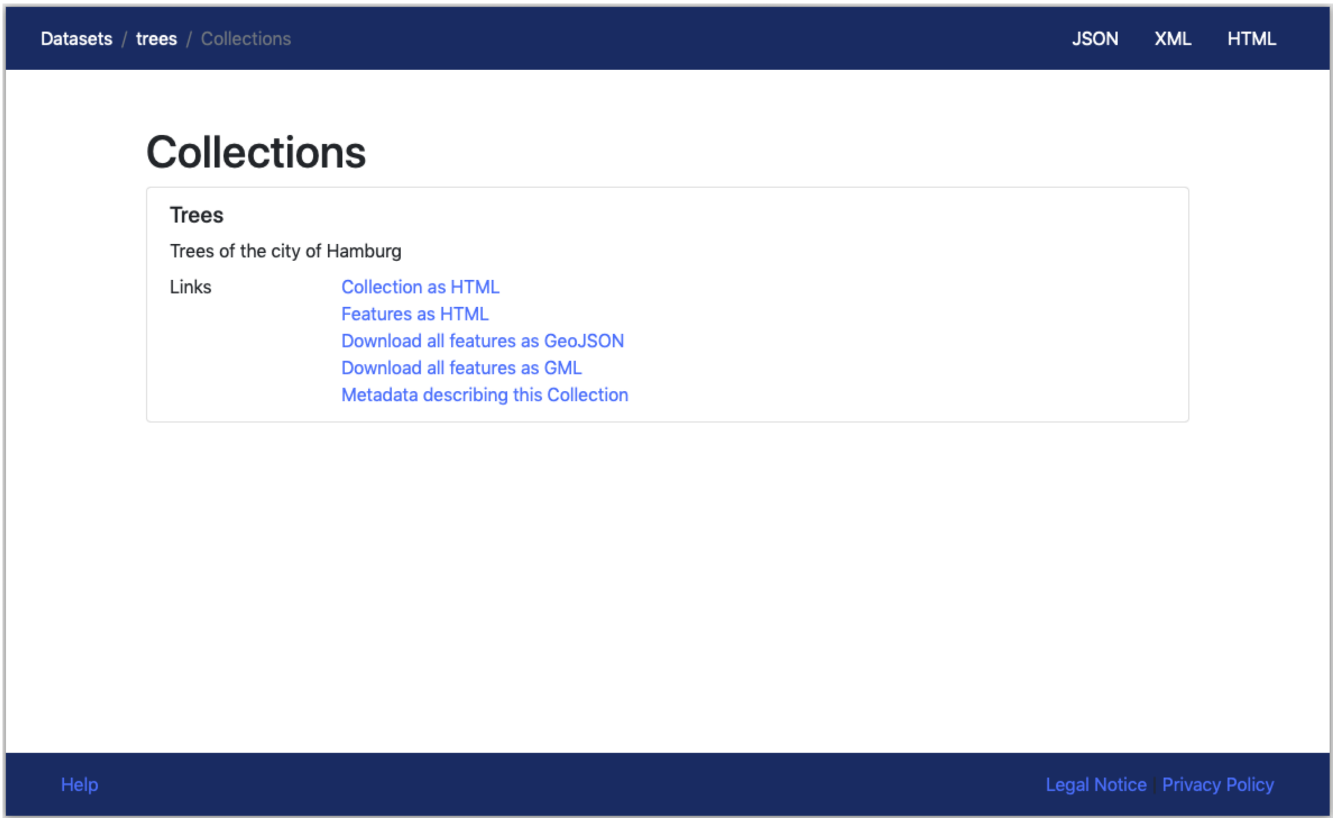
Collections	Supported Feature Collections as HTML
API Definition	API definition as HTML
Conformance Classes	OGC API conformance classes as HTML
Metadata	Metadata describing this dataset

Help Legal Notice | Privacy Policy

Figure 5: Landing page in HTML encoding

The landing page provides links to all resources of a dataset.

When navigating to the feature collections and feature collection resource links to access the referenced metadata and bulk download in GeoJson and GML encoding are provided.


