OGC Testbed-14

Next Generation Web APIs - WFS 3.0 Engineering Report

Table of Contents

Publication Date: 2019-03-07

Approval Date: 2018-12-13

Submission Date: 2018-11-22

Reference number of this document: OGC 18-045

Reference URL for this document: http://www.opengis.net/doc/PER/t14-D021

Category: Public Engineering Report

Editor: Jeff Harrison, Panagiotis (Peter) A. Vretanos

Title: OGC Testbed-14: Next Generation Web APIs - WFS 3.0 Engineering Report

OGC Engineering Report

indicate compliance with any LICENSOR standards or specifications.

This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

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

Chapter 1. Summary

The objective of the Next Generation APIs - WFS 3.0 effort in OGC Testbed-14 was to develop and test the Web Feature Service (WFS) version 3.0 candidate standard. The initiative assessed OpenAPI, security based on OpenID Connect and OAuth 2.0 and WFS 3.0 extensions. The effort also began to assess methods to ease geospatial enterprise transition to next generation Application Programming Interfaces (APIs).

The purpose of this effort was not to preempt other next generation work taking place in OGC, but rather to inform and complement that work.

This Engineering Report (ER) describes the implementations and experiments conducted by OGC Testbed-14 participants to test next generation Web APIs. It includes descriptions of APIs to simplify and secure access to geospatial feature resources, and was tested in a scenario that showed how WFS 3.0 can support humanitarian relief activities.

1.1. Requirements & Research Motivation

The motivation behind this ER resides in the need to secure the next generation open geospatial APIs, and ease transition from legacy services to geospatial enterprise architectures aligned with current Web architecture and Spatial Data on the Web Best Practices.

Therefore, the requirements addressed by this ER were:

- Adopt API components used in mainstream IT, and build in security.
- Use well-known resource types, and deploy geospatial feature resources as API components.
- Deploy feature resources in a simple core specified as OpenAPI components.
- Implement and test modular extensions for complex functions.
- Deploy facades and versioning to assess legacy service transition to next generation Web APIs.
- Develop and deploy client applications able to access feature resources using HTTP methods. Leverage openly available, browser-based tools such as Swagger to exercise the OpenAPI components.
- Secure the clients and services via an Authorization Server using OpenID Connect and OAuth 2.0. OpenID Connect is an authentication layer on top of OAuth 2.0, an authorization framework. OpenID Connect specifies an API using JavaScript Object Notation (JSON) as a data format.
- Conduct Technology Integration Experiments (TIEs) and document the ability of next generation APIs to support simulated users in a humanitarian relief scenario.

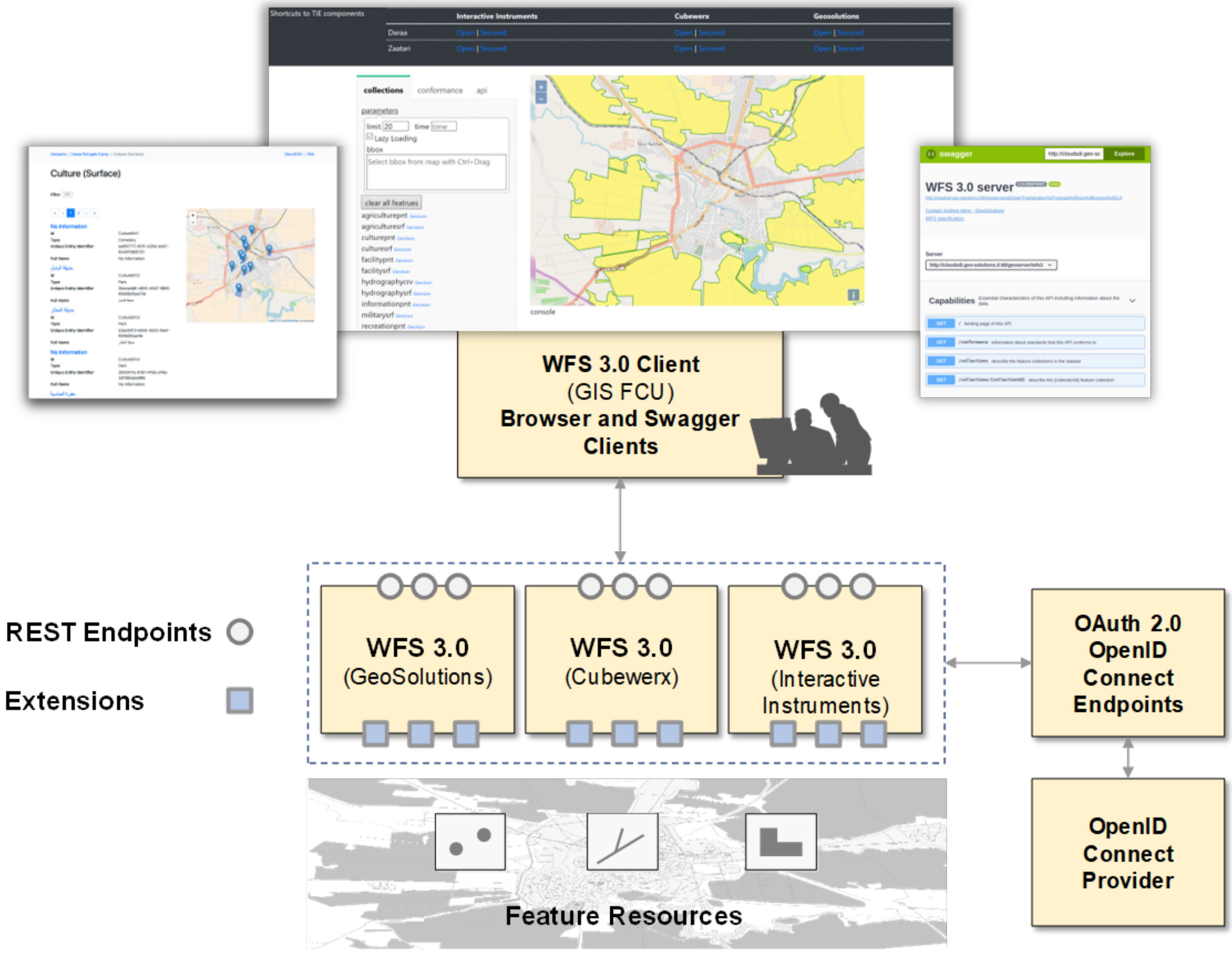
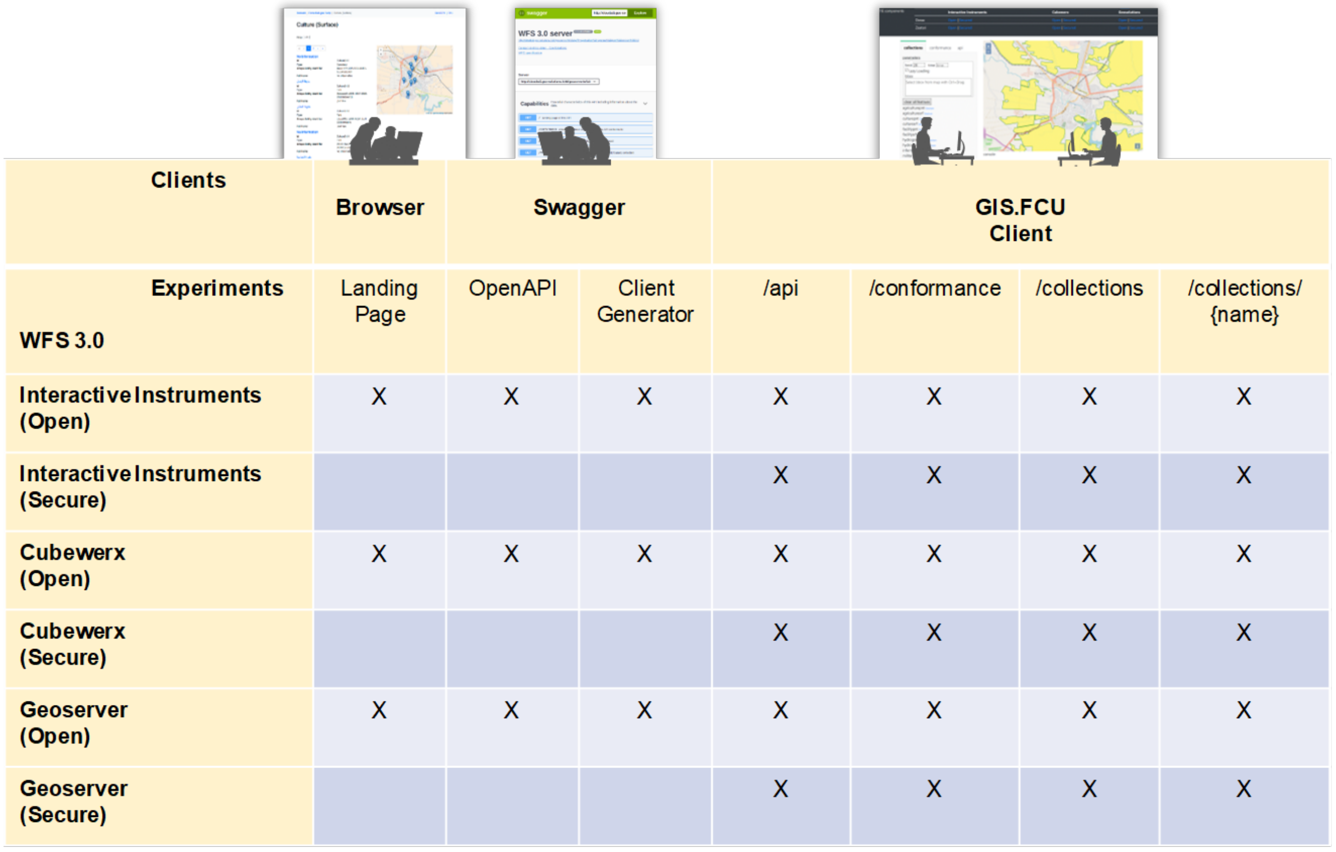


Figure 1. Overview of the Testbed-14 Next Generation APIs experiments architecture

Technology Integration Experiments conducted during Testbed-14 were informed by activity in the OGC Vector Tiles Pilot (VTP) which began in August 2018. The objective of the VTP was to extend WFS 3.0 and other OGC standards to deliver Vector Tiles.

Results of Testbed-14 indicate the simple core of WFS 3.0 specified as OpenAPI can be implemented rapidly by software developers, and deliver geospatial feature resources secured by OpenID Connect and OAuth 2.0. In addition, findings indicated feasible methods to ease transition to next-generation APIs are available through WFS versioning and facades.



Clients	Browser	Swagger		GIS.FCU Client			
Experiments / WFS 3.0	Landing Page	OpenAPI	Client Generator	/api	/conformance	/collections	/collections/{name}
Interactive Instruments (Open)	X	X	X	X	X	X	X
Interactive Instruments (Secure)				X	X	X	X
Cubewerx (Open)	X	X	X	X	X	X	X
Cubewerx (Secure)				X	X	X	X
Geoserver (Open)	X	X	X	X	X	X	X
Geoserver (Secure)				X	X	X	X

Figure 2. Technology Integration Experiments

1.2. Prior-After Comparison

Prior to Testbed-14 WFS 3.0 development has focused mainly on revising OGC's Web Feature Service standard for querying geospatial information on the web, concentrating on a simple core specified as reusable OpenAPI components with responses in JSON and Hypertext Markup Language (HTML). A key element of OpenAPI is the ability to provide standard, language-agnostic description for APIs and support to modern tools such as Swagger.

OpenAPI and Swagger allow both humans and computers to discover and understand the capabilities of an API without access to source code, documentation or through network traffic inspection. In addition, OpenAPI documents can be used by code generation tools to generate servers and clients in various programming languages.

Security discussions prior to Testbed-14 addressed a variety of approaches, but lacked an industry framework for integrating security controls into next generation APIs.

On the integration of OGC standards and definition of best practices for integration, the OGC® Web Services Security specification was published on the 28th January 2019 as an OGC Implementation Standard. The standard has been developed by the OGC Web Services (OWS) Common Security Standards Working Group (SWG) with the aim being to standardize the security aspects of current Web Services Standards while providing backward compatibility and interoperability.

On the other hand security discussions prior to this Testbed include concerns regarding Federated Identity Management, but these are not specifically addressed in enough detail.

This situation highlights the need for a general approach to security architectures that utilize the current state-of-the-art standards.

As a result of activities performed during this Testbed, the implementations documented in this Engineering Report serve as an initial step towards a complete high-level architecture to simplify and secure access to geospatial feature resources - including definition of security requirements objects aligned with OpenAPI. Documented implementations also indicate the potential of WFS 3.0 extensions to support Vector Tiles, maps and other innovations.

1.3. Recommendations for Future Work

As a result of the activities performed during this Testbed, several future work points have been identified:

- Assess and advance next generation APIs to support transactions against geospatial feature resources, perform geometry simplification and other functions.
- Advance a "WMTS 2.0" implemented as reusable OpenAPI components with security based on OpenID Connect and OAuth 2.0.
- Enhance the ability of next generation APIs to describe and deliver emerging geospatial resources such as Vector Tiles.
- Assess the ability of next generation APIs to support access control and security metadata, optionally enclosed within dissemination formats for binding assertion metadata with geospatial resources.
- Assess how security specifications, access control and dissemination may further enable JSON, HTML and Vector Tiles-based information exchange.

1.4. Document contributor contact points

All questions regarding this document should be directed to the editor or the contributors:

Contacts

Name	Organization
Jeff Harrison, Editor	US Army Geospatial Center
Peter Vretanos, Editor	Cubewerx
Clemens Portele	Interactive Instruments
Chia-Cheng (Ricky) Lin	Feng Chia University
Simone Giannecchini	GeoSolutions
Andrea Aime	GeoSolutions
Stefano Bovio	GeoSolutions
Hector Rodriguez	Deimos Space
Juan Jose Doval	Deimos Space

1.5. Review SWGs

The content of this ER will be reviewed by the following OGC standards working groups:

- Web Feature Service/Filter Encoding Specification (WFS/FES) SWG

1.6. Foreword

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

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

Chapter 2. References

The following normative documents are referenced in this document.

- OGC: OGC 06-121r9, OGC® Web Services Common Standard [https://portal.opengeospatial.org/files/?artifact_id=38867&version=2]
- OGC: OGC Web Feature Service 3.0 [https://github.com/opengeospatial/WFS_FES]
- Open API Initiative: OpenAPI Specification 3.0.1 [https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.1.md]
- IETF: RFC 6749 - The OAuth 2.0 Authorization Framework [https://tools.ietf.org/html/rfc6749]
- IETF: RFC 6750 - The OAuth 2.0 Authorization Framework: Bearer Token Usage [https://tools.ietf.org/html/rfc6750]
- IETF: RFC 7662 - OAuth 2.0 Token Introspection [https://tools.ietf.org/html/rfc7662]
- IETF: RFC 7662 - OAuth 2.0 Dynamic Client Registration Protocol [https://tools.ietf.org/html/rfc7591]
- OIDF, OpenID Connect Core 1.0 incorporating errata set 1, 2014 [https://openid.net/specs/openid-connect-core-1_0.html]
- W3C and OGC: Spatial Data on the Web Best Practice, 2017 [https://www.w3.org/TR/sdw-bp/]
- IETF: The OAuth 2.0 Authorization Framework, 2012 [https://tools.ietf.org/html/rfc6749]
- OpenAPI Initiative: OpenAPI Specification [https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.0.md#securityRequirementObject]

Chapter 3. Terms and definitions

For the purposes of this report, the definitions specified in Clause 4 of the OWS Common Implementation Standard OGC 06-121r9 [https://portal.opengeospatial.org/files/?artifact_id=38867&version=2] shall apply. In addition, the following terms and definitions apply.