Datasets / trees / Collections JSON XML HTML

Collections

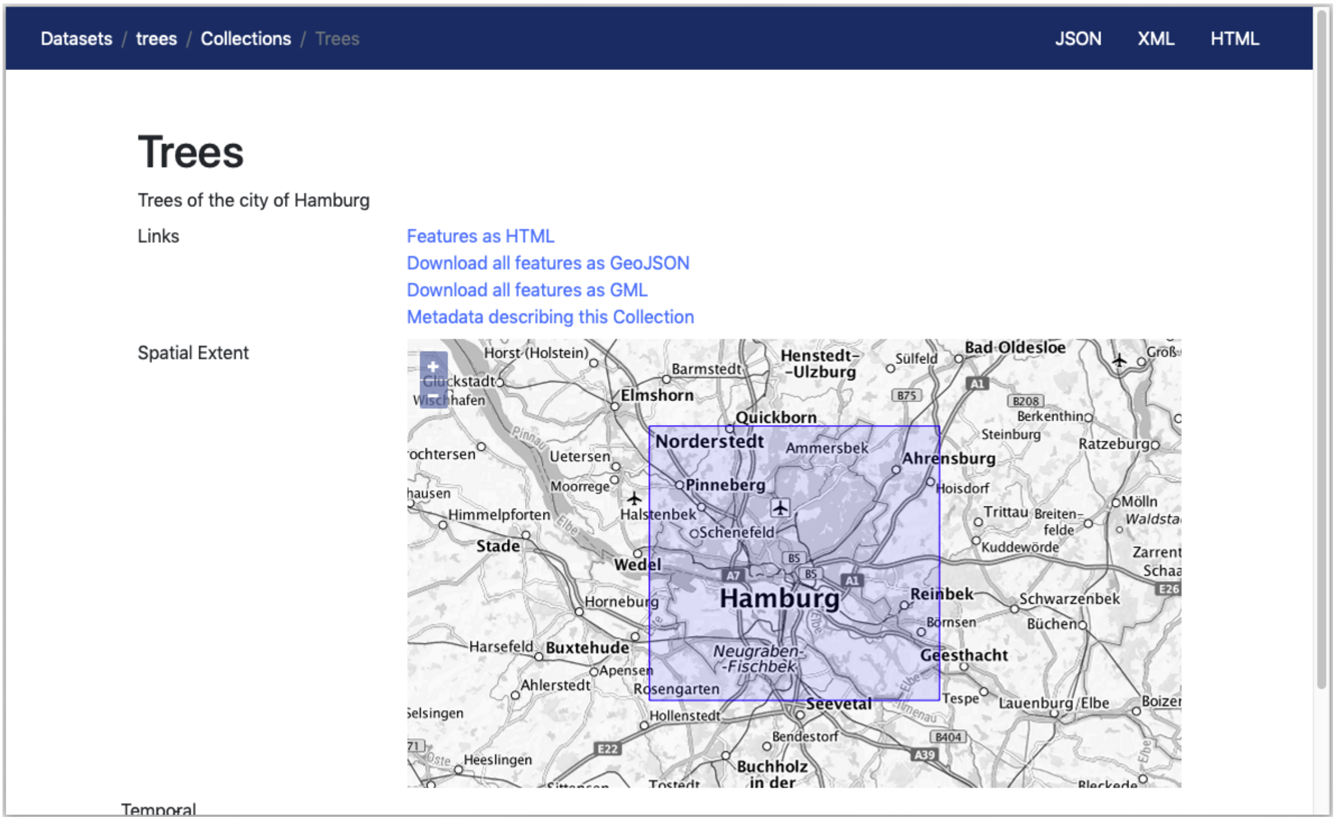
Trees

Trees of the city of Hamburg

Links
Collection as HTML
Features as HTML
Download all features as GeoJSON
Download all features as GML
Metadata describing this Collection