- Authorization Code Grant Flow

  ```
  A security flow that can be used by confidential and public clients to exchange an
  authorization code for an access token.
  ```

- Dataset

  ```
  A collection of data, published or curated by a single agent, and available for
  access or download in one or more formats.
  ```

- Distribution

  ```
  Represents an accessible form of a dataset.
  ```

- Feature

  ```
  An abstraction of real world phenomena [ISO 19101-1:2014].
  ```

- Feature Collection | Collection

  ```
  A set of features from a dataset.
  ```

- Implicit Grant Flow

  ```
  A simplified security flow that can be used by public clients, where the access
  token is returned immediately without an extra authorization code exchange step.
  ```

- OpenAPI

  ```
  A specification for machine-readable interface files for describing, producing,
  consuming, and visualizing RESTful APIs.
  ```

- Swagger

```
Tools to visualize and interact with an API’s resources without having any of the
implementation logic in place, automatically generated from an OpenAPI document.
```

- Web Feature Service 3.0

```
A revision of the OGC's Web Feature Service standard for querying geospatial
resources on the web, focusing on a simple RESTful core specified as reusable
OpenAPI components with responses in JSON, HTML and emerging forms such as Vector
Tiles.
```

3.1. Abbreviated Terms

Some of the more frequently used abbreviated terms in this document include:

- API Application Programming Interface
- CORS Cross-Origin Resource Sharing
- COTS Commercial Off The Shelf
- ER Engineering Report
- HTML Hypertext Markup Language
- HTTP Hypertext Transfer Protocol
- JSON JavaScript Object Notation
- OIDC OpenID Connect
- OGC Open Geospatial Consortium
- OSM OpenStreetMap
- REST Representational State Transfer
- TIE Technology Integration Experiment
- TDS Topographic Data Store
- URL Uniform Resource Locator
- W3C World Wide Web Consortium
- WWW World Wide Web
- WFS Web Feature Service
- YAML YAML Ain’t Markup Language

Chapter 4. Overview

The objective of the Next Generation APIs - WFS 3.0 effort in Testbed-14 was to develop and test WFS 3.0. The initiative assessed OpenAPI, security based on OpenID Connect and OAuth 2.0 and WFS 3.0 extensions. The effort also began to assess methods to ease geospatial enterprise transition to next generation APIs.

This document contains the following sections:

- **Preface** - This section presents information on administrative and legal aspects of this Engineering Report (ER).
- Summary - This section presents information on scope, what this Engineering Report means for the OGC in general and document contributor contact points. It also provides forward-looking recommendations.
- References - This section presents information on documents that are referenced in this Engineering Report.
- Terms - This section presents information on terms and abbreviations that are used in this Engineering Report.
- Background - This section presents background information on technologies implemented in this Engineering Report.
- Experiments - This section presents information on the component implementations, architecture and the results of Technology Integration Experiments conducted.
- Implementations - This section presents detailed information on service endpoints and applications implemented in this Engineering Report.
- Extensions - This section presents information on extensions to the core API.
- Findings - This section summarizes the findings. It also provides forward-looking recommendations.

Chapter 5. Background

The objective of the Next Generation APIs - WFS 3.0 effort in Testbed-14 was to develop and test WFS 3.0. The goal was to experiment with the new WFS 3.0 specification, OpenAPI and Swagger and add security mechanisms based on OpenID Connect and OAuth 2.0. The effort also began to assess WFS 3.0 extensions and methods to ease geospatial enterprise transition to next generation APIs.

This section provides background information on WFS 3.0, OpenAPI, Swagger, OpenID Connect and OAuth 2.0 and an introduction to key technologies tested.

5.1. Introduction to WFS 3.0 and OpenAPI

The foundation of WFS 3.0 is a set of interfaces which define the 'core' of the specification. The core provides a simple API to access geospatial feature resources as 'collections'. For example, this path -

```
GET /collections
```

lists the collections on the server that can be queried. GeoJSON is a recommended encoding for collections provided by WFS 3.0, along with HTML. The core specification supports several basic filters such as the familiar bbox, the ability to get geographic features and time. For example, this path -

```
GET /collections/agriculturepnt
```

returns a geospatial feature collection - which is something in real-world terrain (i.e. buildings or roads). Feature collections are typically described by geometries plus other properties. This approach provides a resource-oriented way to access geospatial features.

In this approach, the feature resource is accessed using HTTP verbs the same way they are used in HTTP itself. Unique operations like GetFeature or Update are no longer used to access or modify feature resources. Instead, we use HTTP methods like GET, POST, PUT, DELETE - which can make things easier for API and client application developers.

HTTP status codes are used to handle error situations, describe specific security schemes and other functions.

The WFS 3.0 approach is consistent with emerging OGC Web API Guidelines. It is also consistent with the resource-oriented approach described in the Testbed-12 OGC REST Users Guide summarized below.

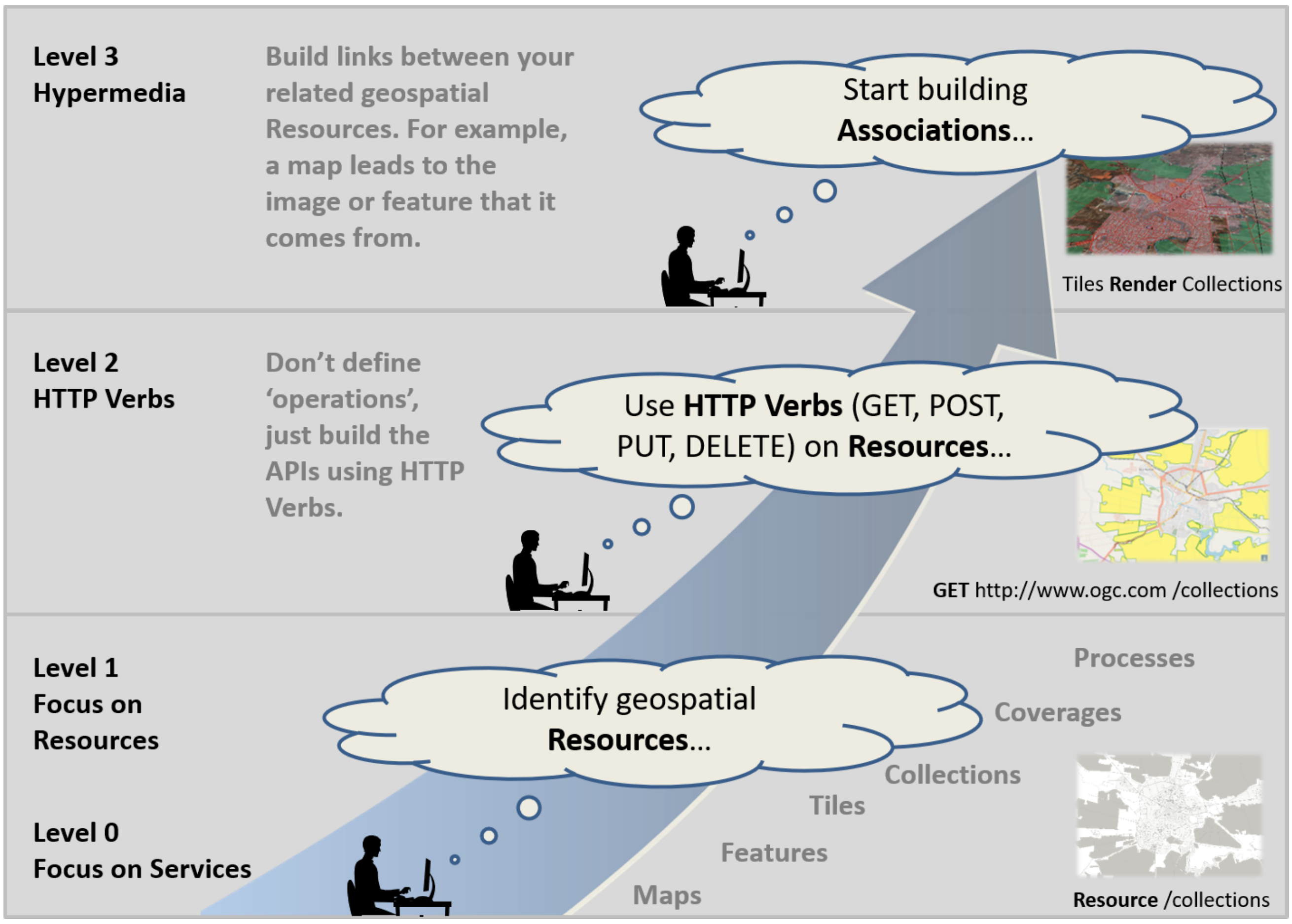
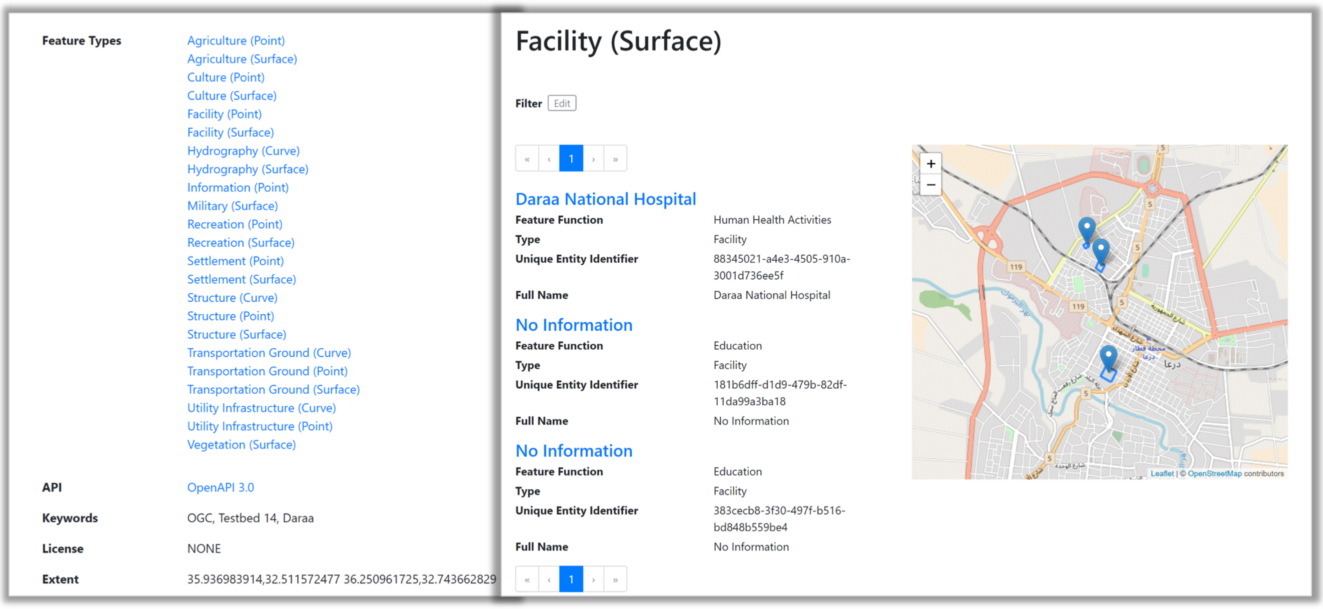

Level 3
Hypermedia
Build links between your related geospatial Resources. For example, a map leads to the image or feature that it comes from.
Start building Associations...
Tiles Render Collections
Level 2
HTTP Verbs
Don't define 'operations', just build the APIs using HTTP Verbs.
Use HTTP Verbs (GET, POST, PUT, DELETE) on Resources...
GET http://www.ogc.com /collections
Level 1
Focus on Resources
Identify geospatial Resources...
Processes
Coverages
Collections
Tiles
Features
Maps
Level 0
Focus on Services
Resource /collections

Figure 3. REST levels

Advanced functionality is separated into WFS 3.0 extensions. For example, vector tile delivery, transactions for updates and generalization of features at different resolutions are provided as WFS 3.0 extensions.

Each WFS 3.0 also deploys a landing page which provides easy-to-consume blocks of content describing the API, supported conformance classes and feature resources (collections) [1]. An example of Testbed-14 WFS 3.0 landing pages for the dataset over Daraa, Syria is shown below.



Feature Types	Agriculture (Point) Agriculture (Surface) Culture (Point) Culture (Surface) Facility (Point) Facility (Surface) Hydrography (Curve) Hydrography (Surface) Information (Point) Military (Surface) Recreation (Point) Recreation (Surface) Settlement (Point) Settlement (Surface) Structure (Curve) Structure (Point) Structure (Surface) Transportation Ground (Curve) Transportation Ground (Point) Transportation Ground (Surface) Utility Infrastructure (Curve) Utility Infrastructure (Point) Vegetation (Surface)
API	OpenAPI 3.0
Keywords	OGC, Testbed 14, Daraa
License	NONE
Extent	35.936983914,32.511572477 36.250961725,32.743662829

Facility (Surface)

Filter Edit

« ‹ 1 › »

Daraa National Hospital

Feature Function	Human Health Activities
Type	Facility
Unique Entity Identifier	88345021-a4e3-4505-910a-3001d736ee5f
Full Name	Daraa National Hospital

No Information

Feature Function	Education
Type	Facility
Unique Entity Identifier	181b6dff-d1d9-479b-82df-11da99a3ba18
Full Name	No Information

No Information

Feature Function	Education
Type	Facility
Unique Entity Identifier	383cecb8-3f30-497f-b516-bd848b559be4
Full Name	No Information

« ‹ 1 › »

Leaflet | © OpenStreetMap contributors

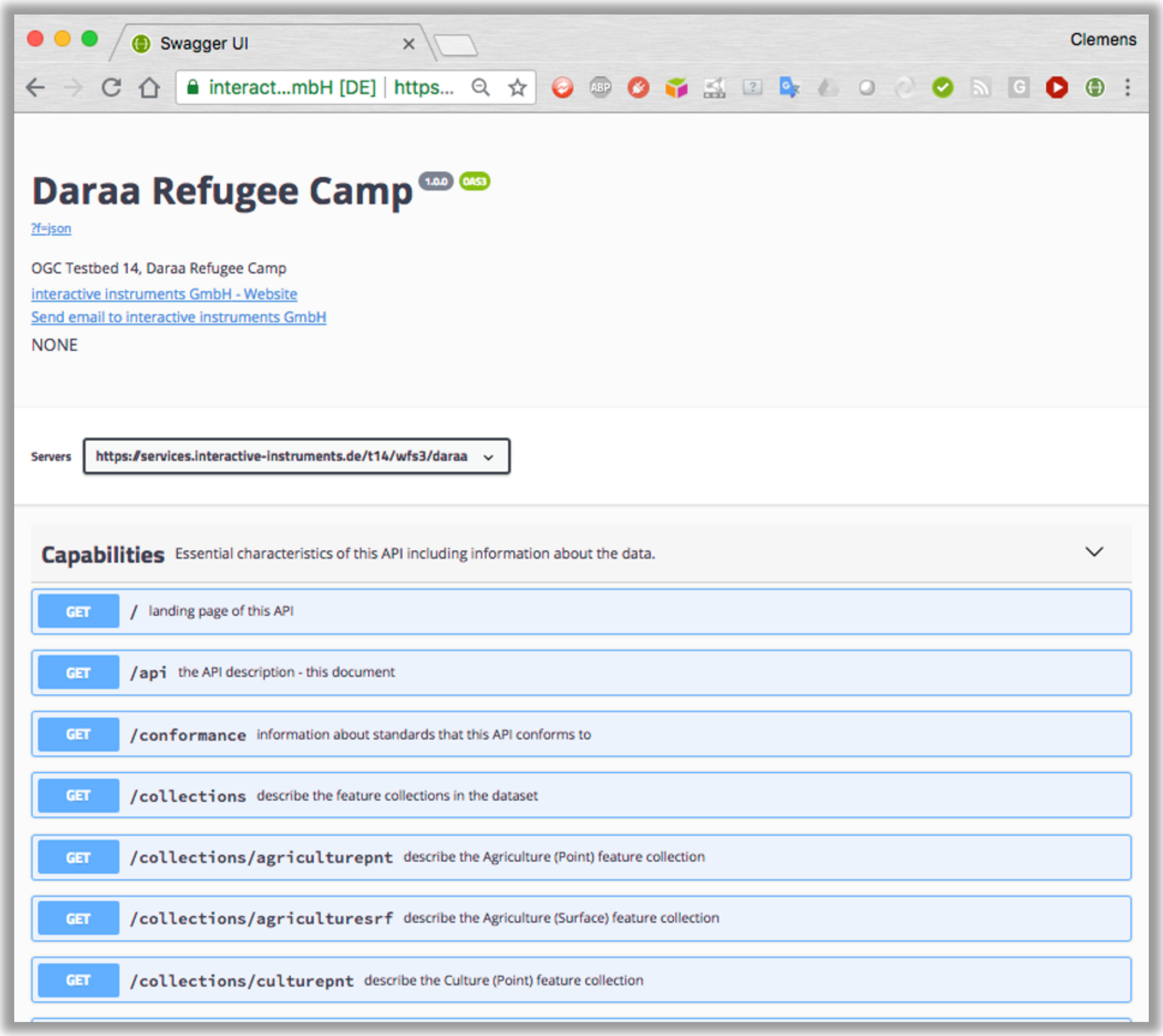
Figure 4. Example of Testbed-14 WFS 3.0 landing pages

The landing page is available at the 'root' path of a WFS 3.0. For example, it is simply this path -

```
GET /
```

WFS 3.0 minimizes the use of WFS-specific components as much as possible and instead uses industry standards that are commonly used by developers. The most important example of this is the use of OpenAPI documents instead of OGC-specific "Capabilities" documents. Accordingly, a key element of the initiative was an assessment of the ability of OpenAPI to provide a simple, language-agnostic description of the APIs. Testbed-14 also included TIEs to assess the ability of WFS 3.0 to support modern API specifications and tools such as Swagger.