Help Legal Notice | Privacy Policy

Figure 6: Feature collections page in HTML encoding



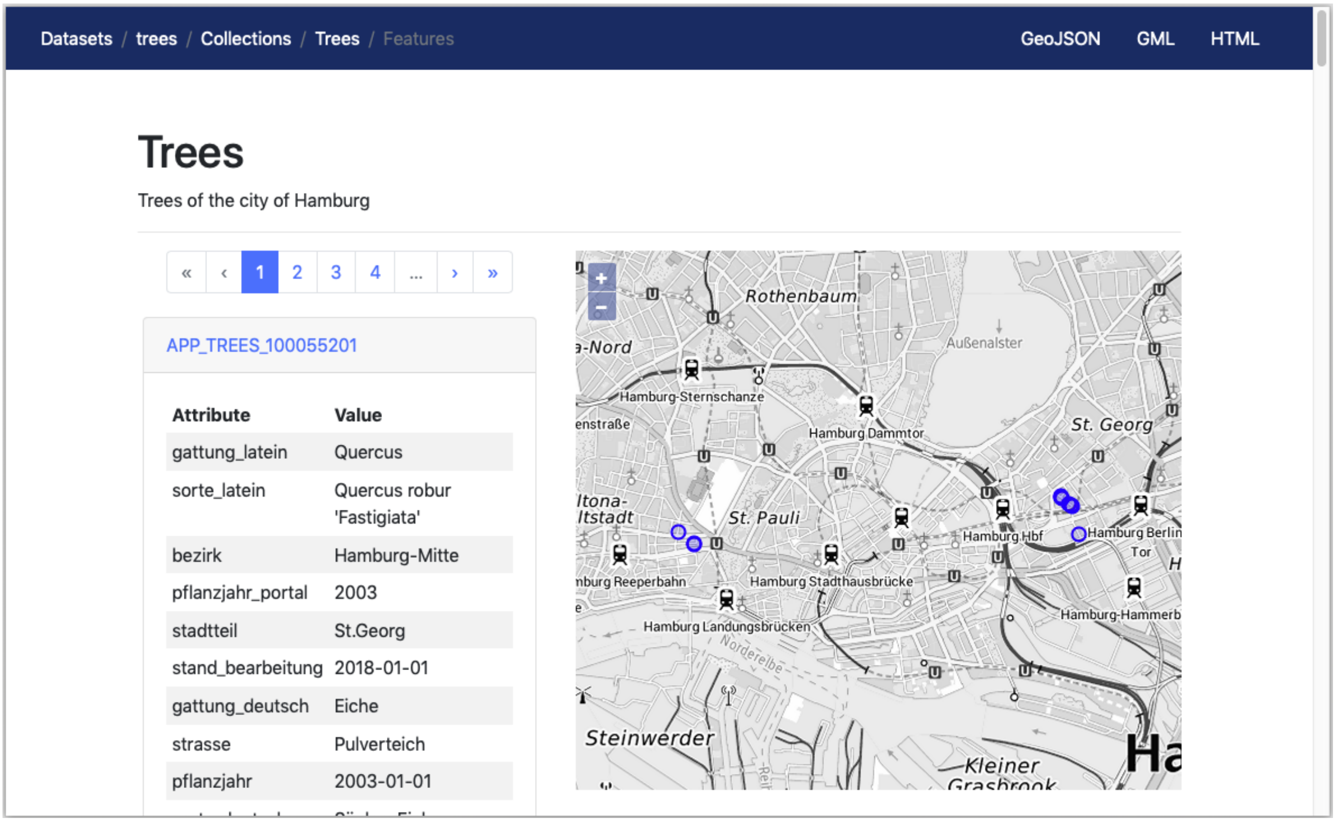
Trees

Trees of the city of Hamburg

Links	Features as HTML Download all features as GeoJSON Download all features as GML Metadata describing this Collection
Spatial Extent	