An example of one of the Testbed-14 WFS 3.0 OpenAPI documents accessed in the Swagger UI is shown below.



Daraa Refugee Camp

?f=json

OGC Testbed 14, Daraa Refugee Camp

interactive instruments GmbH - Website

Send email to interactive instruments GmbH

NONE

Servers https://services.interactive-instruments.de/t14/wfs3/daraa

Capabilities Essential characteristics of this API including information about the data.

GET / landing page of this API

GET /api the API description - this document

GET /conformance information about standards that this API conforms to

GET /collections describe the feature collections in the dataset

GET /collections/agriculturepnt describe the Agriculture (Point) feature collection

GET /collections/agriculturesrf describe the Agriculture (Surface) feature collection

GET /collections/culturepnt describe the Culture (Point) feature collection

Figure 5. An example of one of the Testbed-14 WFS 3.0 OpenAPI documents

The OpenAPI document is available at the 'root' path of the API. For example, it is simply this path -

```
GET /api
```

OpenAPI and Swagger allow both humans and computers to understand the capabilities of the API without access to source code, documentation or through network traffic inspection. In addition, OpenAPI documents can be used by code generation tools to generate servers and clients in various programming languages.

5.1.1. Transitioning to Next-Generation APIs

WFS 3.0 with OpenAPI is intended to help the geospatial community 'break free of legacy' and implement modern approaches that align with current Web architecture and Spatial Data on the Web Best Practices. However, a cutover capability should be available for geospatial enterprises to provide transition and legacy support.

To ease the transition for geospatial enterprises, Testbed-14 began assessing WFS versioning and facades. For example, some WFS 3.0 implementations also supported WFS 2.x, 1.1 etc. and exposed this functionality through facades. Facades can represent legacy OGC web services as resource oriented WFS 3.0 APIs, easing transition.

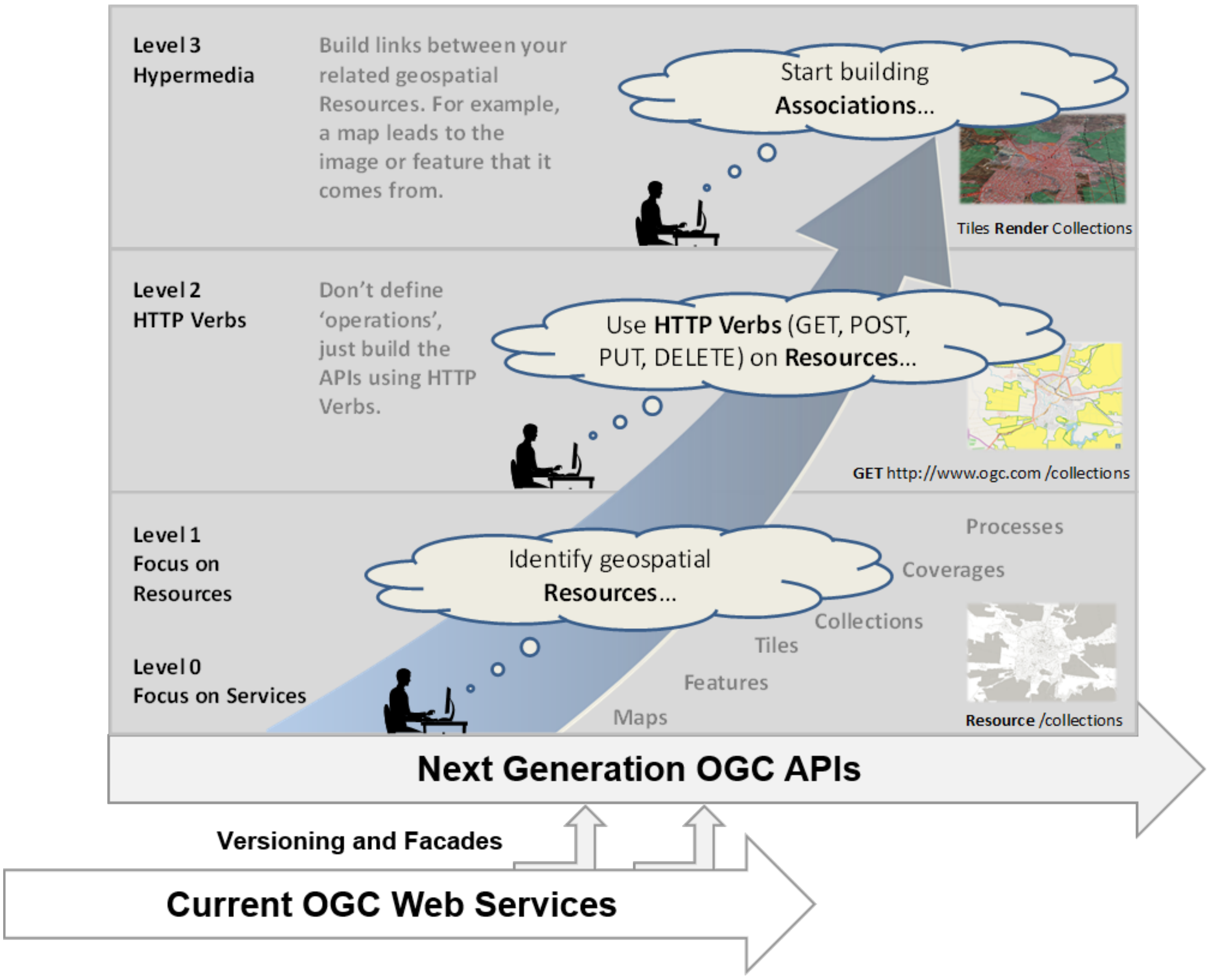
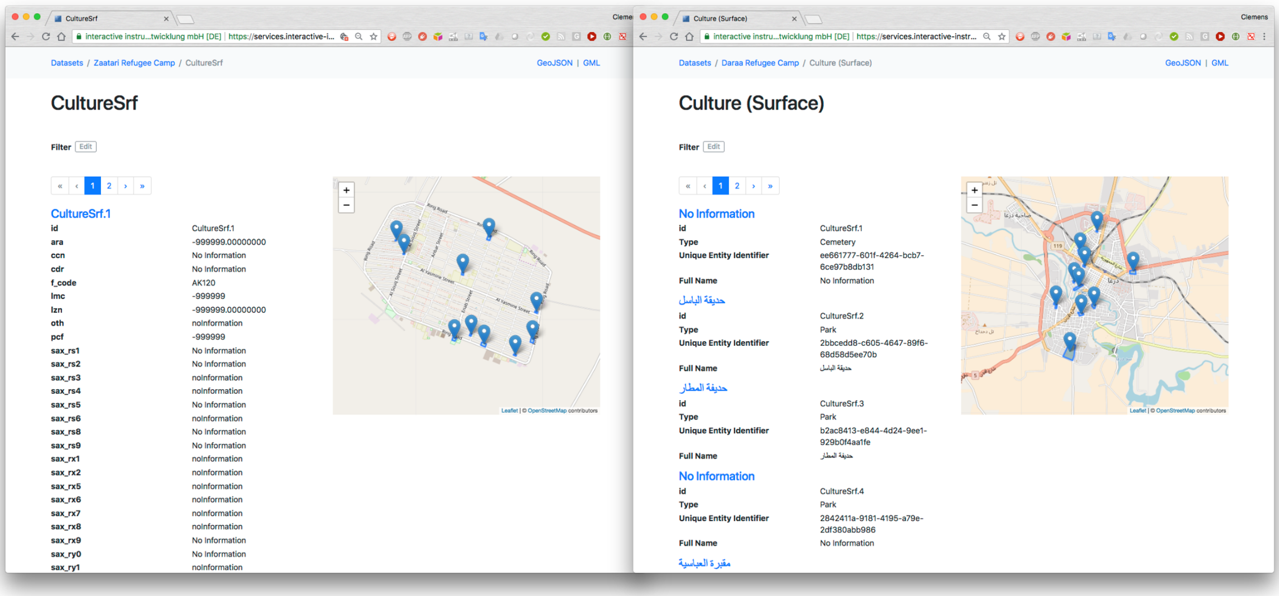

Level 3
Hypermedia
Build links between your related geospatial Resources. For example, a map leads to the image or feature that it comes from.
Start building Associations...
Tiles Render Collections
Level 2
HTTP Verbs
Don't define 'operations', just build the APIs using HTTP Verbs.
Use HTTP Verbs (GET, POST, PUT, DELETE) on Resources...
GET http://www.ogc.com /collections
Level 1
Focus on Resources
Identify geospatial Resources...
Processes
Coverages
Collections
Tiles
Features
Maps
Resource /collections
Level 0
Focus on Services
Next Generation OGC APIs
Versioning and Facades
Current OGC Web Services

Figure 6. Facades

In addition, legacy data models may be complex, with properties that are seldom used by data producers. This can needlessly increase the size and complexity of datasets and reduce the ease of implementation. To address this challenge some WFS 3.0 in Testbed-14 implemented the ability to simplify feature resources exposed.

For example, the screenshot on the left below is from the Zaatari dataset, without additional configuration. the Topographic Data Store (TDS) OpenStreepMap dataset is presented as published in the WFS 2.0 servers, and all properties are delivered to applications. The screenshot on the right shows the same feature resource in the Daraa dataset after the configuration of the WFS 3.0 proxy service to simplify the data. Properties that are never present are not shown, labels have been changed to more human-friendly text, code values have been translated and the name has been set up as the label of a feature.


CultureSrf
interactive instru...twicklung mbH [DE] | https://services.interactive-i...
Datasets / Zaatari Refugee Camp / CultureSrf
GeoJSON | GML
Filter Edit
« ‹ 1 2 › »
Leaflet | © OpenStreetMap contributors
Culture (Surface)
interactive instru...twicklung mbH [DE] | https://services.interactive-instr...
Datasets / Daraa Refugee Camp / Culture (Surface)
Clemens

CultureSrf

CultureSrf.1

id	CultureSrf.1
ara	-999999.00000000
ccn	No Information
cdr	No Information
f_code	AK120
lmc	-999999
lzn	-999999.00000000
oth	noInformation
pcf	-999999
sax_rs1	No Information
sax_rs2	No Information
sax_rs3	noInformation
sax_rs4	noInformation
sax_rs5	No Information
sax_rs6	noInformation
sax_rs8	No Information
sax_rs9	No Information
sax_rx1	noInformation
sax_rx2	noInformation
sax_rx5	noInformation
sax_rx6	noInformation
sax_rx7	noInformation
sax_rx8	noInformation
sax_rx9	No Information
sax_ry0	No Information
sax_ry1	noInformation

Culture (Surface)

No Information

id	CultureSrf.1
Type	Cemetery
Unique Entity Identifier	ee661777-601f-4264-bcb7-6ce97b8db131
Full Name	No Information

حديقة الباسل

id	CultureSrf.2
Type	Park
Unique Entity Identifier	2bbcedd8-c605-4647-89f6-68d58d5ee70b
Full Name	حديقة الباسل

حديقة المطار

id	CultureSrf.3
Type	Park
Unique Entity Identifier	b2ac8413-e844-4d24-9ee1-929b0f4aa1fe
Full Name	حديقة المطار

No Information

id	CultureSrf.4
Type	Park
Unique Entity Identifier	2842411a-9181-4195-a79e-2df380abb986
Full Name	No Information

مقبرة العباسية

Figure 7. Zaatari and Daraa dataset screenshots

This section provided a brief introduction to WFS 3.0. Additional information on testing, implementations and security frameworks in Testbed-14 are provided in the Experiments and Implementations sections of this report.

5.2. OpenID Connect and OAuth 2.0

OpenAPI on WFS 3.0 supports multiple security frameworks. For Testbed-14, OpenID Connect and OAuth 2.0 were assessed. OpenID Connect is an authentication layer on top of OAuth 2.0, an authorization framework. OpenID Connect specifies a RESTful API using JSON as a data format. This section provides background information on OpenID Connect and OAuth 2.0 and an introduction to key technologies tested.

OAuth 2.0 provides a standard protocol for authorization. OAuth 2.0 supersedes the work done on the original OAuth protocol created in 2006. OAuth 2.0 focuses on client developer simplicity while providing specific authorization flows for web applications, desktop applications, mobile phones, and living room devices.

OpenID Connect (OIDC) provides a simple identity layer on top of the OAuth 2.0 protocol. It allows Clients to verify the identity of the End-User based on the authentication performed by an

Authorization Server, as well as to obtain basic profile information about the End-User in an interoperable and REST-like manner. OpenID Connect allows clients of all types, including Web-based, mobile, and JavaScript clients, to request and receive information about authenticated sessions and end-users. The specification suite is extensible, allowing participants to use optional features such as encryption of identity data, discovery of OpenID Providers, and session management, when it makes sense for them.

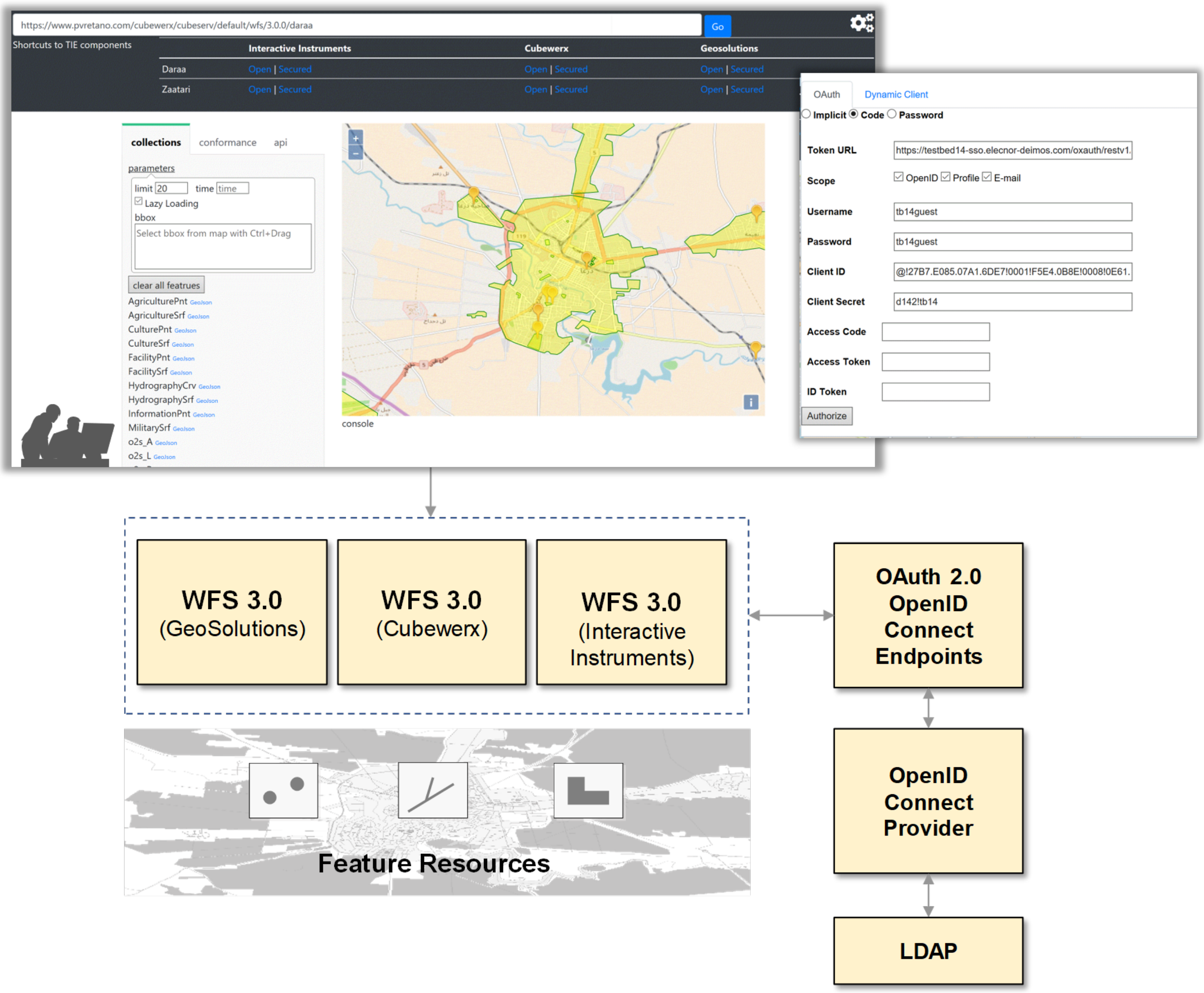
5.2.1. OpenID Connect Security Environment

For Testbed-14, participants implemented an Authorization Server, which needs to be compliant with OAuth2.0 and OpenID Connect.