Temporal

Figure 7: Feature collection page in HTML encoding



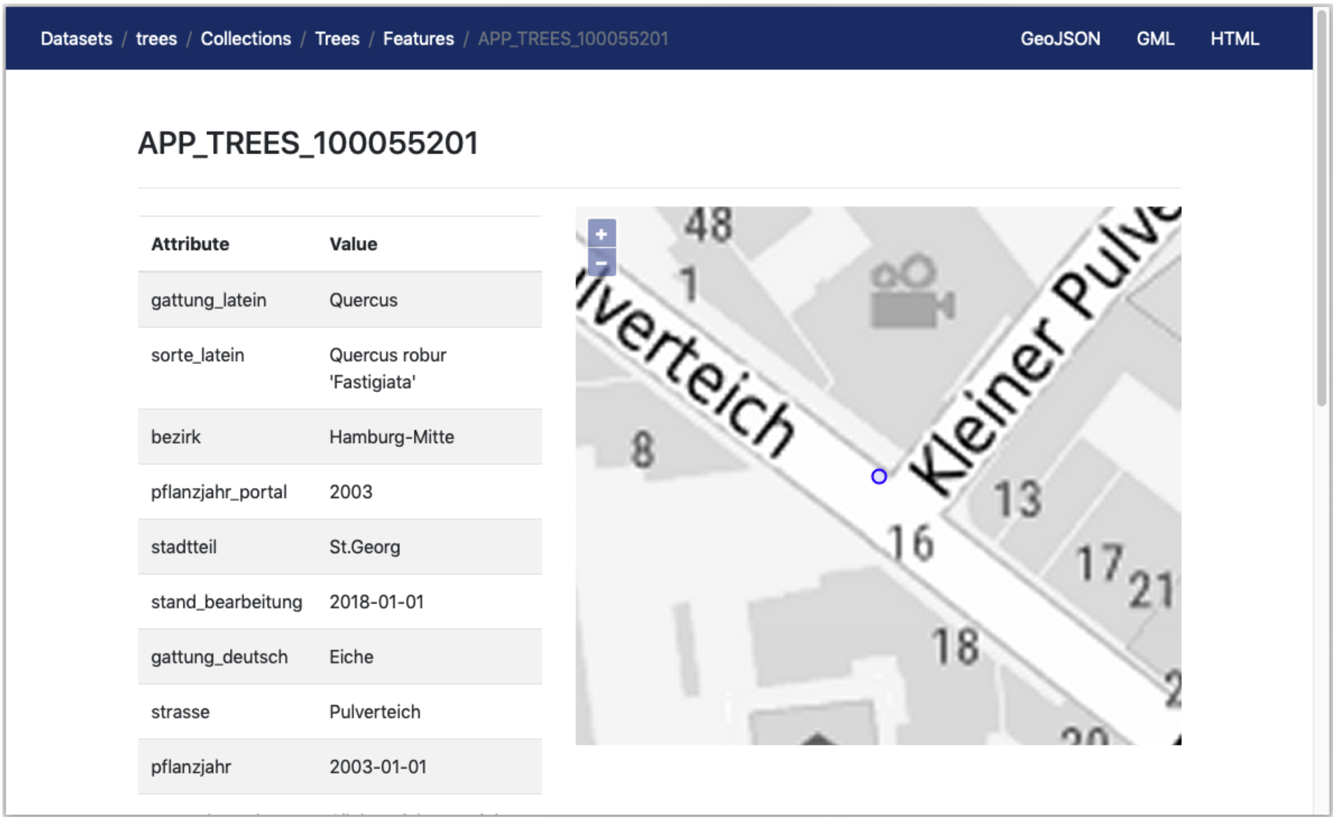
Trees

Trees of the city of Hamburg

APP_TREES_100055201

Attribute	Value
gattung_latein	Quercus
sorte_latein	Quercus robur 'Fastigiata'
bezirk	Hamburg-Mitte
pflanzjahr_portal	2003
stadtteil	St.Georg
stand_bearbeitung	2018-01-01
gattung_deutsch	Eiche
strasse	Pulverteich
pflanzjahr	2003-01-01

Figure 8: Feature items page in HTML encoding


Datasets / trees / Collections / Trees / Features / APP_TREES_100055201 GeoJSON GML HTML

APP_TREES_100055201

Attribute	Value
gattung_latein	Quercus
sorte_latein	Quercus robur 'Fastigiata'
bezirk	Hamburg-Mitte
pflanzjahr_portal	2003
stadtteil	St.Georg
stand_bearbeitung	2018-01-01
gattung_deutsch	Eiche
strasse	Pulverteich
pflanzjahr	2003-01-01

Figure 8: Feature item page in HTML encoding

6.9. Using query parameters

The following query parameters are supported when using HTTP GET:

Query parameter name	Value type	Example value	Description
`crs`	String	EPSG:4326	EPSG code defines the CRS of the returned data
`bbox`	Comma separated floating point values	567190,5934330, 567200,5934360	List of comma separated floating point values defining a bounding box
`bbox-crs`	String	EPSG:4326	EPSG code defines the CRS of the coordinates of the `bbox` parameter
`f`	String	json	Requested encoding of a given resource, can be `json`, `html` or `gml`
`limit`	integer	10	Limit the numbers of items per page
`offset`	integer	0	Start index of items
`bulk`	boolean	true	Applicable for features resource only, can be combined with parameter `f`