The authorization server adds registration and management capabilities (that populate the LDAP service) on top of the authorization endpoints and makes information about the service available by means of an OpenID Connect Discovery document that lies on a commonly used relative path (/.well-known/openid-configuration).

The Basic Authentication Service is made available for potential use on profile management, and can either use LDAP credentials or utilize the authentication and authorization scheme. The end-users can then interact with clients to retrieve tokens and access services. Additional information on the Testbed-14 OpenID Connect security environment is provided in the OGC Testbed-14 Security Engineering Report.

The configuration of OAuth2.0 and OpenID Connect in the Next Generation APIs - WFS 3.0 component implementation design is shown below. The client application with security handling is provided by GIS.FCU. The WFS 3.0 are provided by Interactive Instruments, GeoSolutions and Cubewerx. The authorization server is provided by Deimos.


https://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/daraa

Shortcuts to TIE components	Interactive Instruments	Cubewerx	Geosolutions
Daraa	Open | Secured	Open | Secured	Open | Secured
Zaatari	Open | Secured	Open | Secured	Open | Secured

collections conformance api

parameters

limit 20 time time

☑ Lazy Loading

bbox

Select bbox from map with Ctrl+Drag

clear all featrues

AgriculturePnt GeoJson
AgricultureSrf GeoJson
CulturePnt GeoJson
CultureSrf GeoJson
FacilityPnt GeoJson
FacilitySrf GeoJson
HydrographyCrv GeoJson
HydrographySrf GeoJson
InformationPnt GeoJson
MilitarySrf GeoJson
o2s_A GeoJson
o2s_L GeoJson

console

OAuth Dynamic Client

○ Implicit ● Code ○ Password

Token URL	https://testbed14-sso.elecnor-deimos.com/oauth/restv1.
Scope	☑ OpenID ☑ Profile ☑ E-mail
Username	tb14guest
Password	tb14guest
Client ID	@!27B7.E085.07A1.6DE7!0001!F5E4.0B8E!0008!0E61.
Client Secret	d142!tb14
Access Code	
Access Token	
ID Token	

Authorize

Figure 8. Configuration of OAuth2.0 and OpenID Connect in the WFS 3.0 implementation design

This section provided a brief introduction to OAuth2.0 and OpenID Connect. Additional information on testing, implementations and security frameworks in Testbed-14 are provided in the Experiments and Implementations sections of this report.

Chapter 6. Experiments

The objective of the Next Generation APIs - WFS 3.0 effort in Testbed-14 was to develop and test Web Feature Services (WFS) version 3.0 and its draft extensions in a secured environment.

This section describes tests conducted by Testbed-14 participants implementing WFS 3.0 using OpenAPI, secured by OpenID Connect and OAuth 2.0. It includes descriptions and testing results of APIs to help simplify and secure access to geospatial feature data. It also describes how a WFS 3.0 can advertise security requirements for accessing resources in its OpenAPI document.

6.1. Demonstration Scenario

In OGC Testbed-14, participants assessed the ability of WFS 3.0 to support simulated users in a humanitarian relief scenario. In this scenario large numbers of people are displaced from the Daraa region of Syria to Zaatari refugee camp due to conflict. Understanding the situation, the infrastructure and helping refugees is a challenge. To assist this effort users accessed OpenStreetMap data converted into the NGA Topographic Data Store (TDS) model to better understand the environment and infrastructure.

The following graphic provides a sample of the scenario and data involved in testing WFS 3.0.

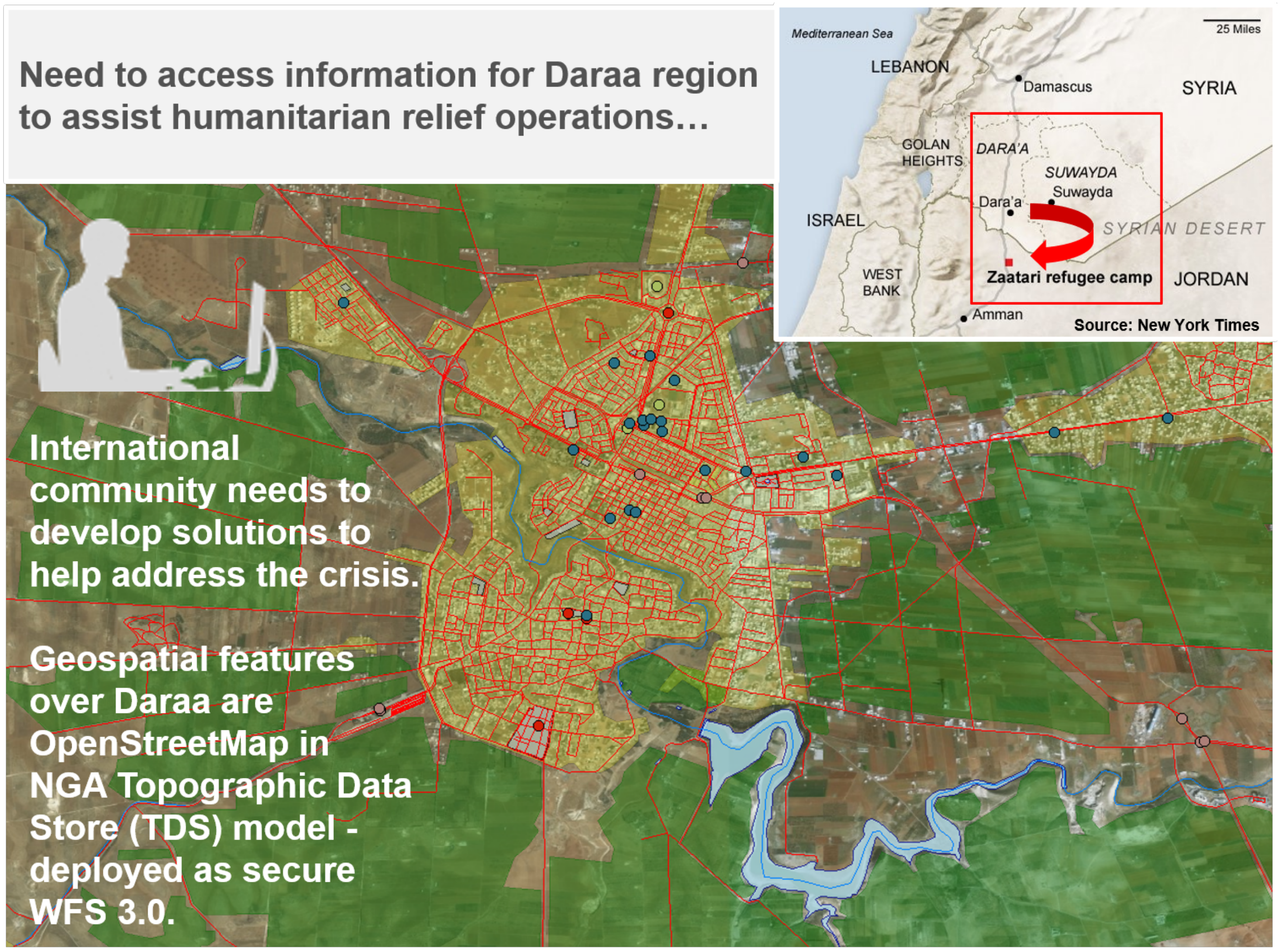

Need to access information for Daraa region to assist humanitarian relief operations…
International community needs to develop solutions to help address the crisis.
Geospatial features over Daraa are OpenStreetMap in NGA Topographic Data Store (TDS) model - deployed as secure WFS 3.0.
Mediterranean Sea
25 Miles
LEBANON
Damascus
SYRIA
GOLAN HEIGHTS
DARA'A
SUWAYDA
Suwayda
Dara'a
ISRAEL
SYRIAN DESERT
WEST BANK
Zaatari refugee camp
JORDAN
Amman
Source: New York Times

Figure 9. The scenario and data involved in testing WFS 3.0

6.2. Component Implementation Design

Based on the requirements the WFS 3.0 test architecture was developed and is shown below.

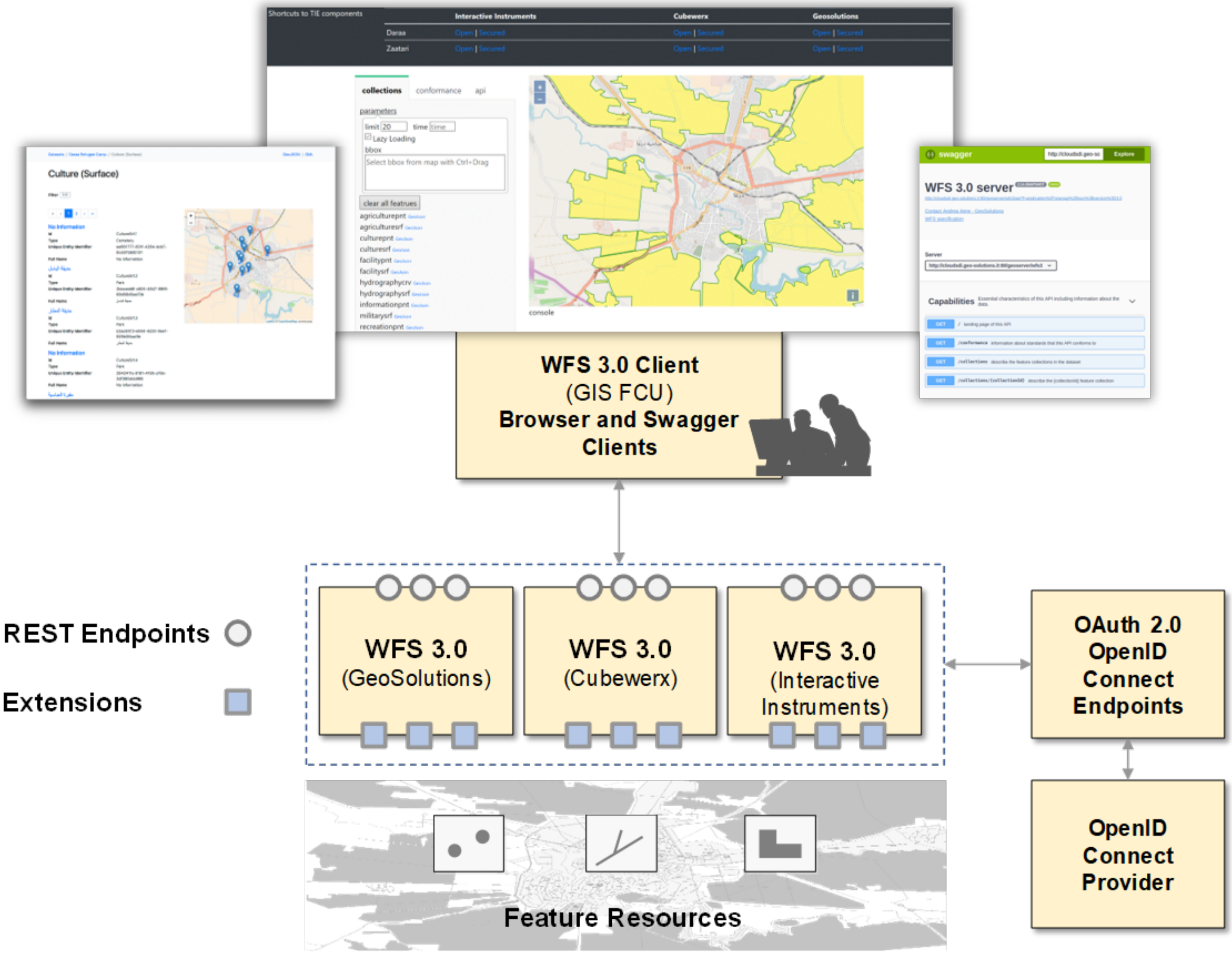


Figure 10. Overview of the Testbed-14 Next Generation APIs experiments architecture

The test architecture illustrates a sequence of interactions between APIs, client applications and security frameworks for this part of the OGC Testbed-14 -

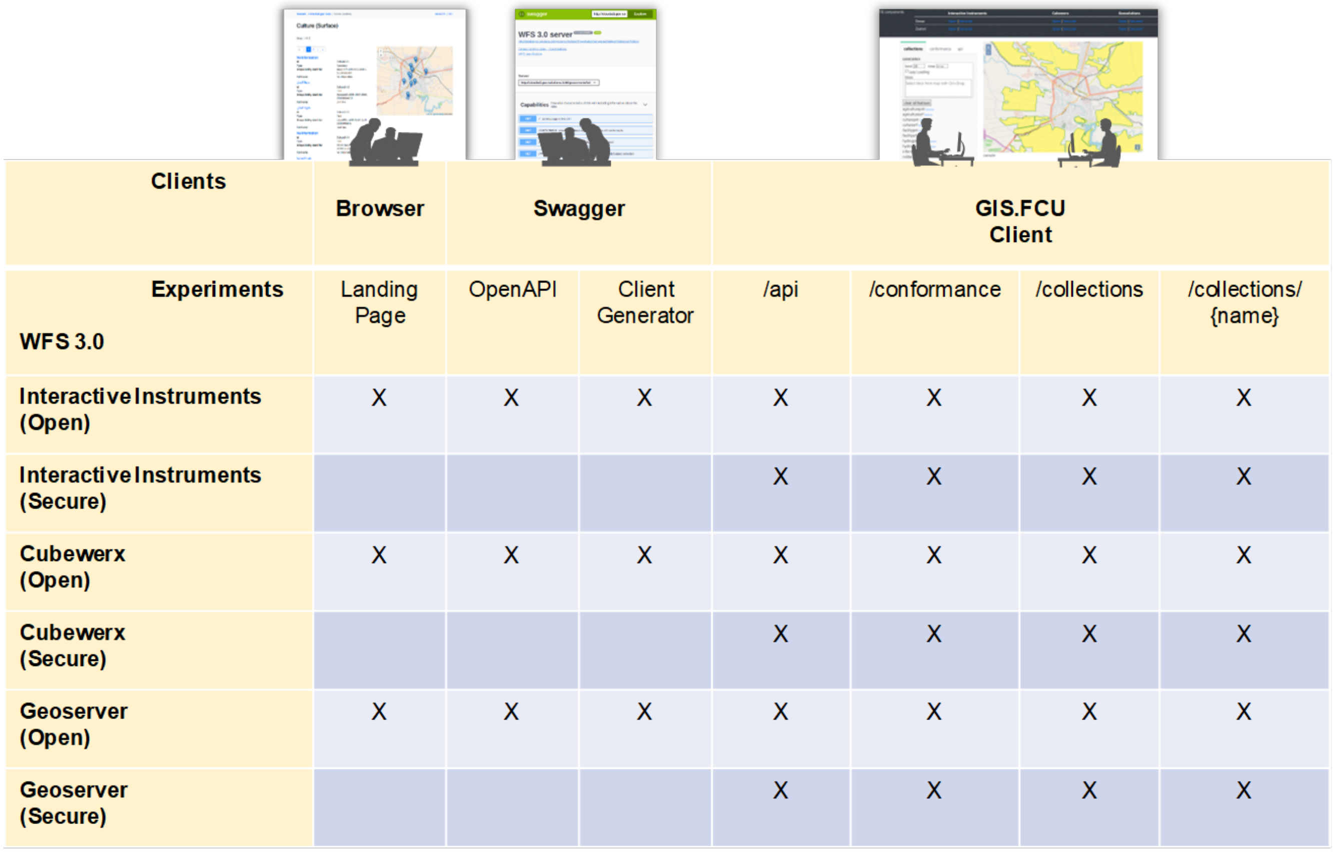
- **APIs** - NGA TDS OpenStreetMap data for the Daraa and Zaatari areas was deployed on multiple OGC WFS 3.0s including services from GeoSolutions, Interactive Instruments and Cubewerx. The WFS 3.0s focused on a simple RESTful core specified as reusable OpenAPI components, with WFS 3.0 Extensions for Generalization, Maps and other functions implemented by different participants.
- **Client Applications** - Several types of client applications were exercised including a client from GIS.FCU designed to exercise the reusable OpenAPI components of WFS 3.0 core. In addition, off-the-shelf web browsers were used to access the Landing Page paths of the WFS 3.0. Finally, the Swagger UI was tested as a client to visualize and interact with the resources of the OpenAPI implementations - and to automatically generate a variety of clients in HTML and other languages.
- **Security Frameworks** - WFS 3.0 clients and services were secured via an Authorization Server using OpenID Connect and OAuth 2.0. Client applications logged in using credentials valid on the Authorization Server and then performed requests to the WFS 3.0. If the credentials are

valid the user is granted access to the WFS 3.0 via the client application.