Query parameter name	Value type	Example value	Description
`filter`	String	S_INTERSECTS({spatialQueryable},{spatialInstance}), T_AFTER({temporalQueryable},{temporalInstance})	Filter limited to `S_INTERSECTS` with first operand `{spatialQueryable}` defining the property name and the second operand `{spatialInstance}` the basic spatial data type point or bounding box. Filter limited to `T_AFTER` with first operand `{temporalQueryable}` defining the property name and the second operand `{temporalInstance}` with a date `DATE('2026-01-01')` or datetime `TIMESTAMP('2025-04-14T08:59:30Z')`. Available `{temporalQueryable}` are listed as additionale queryable in the OpenAPI document document (type `date` or `date-time`), see note below.
`filter-lang`	String	cql2-text	Defines the filtering language, indicates that the value of the `filter` parameter is the text encoding of CQL2, can be combined with parameter `filter`
`filter-crs`	String	EPSG:4326	Allows clients to assert which CRS is being used to encode geometric values in a `filter` expression, can be combined with parameter `filter`

NOTE Check the OpenAPI document on which resources the listed query parameters are supported. Additional query parameters may be available depending on the resource.

Chapter 7. Help

This section show how to get help in case of problems.

7.1. Known Issues

- Mixing legacy service configuration with ogcapi configuration is currently not supported by deegree ogcapi webapp. See the FAQ how to serve your data with deegree webservices and deegree ogcapi from the same workspace configuration.
- Startup time of deegree ogcapi webapp may increase noticeably when the deegree workspace contains many resources (e.g. more than 100 files).
- Using the browser's navigation buttons when using the HTML view may result in empty pages or the switch to JSON encoding. Affected browsers: Google Chrome, Chromium. You can avoid this by enabling the Vary HTTP response header in your HTTP server configuration. Check the HTTP server documentation how to configure this.
- The implementation supports only parts of OGC API - Features - Part 3. The implementation is not fully supporting the following requirements classes: "Queryables", "Queryables as Query Parameters", "Filter" with "CQL2 Functions", and "Features Filter".

7.2. Support

Use the deegree support options (mailing lists, commercial support) to get help.

7.3. FAQ

1. *How to update the bbox_cache.properties?*

   Use REST-API operation `/config/update/bboxcache` to recalculate the bounding boxes of the feature stores. See section deegree config REST-API for more information how to use the REST-API.

2. *How to migrate an existing deegree workspace?*

   Given that an existing deegree workspace contains at least one feature store the following steps need to be done to provide the data via deegree ogcapi webapp:

   - Add a dataset configuration file in subdirectory *ogcapi/*, e.g. *streets.xml* as described in the example workspace in chapter Dataset configuration.
   - Configure the HTML encoding by adding a file in subdirectory *html/* (optional), e.g. *streetsview.xml* as described in the example workspace in chapter HTML encoding configuration.
   - Deploy the deegree ogcapi webapp as described in chapter Deploy the webapp.
   - Add mapping entry to *webapps.properties* file to select the deegree ogcapi workspace for the deegree ogcapi webapp.

3. *How to use the same workspace with deegree ogcapi and deegree webservices?*

   Deploy both webapps for deegree ogcapi and deegree webservices configured to use the same workspace called *ogcapi-workspace*. Use the file *webapps.properties* to configure this. Keep in mind that deegree ogcapi will provide the OGC API - Features and deegree webservices will provide only the legacy service implementations. You may see warnings in the logging messages for both instances that configuration files have been ignored.

4. *How to deploy multiple instances of the deegree ogcapi webapp?*

   Multiple deployments of the webapp are possible. Since each instance of the runtime environment can host one or more webapp. For every deployed webapp there must be a deegree workspace available with the name ***ogcapi-workspace***. It is possible that multiple instances of the webapp can share one single instance of a deegree workspace, for example a setup serving the same data by multiple instances in a load-balancing scenario. To serve different data configured in different deegree workspace's you need to run multiple instances of the runtime environment. Use the environment variable `DEEGREE_WORKSPACE_ROOT` to set the path to the deegree workspace directory. Read further in section Configuration about the deegree workspace and configuration files.

5. *How to configure server side caching for HTML view?*

   Server side caching must be disabled to access the HTML view of deegree OGC API. This is because the HTML view is generated from the JSON resource on-the-fly. Otherwise a client may show outdated content from the cache which was generated by a different client request. When requesting any resource of deegree OGC API the HTTP request header `Cache-Control: no-cache` shall be set to prevent caching. This also applies to proxy servers between the actual client and the deegree OGC API instance.