6.3. Technology Integration Experiments

The architecture was tested in a series of Technology Integration Experiments (TIEs) and demonstrated in the context of unsecured APIs and Clients and secure APIs and Clients. Unsecured APIs and Clients did not implement the security framework and secure APIs and Clients implemented OpenID Connect and OAuth 2.0.

Interoperability experiments conducted are outlined in the following TIE table -



Clients	Browser	Swagger		GIS.FCU Client			
Experiments WFS 3.0	Landing Page	OpenAPI	Client Generator	/api	/conformance	/collections	/collections/{name}
Interactive Instruments (Open)	X	X	X	X	X	X	X
Interactive Instruments (Secure)				X	X	X	X
Cubewerx (Open)	X	X	X	X	X	X	X
Cubewerx (Secure)				X	X	X	X
Geoserver (Open)	X	X	X	X	X	X	X
Geoserver (Secure)				X	X	X	X

Figure 11. Technology Integration Experiments

The following sections describe experiments conducted with unsecured and secured WFS 3.0 APIs and Clients.

6.3.1. Unsecured WFS 3.0

This section describes TIEs on Services and Clients that did not implement the security framework of OpenID Connect and OAuth 2.0. Testing included assessments of the OpenAPI implementations and the ability of WFS 3.0 to support modern API tools such as Swagger.

The following sections describe results of TIEs for landing pages, OpenAPIs, core conformance classes conducted for WFS 3.0 during Testbed-14.

Browser Tests

In the first set of Technology Integration Experiments, standard web browsers were used to access the 'root path' of WFS 3.0. For example, if an application or user follows the root path of a WFS 3.0 such as

```
GET /
```

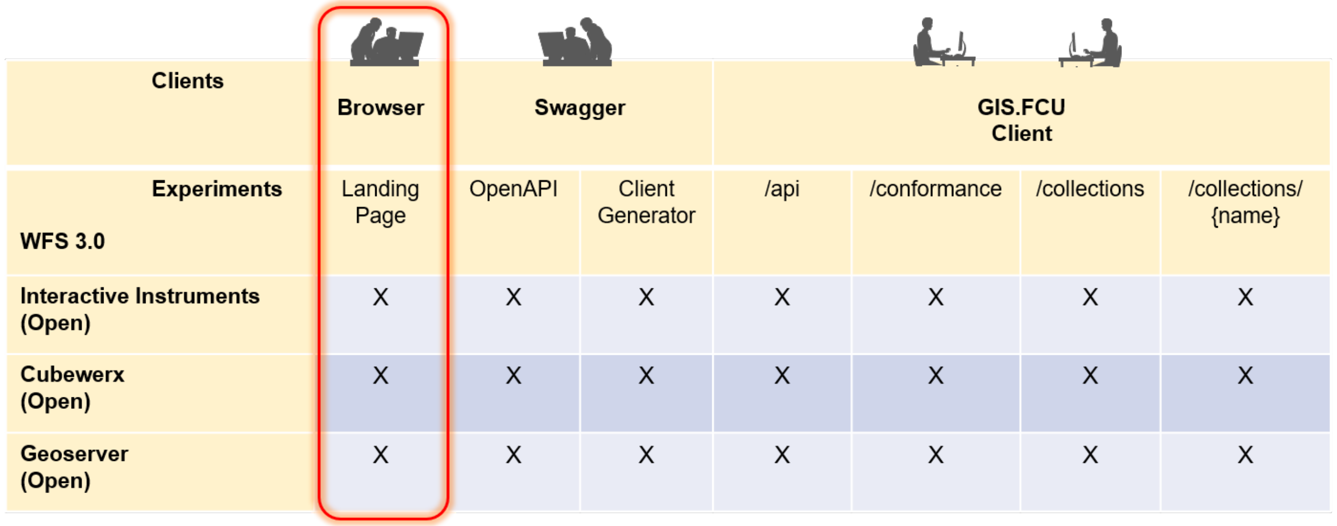
The server responds with a 'landing page' that includes links to the following resources

```
/api

/conformance

/collections
```

WFS 3.0 from GeoSolutions, Cubewerx and Interactive Instruments were assessed for the ability to provide landing pages at the root path of each API. To access the WFS 3.0 landing pages a standard web browser was used to query the path and conduct the TIE -

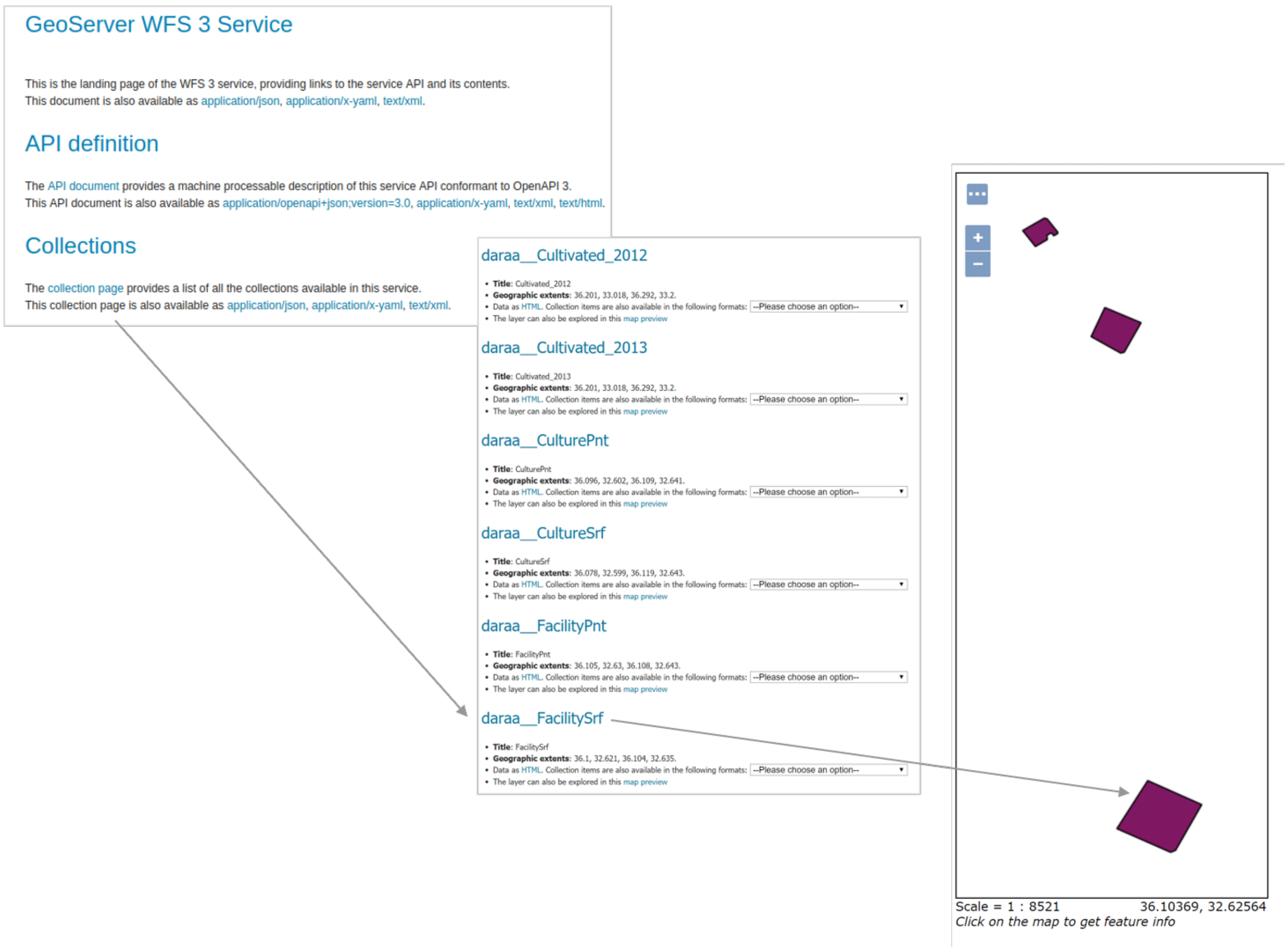


Clients	Browser	Swagger		GIS.FCU Client			
Experiments WFS 3.0	Landing Page	OpenAPI	Client Generator	/api	/conformance	/collections	/collections/{name}
Interactive Instruments (Open)	X	X	X	X	X	X	X
Cubewerx (Open)	X	X	X	X	X	X	X
Geoserver (Open)	X	X	X	X	X	X	X

Figure 12. Browser Technology Integration Experiments

Landing pages were successfully provided by WFS 3.0 from GeoSolutions, Cubewerx and Interactive Instruments. Each WFS 3.0 Landing Page provided information on the API, conformance classes and collections.

In the GeoServer WFS 3.0 example below, the HTML landing page has basically the same contents as a JSON landing page, just in human readable form and with alternative format links for every entry. In the collections section the daraa_FacilitySrf resource is selected and a simple rendering of a map preview is displayed in the browser.



GeoServer WFS 3 Service

This is the landing page of the WFS 3 service, providing links to the service API and its contents.
This document is also available as application/json, application/x-yaml, text/xml.

API definition

The API document provides a machine processable description of this service API conformant to OpenAPI 3.
This API document is also available as application/openapi+json;version=3.0, application/x-yaml, text/xml, text/html.

Collections

The collection page provides a list of all the collections available in this service.
This collection page is also available as application/json, application/x-yaml, text/xml.

daraa__Cultivated_2012

- **Title**: Cultivated_2012
- **Geographic extents**: 36.201, 33.018, 36.292, 33.2.
- Data as HTML. Collection items are also available in the following formats: --Please choose an option--
- The layer can also be explored in this map preview

daraa__Cultivated_2013

- **Title**: Cultivated_2013
- **Geographic extents**: 36.201, 33.018, 36.292, 33.2.
- Data as HTML. Collection items are also available in the following formats: --Please choose an option--
- The layer can also be explored in this map preview

daraa__CulturePnt

- **Title**: CulturePnt
- **Geographic extents**: 36.096, 32.602, 36.109, 32.641.
- Data as HTML. Collection items are also available in the following formats: --Please choose an option--
- The layer can also be explored in this map preview

daraa__CultureSrf

- **Title**: CultureSrf
- **Geographic extents**: 36.078, 32.599, 36.119, 32.643.
- Data as HTML. Collection items are also available in the following formats: --Please choose an option--
- The layer can also be explored in this map preview

daraa__FacilityPnt

- **Title**: FacilityPnt
- **Geographic extents**: 36.105, 32.63, 36.108, 32.643.
- Data as HTML. Collection items are also available in the following formats: --Please choose an option--
- The layer can also be explored in this map preview

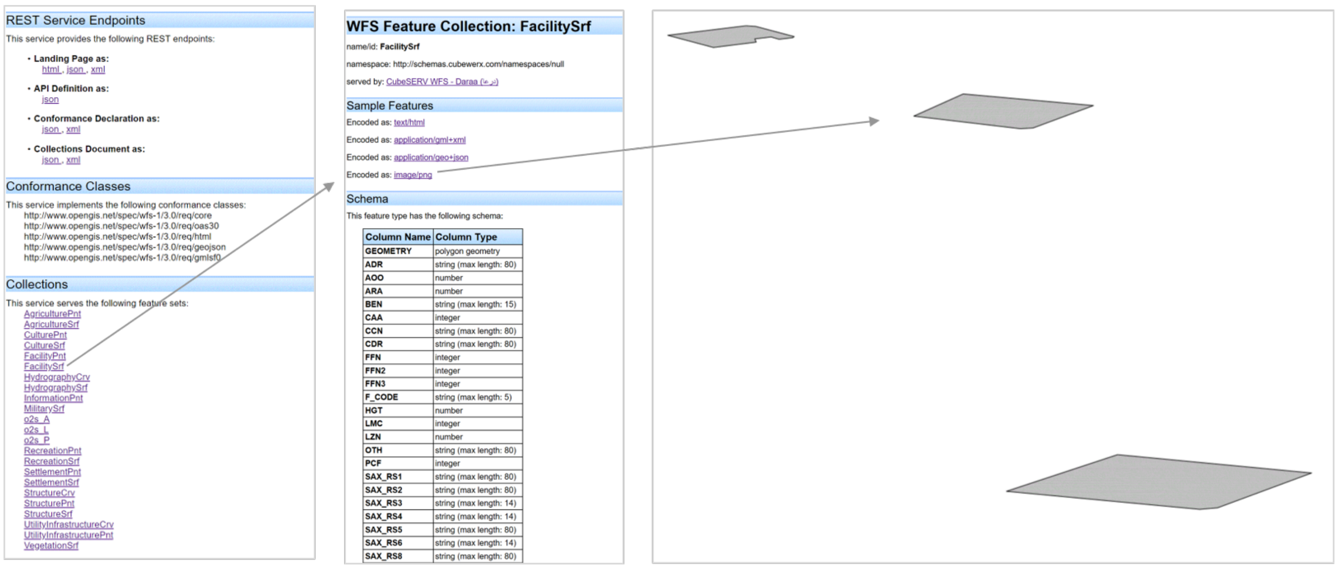
daraa__FacilitySrf

- **Title**: FacilitySrf
- **Geographic extents**: 36.1, 32.621, 36.104, 32.635.
- Data as HTML. Collection items are also available in the following formats: --Please choose an option--
- The layer can also be explored in this map preview

Scale = 1 : 8521 36.10369, 32.62564
Click on the map to get feature info

Figure 13. GeoSolutions landing pages

In the Cubewerx example below the landing page is also HTML but is laid out in a slightly different form, based on the preferences of the API developer. In the collections section the FacilitySrf resource is once again selected, and this time a jpeg image preview is displayed in the browser.



REST Service Endpoints

This service provides the following REST endpoints:

- **Landing Page as:** html , json , xml
- **API Definition as:** json
- **Conformance Declaration as:** json , xml
- **Collections Document as:** json , xml

Conformance Classes

This service implements the following conformance classes:
http://www.opengis.net/spec/wfs-1/3.0/req/core
http://www.opengis.net/spec/wfs-1/3.0/req/oas30
http://www.opengis.net/spec/wfs-1/3.0/req/html
http://www.opengis.net/spec/wfs-1/3.0/req/geojson
http://www.opengis.net/spec/wfs-1/3.0/req/gmlsf0

Collections

This service serves the following feature sets:
AgriculturePnt
AgricultureSrf
CulturePnt
CultureSrf
FacilityPnt
FacilitySrf
HydrographyCrv
HydrographySrf
InformationPnt
MilitarySrf
o2s_A
o2s_L
o2s_P
RecreationPnt
RecreationSrf
SettlementPnt
SettlementSrf
StructureCrv
StructurePnt
StructureSrf
UtilityInfrastructureCrv
UtilityInfrastructurePnt
VegetationSrf

WFS Feature Collection: FacilitySrf

name/id: **FacilitySrf**

namespace: http://schemas.cubewerx.com/namespaces/null

served by: CubeSERV WFS - Daraa (درعا)

Sample Features

Encoded as: text/html

Encoded as: application/gml+xml

Encoded as: application/geo+json

Encoded as: image/png

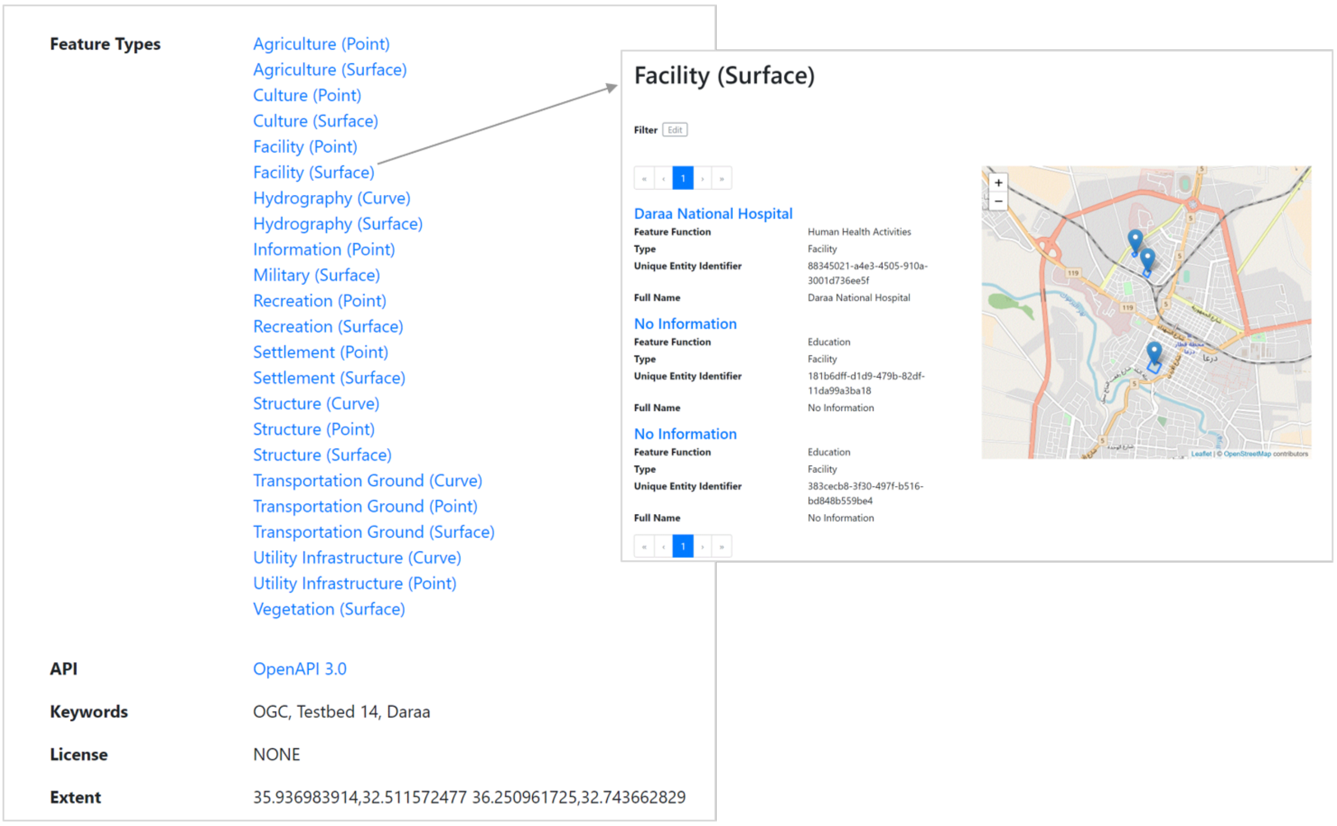
Schema

This feature type has the following schema:

Column Name	Column Type
GEOMETRY	polygon geometry
ADR	string (max length: 80)
AOO	number
ARA	number
BEN	string (max length: 15)
CAA	integer
CCN	string (max length: 80)
CDR	string (max length: 80)
FFN	integer
FFN2	integer
FFN3	integer
F_CODE	string (max length: 5)
HGT	number
LMC	integer
LZN	number
OTH	string (max length: 80)
PCF	integer
SAX_RS1	string (max length: 80)
SAX_RS2	string (max length: 80)
SAX_RS3	string (max length: 14)
SAX_RS4	string (max length: 14)
SAX_RS5	string (max length: 80)
SAX_RS6	string (max length: 14)
SAX_RS8	string (max length: 80)

Figure 14. Cubewerx landing pages

In the Interactive Instruments example below the WFS 3.0 landing page presents another type of layout, but again with the same information. In the collections section the FacilitySrf resource is once again selected, and this time markers on an OpenStreetMap background map are displayed showing the locations of the feature resources.



Feature Types	Agriculture (Point) Agriculture (Surface) Culture (Point) Culture (Surface) Facility (Point) Facility (Surface) Hydrography (Curve) Hydrography (Surface) Information (Point) Military (Surface) Recreation (Point) Recreation (Surface) Settlement (Point) Settlement (Surface) Structure (Curve) Structure (Point) Structure (Surface) Transportation Ground (Curve) Transportation Ground (Point) Transportation Ground (Surface) Utility Infrastructure (Curve) Utility Infrastructure (Point) Vegetation (Surface)
API	OpenAPI 3.0
Keywords	OGC, Testbed 14, Daraa
License	NONE
Extent	35.936983914,32.511572477 36.250961725,32.743662829

Facility (Surface)

Filter Edit

Daraa National Hospital

Feature Function	Human Health Activities
Type	Facility
Unique Entity Identifier	88345021-a4e3-4505-910a-3001d736ee5f
Full Name	Daraa National Hospital

No Information

Feature Function	Education
Type	Facility
Unique Entity Identifier	181b6dff-d1d9-479b-82df-11da99a3ba18
Full Name	No Information

No Information

Feature Function	Education
Type	Facility
Unique Entity Identifier	383cecb8-3f30-497f-b516-bd848b559be4
Full Name	No Information

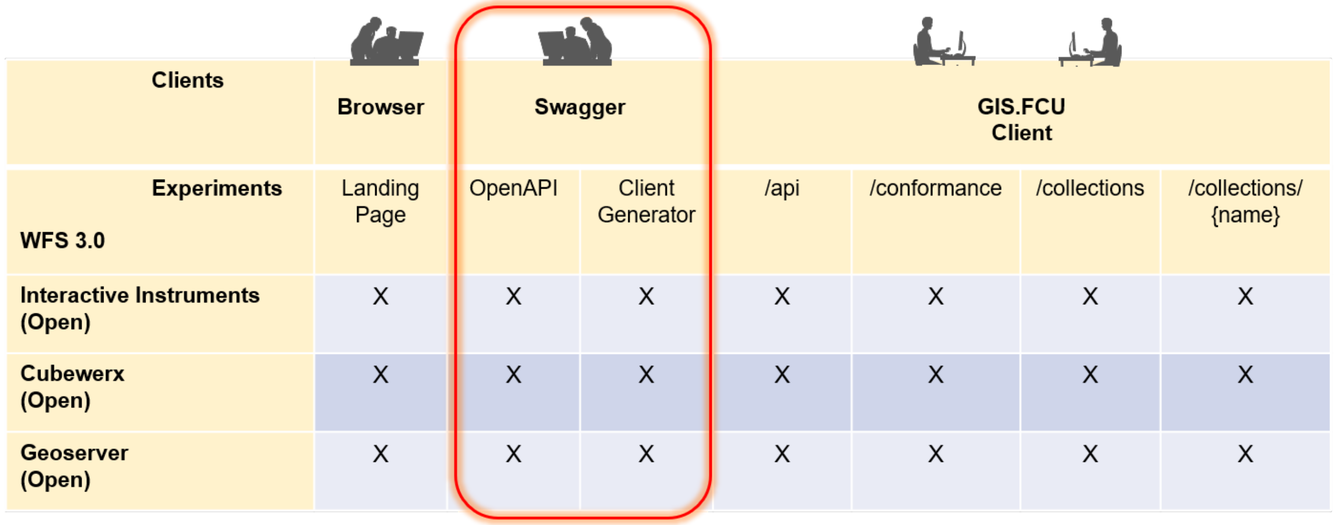
Figure 15. Interactive Instruments landing pages

Swagger UI Tests

WFS 3.0 minimizes the use of WFS-specific components as much as possible and instead uses industry standards that are commonly used by developers. The most important example of this is the use of OpenAPI documents instead of OGC-specific "Capabilities" documents. Some key elements of the initiative were assessments of the ability of OpenAPI to provide a simple, language-agnostic interface to RESTful APIs. Accordingly, Testbed-14 also included TIEs to assess the ability of WFS 3.0 to support modern API tools such as Swagger.

It should be noted that Swagger is no the only OpenAPI tool. Other tools include KaiZen, Microsoft parsers, Apicurio, RepreZen, Google Gnostic and many others. For example, the CITE test for WFS 3.0 uses KaiZen to parse OpenAPI documents into Java classes which are then used in compliance tests.

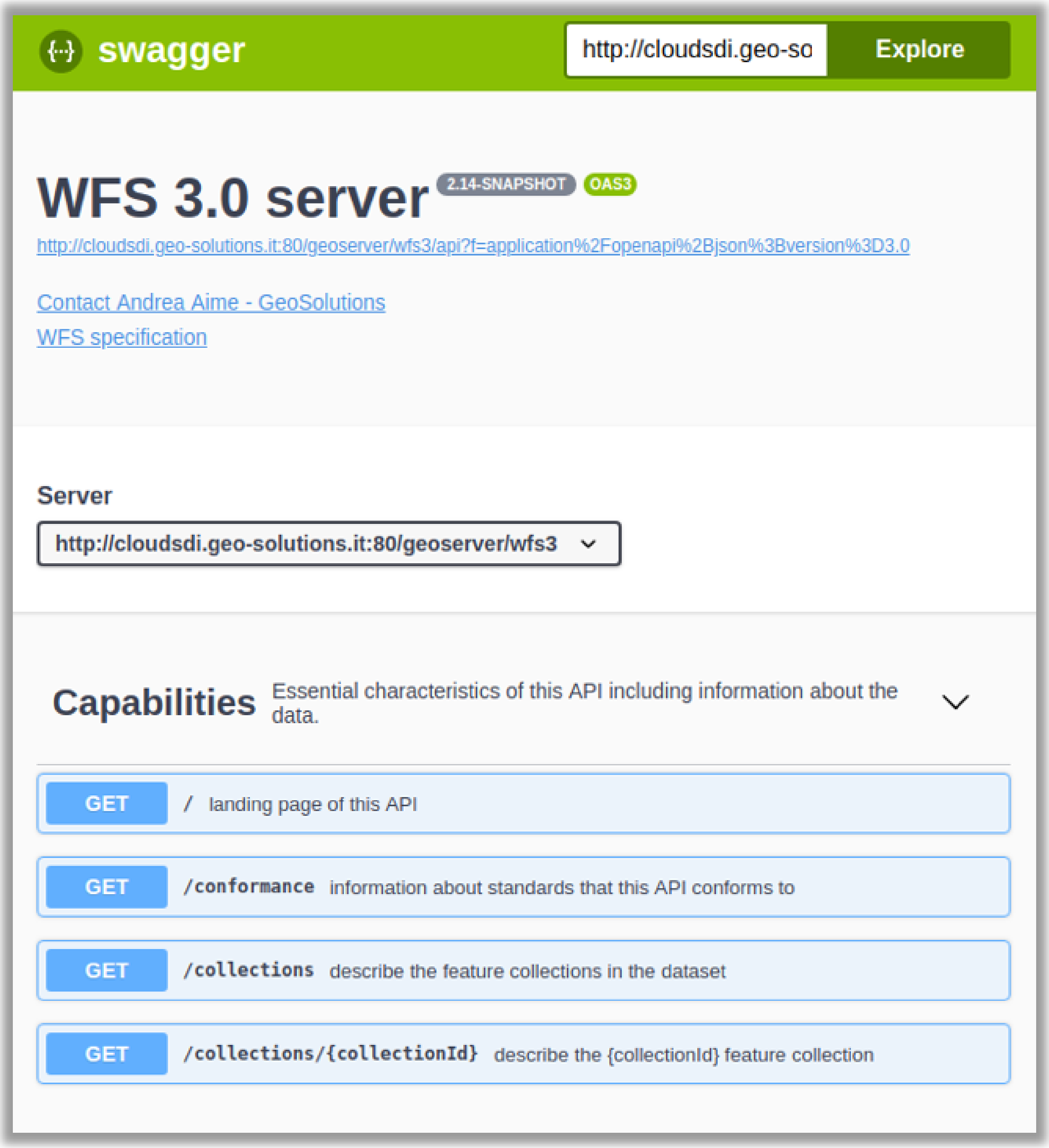
This section provides a brief description of Technology Integration Experiments using Swagger tools to access OpenAPI documents describing the WFS 3.0.



Clients	Browser	Swagger		GIS.FCU Client			
Experiments WFS 3.0	Landing Page	OpenAPI	Client Generator	/api	/conformance	/collections	/collections/{name}
Interactive Instruments (Open)	X	X	X	X	X	X	X
Cubewerx (Open)	X	X	X	X	X	X	X
Geoserver (Open)	X	X	X	X	X	X	X

Figure 16. Swagger Technology Integration Experiments

In Testbed-14 Swagger tools successfully accessed OpenAPI documents provided by WFS 3.0 from GeoSolutions, Cubewerx and Interactive Instruments. In the GeoSolutions example below the resource path for the OpenAPI document is provided to a browser-based Swagger application and the tool returns an easy to understand description of the API components.


swagger

http://cloudsdi.geo-so

Explore

WFS 3.0 server 2.14-SNAPSHOT OAS3

http://cloudsdi.geo-solutions.it:80/geoserver/wfs3/api?f=application%2Fopenapi%2Bjson%3Bversion%3D3.0

Contact Andrea Aime - GeoSolutions

WFS specification

Server

http://cloudsdi.geo-solutions.it:80/geoserver/wfs3

Capabilities Essential characteristics of this API including information about the data.

GET / landing page of this API

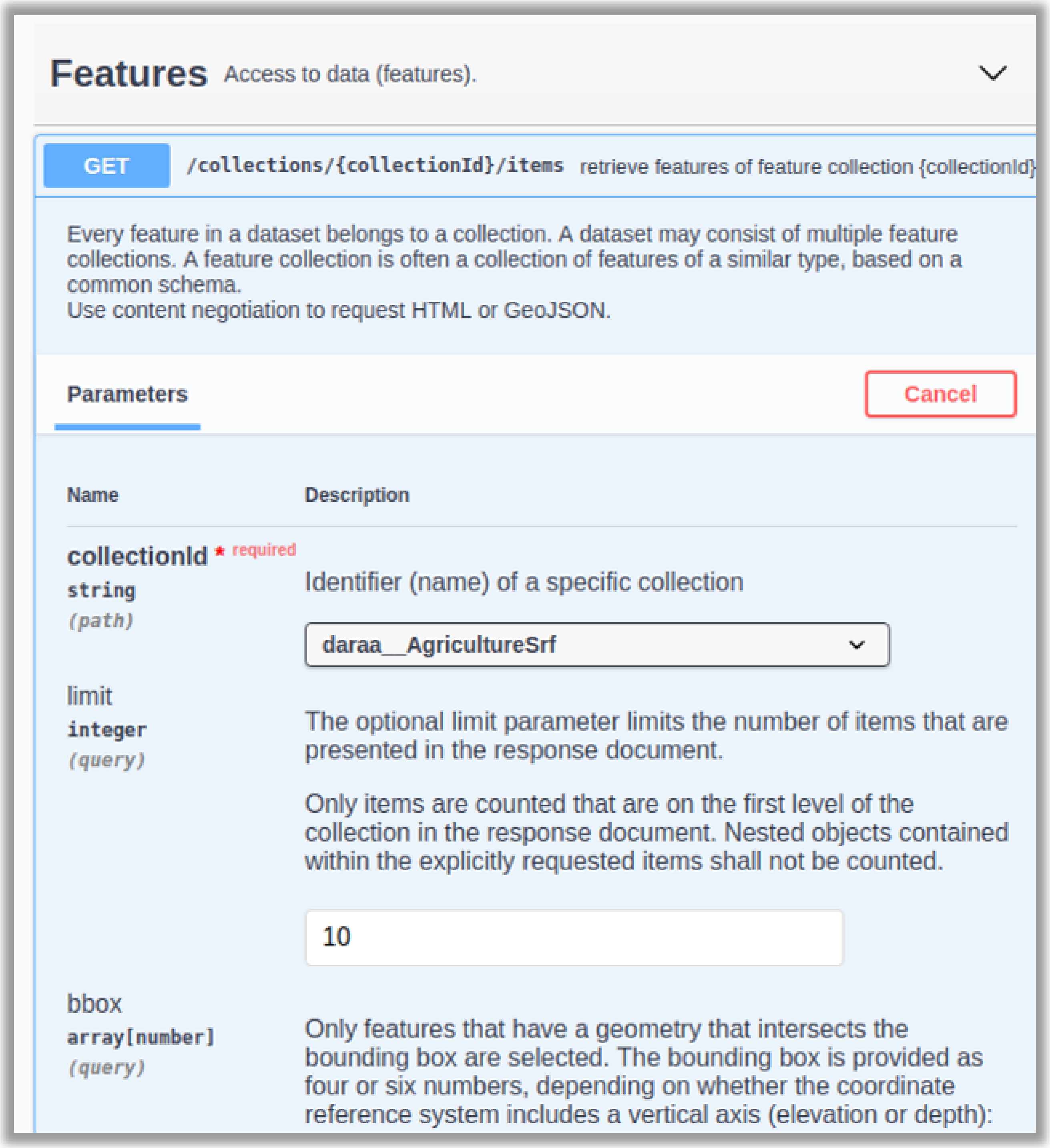
GET /conformance information about standards that this API conforms to

GET /collections describe the feature collections in the dataset

GET /collections/{collectionId} describe the {collectionId} feature collection

Figure 17. GeoSolutions GeoServer Swagger page

In the GeoServer WFS 3.0 example below the /collections/ { collectionsid} API component is accessed and descriptions of the feature collections provided in the Swagger tool.



/collections/{collectionId}/items retrieve features of feature collection {collectionId}

Every feature in a dataset belongs to a collection. A dataset may consist of multiple feature collections. A feature collection is often a collection of features of a similar type, based on a common schema.
Use content negotiation to request HTML or GeoJSON.

collectionId * required
string
(path)
Identifier (name) of a specific collection
daraa__AgricultureSrf

limit
integer
(query)
The optional limit parameter limits the number of items that are presented in the response document.

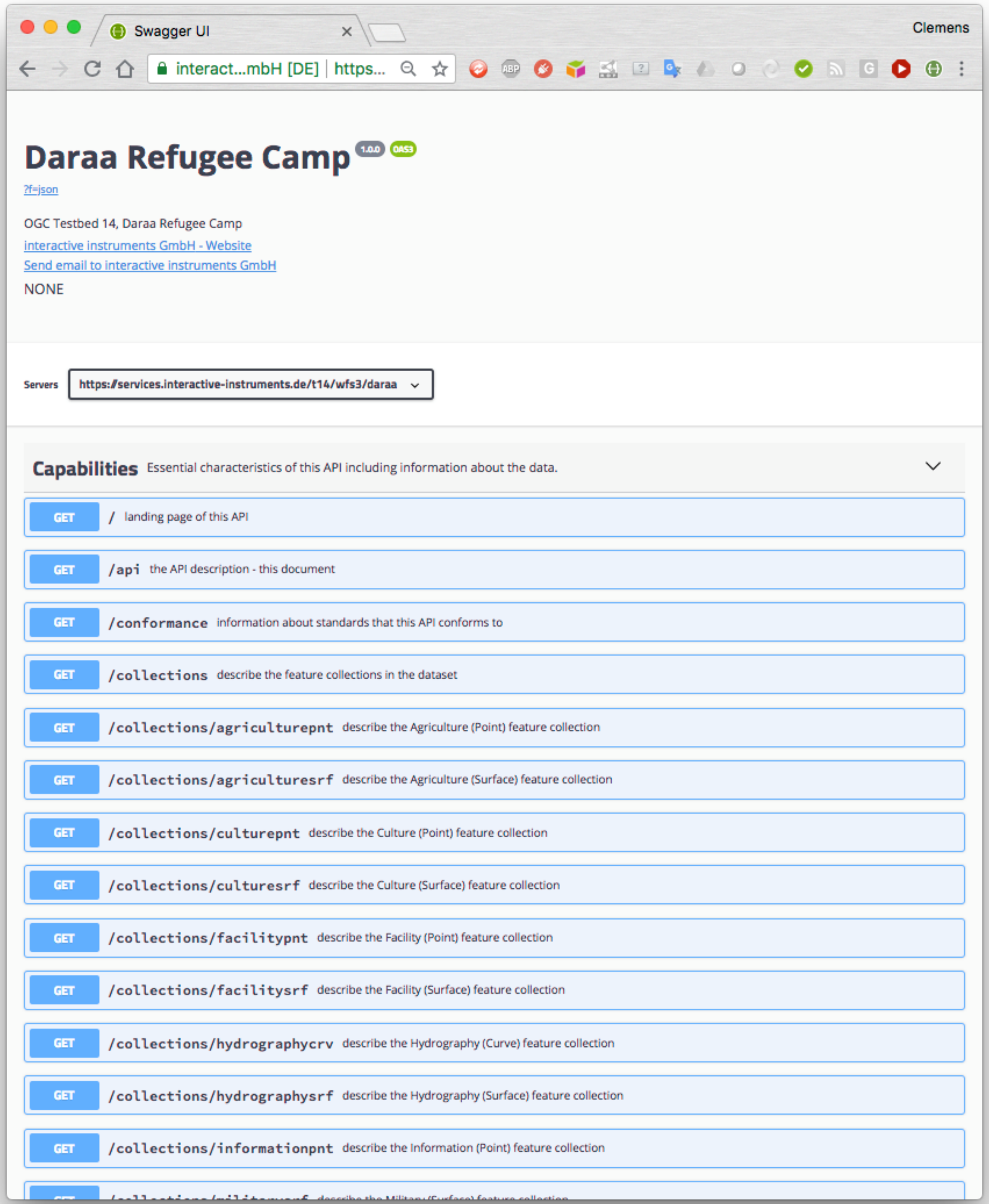
Only items are counted that are on the first level of the collection in the response document. Nested objects contained within the explicitly requested items shall not be counted.

10

bbox
array[number]
(query)
Only features that have a geometry that intersects the bounding box are selected. The bounding box is provided as four or six numbers, depending on whether the coordinate reference system includes a vertical axis (elevation or depth):

Figure 18. GeoSolutions GeoServer Swagger response

In the Interactive Instruments example below the resource path for the OpenAPI document is also provided to a browser-based Swagger application and the tool returns a similar easy to understand description of the API components.


Swagger UI

interact...mbH [DE] | https...

Clemens

Daraa Refugee Camp 1.0.0 OAS3

?f=json

OGC Testbed 14, Daraa Refugee Camp

interactive instruments GmbH - Website

Send email to interactive instruments GmbH

NONE

Servers https://services.interactive-instruments.de/t14/wfs3/daraa

Capabilities Essential characteristics of this API including information about the data.

GET / landing page of this API

GET /api the API description - this document

GET /conformance information about standards that this API conforms to

GET /collections describe the feature collections in the dataset

GET /collections/agriculturepnt describe the Agriculture (Point) feature collection

GET /collections/agriculturesrf describe the Agriculture (Surface) feature collection

GET /collections/culturepnt describe the Culture (Point) feature collection

GET /collections/culturesrf describe the Culture (Surface) feature collection

GET /collections/facilitypnt describe the Facility (Point) feature collection

GET /collections/facilitysrf describe the Facility (Surface) feature collection

GET /collections/hydrographycrv describe the Hydrography (Curve) feature collection

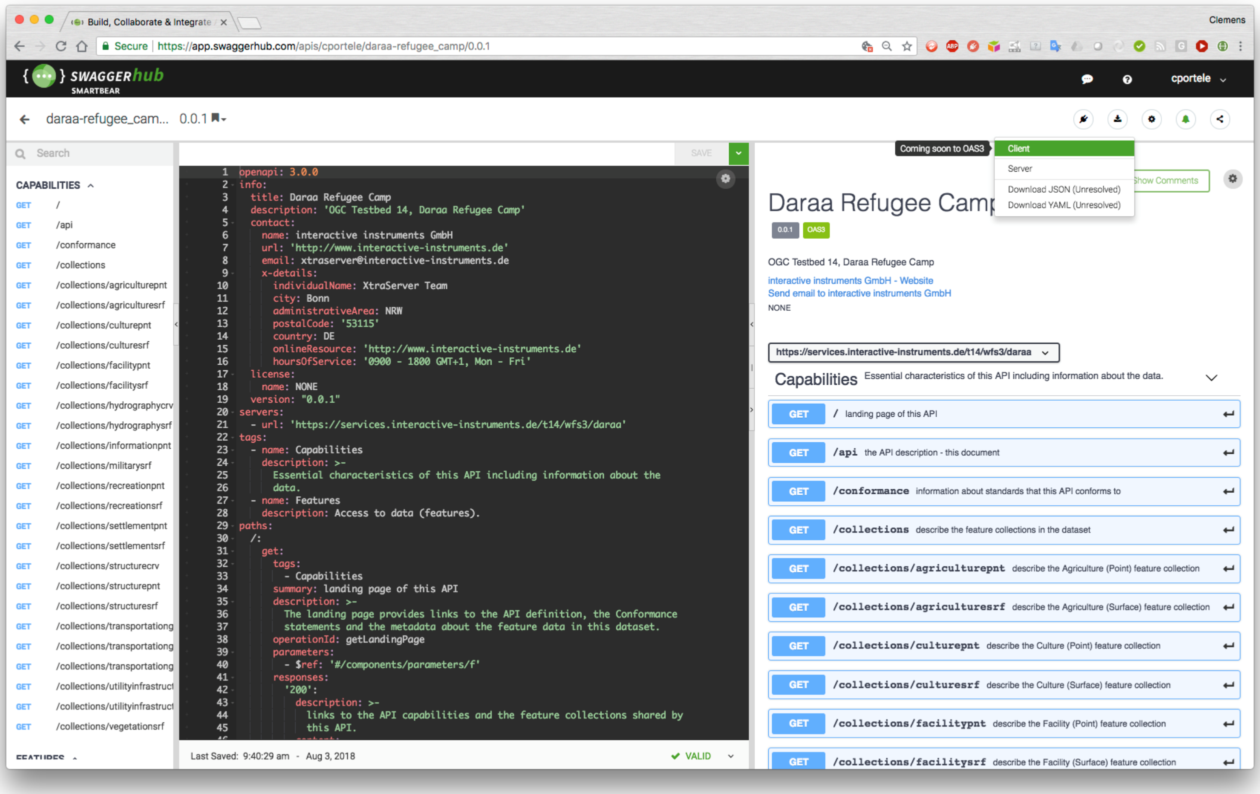
GET /collections/hydrographysrf describe the Hydrography (Surface) feature collection

GET /collections/informationpnt describe the Information (Point) feature collection

GET /collections/militarysrf describe the Military (Surface) feature collection

Figure 19. Interactive Instruments Swagger page

In the Interactive Instruments example below a different Swagger tool, Swagger Hub, is exercised. Specifically, the resource path for the OpenAPI document is provided to Swagger Hub. Swagger Hub is focused on helping streamline the design and development APIs based on OpenAPI.


Build, Collaborate & Integrate
Clemens
Secure | https://app.swaggerhub.com/apis/cportele/daraa-refugee_camp/0.0.1
SWAGGERhub
SMARTBEAR
cportele
daraa-refugee_cam... 0.0.1
Search
SAVE
Coming soon to OAS3
Client
Server
Download JSON (Unresolved)
Download YAML (Unresolved)
Show Comments
0.0.1
OAS3
FEATURES

CAPABILITIES

GET /
GET /api
GET /conformance
GET /collections
GET /collections/agriculturepnt
GET /collections/agriculturesrf
GET /collections/culturepnt
GET /collections/culturesrf
GET /collections/facilitypnt
GET /collections/facilitysrf
GET /collections/hydrographycrv
GET /collections/hydrographysrf
GET /collections/informationpnt
GET /collections/militarysrf
GET /collections/recreationpnt
GET /collections/recreationsrf
GET /collections/settlementpnt
GET /collections/settlementsrf
GET /collections/structurecrv
GET /collections/structurepnt
GET /collections/structuresrf
GET /collections/transportationg
GET /collections/transportationg
GET /collections/transportationg
GET /collections/utilityinfrastruct
GET /collections/utilityinfrastruct
GET /collections/vegetationsrf

```yaml
openapi: 3.0.0
info:
  title: Daraa Refugee Camp
  description: 'OGC Testbed 14, Daraa Refugee Camp'
  contact:
    name: interactive instruments GmbH
    url: 'http://www.interactive-instruments.de'
    email: xtraserver@interactive-instruments.de
    x-details:
      individualName: XtraServer Team
      city: Bonn
      administrativeArea: NRW
      postalCode: '53115'
      country: DE
      onlineResource: 'http://www.interactive-instruments.de'
      hoursOfService: '0900 - 1800 GMT+1, Mon - Fri'
  license:
    name: NONE
  version: "0.0.1"
servers:
  - url: 'https://services.interactive-instruments.de/t14/wfs3/daraa'
tags:
  - name: Capabilities
    description: >-
      Essential characteristics of this API including information about the
      data.
  - name: Features
    description: Access to data (features).
paths:
  /:
    get:
      tags:
        - Capabilities
      summary: landing page of this API
      description: >-
        The landing page provides links to the API definition, the Conformance
        statements and the metadata about the feature data in this dataset.
      operationId: getLandingPage
      parameters:
        - $ref: '#/components/parameters/f'
      responses:
        '200':
          description: >-
            links to the API capabilities and the feature collections shared by
            this API.
```

Last Saved: 9:40:29 am - Aug 3, 2018 ✓ VALID

Daraa Refugee Camp

OGC Testbed 14, Daraa Refugee Camp

interactive instruments GmbH - Website
Send email to interactive instruments GmbH
NONE

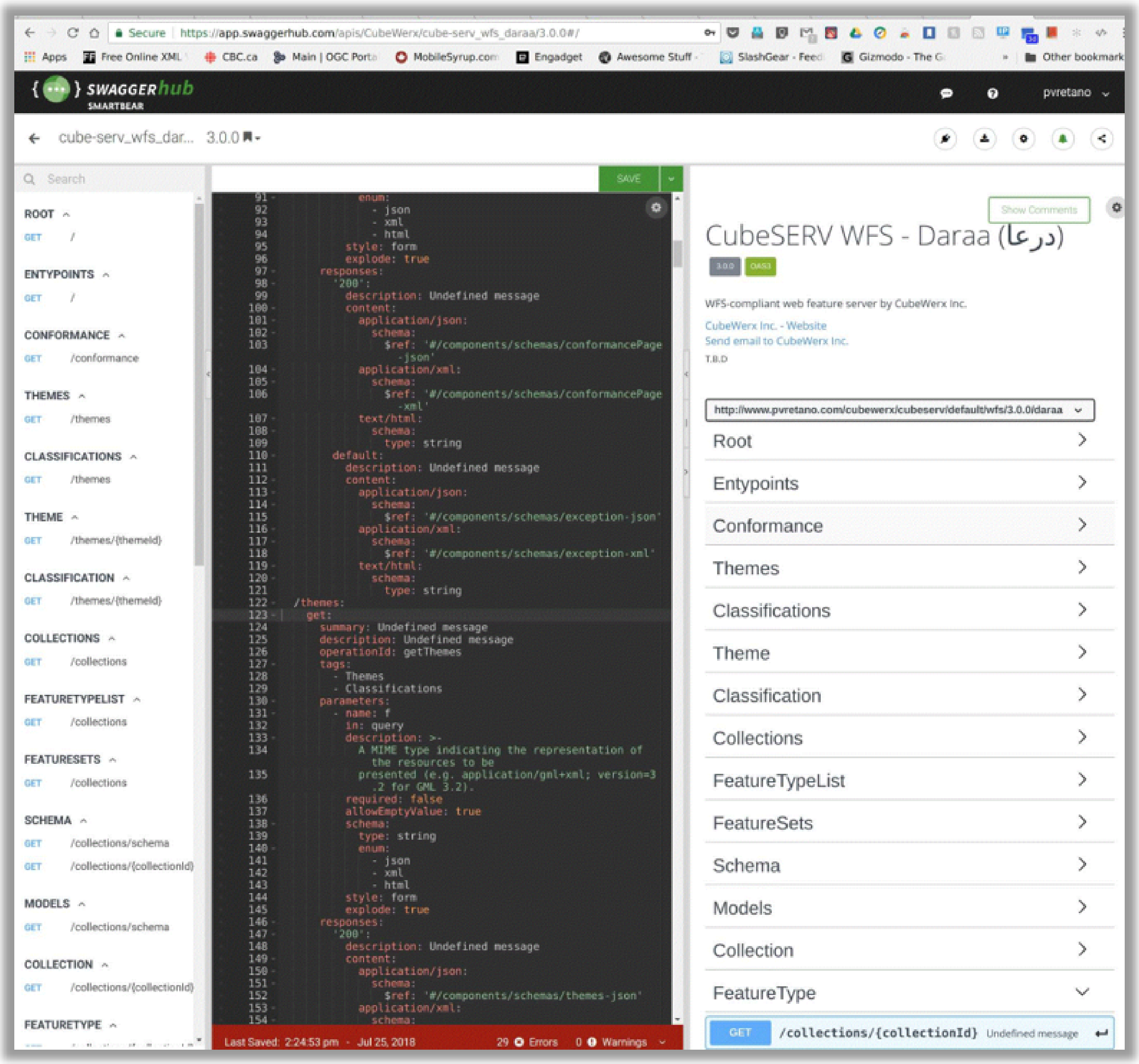
https://services.interactive-instruments.de/t14/wfs3/daraa

Capabilities Essential characteristics of this API including information about the data.

GET / landing page of this API
GET /api the API description - this document
GET /conformance information about standards that this API conforms to
GET /collections describe the feature collections in the dataset
GET /collections/agriculturepnt describe the Agriculture (Point) feature collection
GET /collections/agriculturesrf describe the Agriculture (Surface) feature collection
GET /collections/culturepnt describe the Culture (Point) feature collection
GET /collections/culturesrf describe the Culture (Surface) feature collection
GET /collections/facilitypnt describe the Facility (Point) feature collection
GET /collections/facilitysrf describe the Facility (Surface) feature collection

Figure 20. Interactive Instruments Swagger Hub page

In the Cubewerx example below Swagger Hub is shown at work during the API development process in July 2018, providing a rare glimpse at development in an OGC Testbed.



```yaml
            enum:
              - json
              - xml
              - html
          style: form
          explode: true
      responses:
        '200':
          description: Undefined message
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/conformancePage-json'
            application/xml:
              schema:
                $ref: '#/components/schemas/conformancePage-xml'
            text/html:
              schema:
                type: string
        default:
          description: Undefined message
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/exception-json'
            application/xml:
              schema:
                $ref: '#/components/schemas/exception-xml'
            text/html:
              schema:
                type: string
  /themes:
    get:
      summary: Undefined message
      description: Undefined message
      operationId: getThemes
      tags:
        - Themes
        - Classifications
      parameters:
        - name: f
          in: query
          description: >-
            A MIME type indicating the representation of the resources to be
            presented (e.g. application/gml+xml; version=3.2 for GML 3.2).
          required: false
          allowEmptyValue: true
          schema:
            type: string
            enum:
              - json
              - xml
              - html
          style: form
          explode: true
      responses:
        '200':
          description: Undefined message
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/themes-json'
            application/xml:
              schema:
```

Last Saved: 2:24:53 pm · Jul 25, 2018   29 Errors   0 Warnings

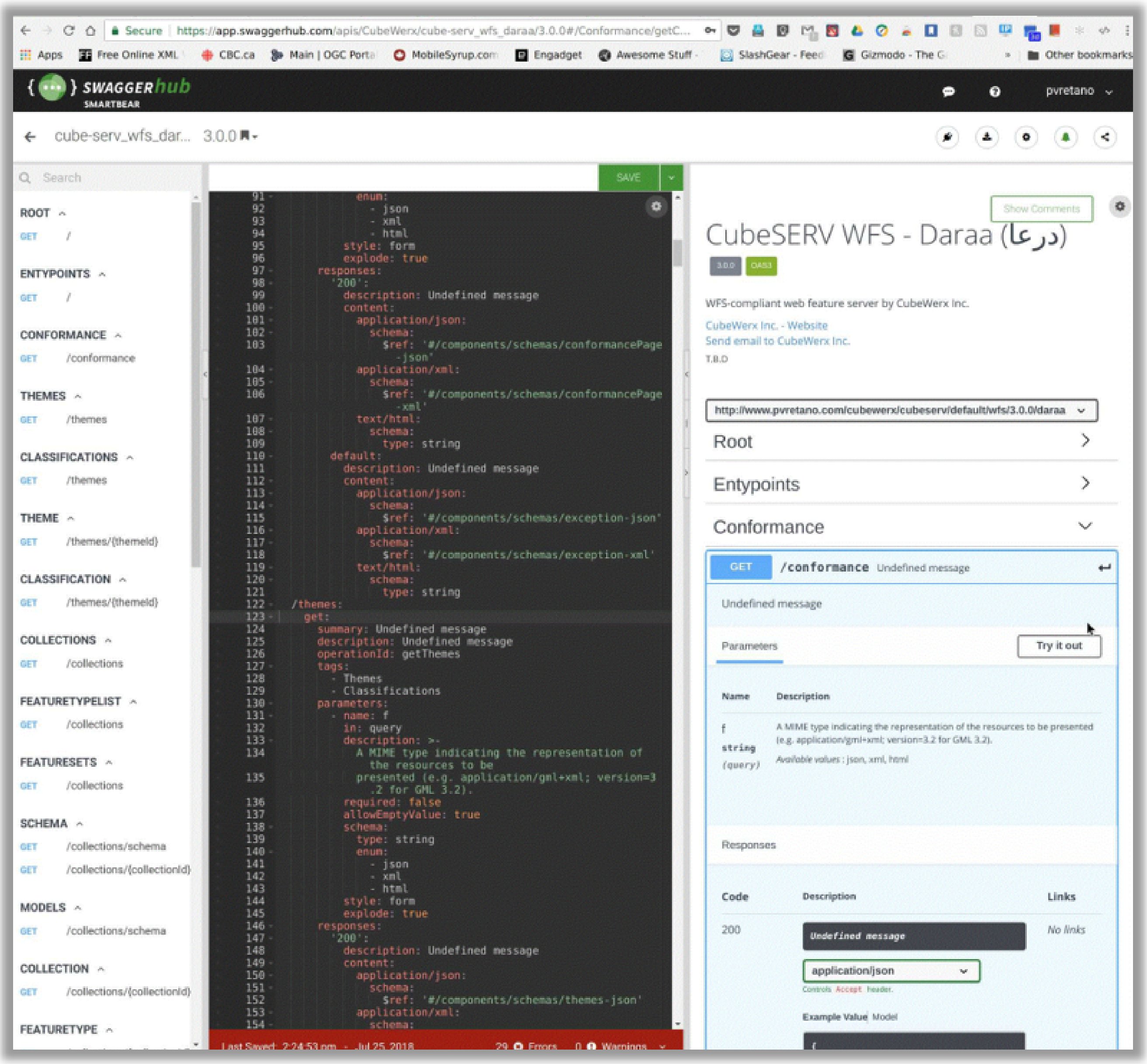
CubeSERV WFS - Daraa (درعا)

WFS-compliant web feature server by CubeWerx Inc.

CubeWerx Inc. - Website
Send email to CubeWerx Inc.
T.B.D

http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/daraa

Figure 21. Cubewerx Swagger page



Secure https://app.swaggerhub.com/apis/CubeWerx/cube-serv_wfs_daraa/3.0.0#/Conformance/getC...
SWAGGERhub SMARTBEAR
pvretano
cube-serv_wfs_dar... 3.0.0
Search
SAVE

ROOT
GET /
ENTYPOINTS
GET /
CONFORMANCE
GET /conformance
THEMES
GET /themes
CLASSIFICATIONS
GET /themes
THEME
GET /themes/{themeId}
CLASSIFICATION
GET /themes/{themeId}
COLLECTIONS
GET /collections
FEATURETYPELIST
GET /collections
FEATURESETS
GET /collections
SCHEMA
GET /collections/schema
GET /collections/{collectionId}
MODELS
GET /collections/schema
COLLECTION
GET /collections/{collectionId}
FEATURETYPE

```
            enum:
              - json
              - xml
              - html
          style: form
          explode: true
      responses:
        '200':
          description: Undefined message
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/conformancePage-json'
            application/xml:
              schema:
                $ref: '#/components/schemas/conformancePage-xml'
            text/html:
              schema:
                type: string
        default:
          description: Undefined message
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/exception-json'
            application/xml:
              schema:
                $ref: '#/components/schemas/exception-xml'
            text/html:
              schema:
                type: string
  /themes:
    get:
      summary: Undefined message
      description: Undefined message
      operationId: getThemes
      tags:
        - Themes
        - Classifications
      parameters:
        - name: f
          in: query
          description: >-
            A MIME type indicating the representation of the resources to be
            presented (e.g. application/gml+xml; version=3.2 for GML 3.2).
          required: false
          allowEmptyValue: true
          schema:
            type: string
            enum:
              - json
              - xml
              - html
          style: form
          explode: true
      responses:
        '200':
          description: Undefined message
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/themes-json'
            application/xml:
              schema:
```

Last Saved: 2:24:53 pm - Jul 25, 2018 29 Errors 0 Warnings

CubeSERV WFS - Daraa (درعا)

3.0.0 OAS3

WFS-compliant web feature server by CubeWerx Inc.

CubeWerx Inc. - Website
Send email to CubeWerx Inc.
T.B.D

http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/daraa

Root

Entypoints

Conformance

GET /conformance Undefined message

Undefined message

Parameters Try it out

Name	Description
f string (query)	A MIME type indicating the representation of the resources to be presented (e.g. application/gml+xml; version=3.2 for GML 3.2). Available values : json, xml, html

Responses

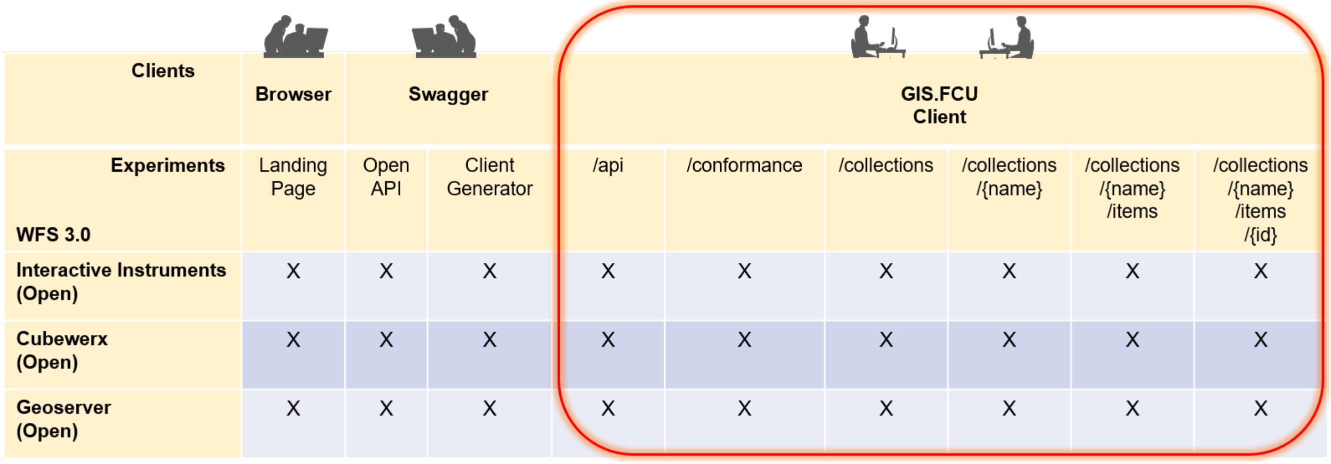
Code	Description	Links
200	Undefined message application/json Controls Accept header. Example Value Model	No links

Figure 22. Cubewerx Swagger page 2

Overall, Swagger tools were used extensively in the next generation APIs effort in Testbed-14, helping streamline design and development in all phases of the project. Although used for API development some participants exercised even more advanced functions of Swagger, including the code generation capability. Swagger Codegen can simplify development by generating server stubs and client SDKs APIs defined with the OpenAPI specification.

WFS 3.0 Client

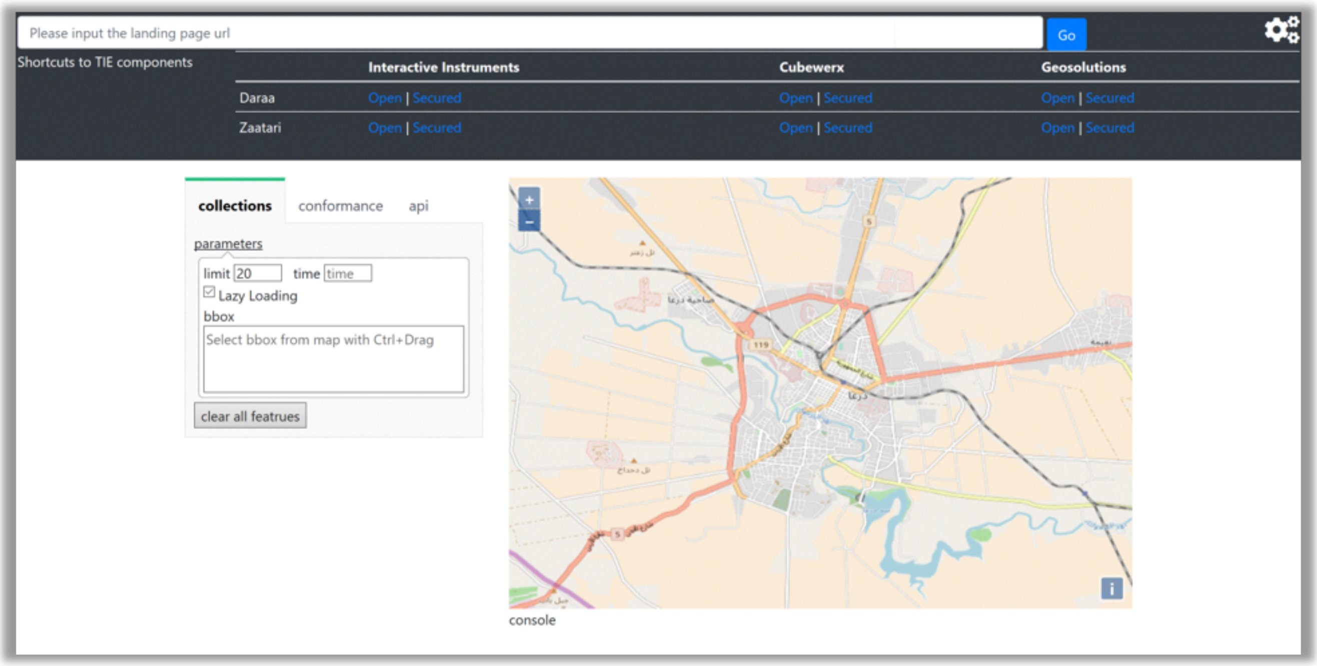
To test and demonstrate functionality a client implementation of WFS 3.0 Core was developed by GIS FCU. Testing focused on the API Definition (path /api), Conformance statements (path /conformance), and the Dataset Distribution (path /collections) resources as outlined in the following TIE table -



Clients	Browser	Swagger		GIS.FCU Client					
Experiments / WFS 3.0	Landing Page	Open API	Client Generator	/api	/conformance	/collections	/collections /{name}	/collections /{name} /items	/collections /{name} /items /{id}
Interactive Instruments (Open)	X	X	X	X	X	X	X	X	X
Cubewerx (Open)	X	X	X	X	X	X	X	X	X
Geoserver (Open)	X	X	X	X	X	X	X	X	X

Figure 23. Client Technology Integration Experiments with unprotected WFS 3.0 implementations

The GIS FCU example below shows the Testbed-14 WFS 3.0 client application. The application is divided into several main areas. Along the top of the application is a text box to enter the URL of the WFS 3.0 API landing page, and shortcuts to TIE components from Interactive Instruments, Cubewerx and GeoSolutions. Below the shortcuts are links to test the /api, /conformance and /collections paths of the Interactive Instruments, Cubewerx and GeoSolutions WFS 3.0, Finally, there is the map view which provides a background map from OpenStreetMap and displays of OpenStreetMap data converted into the NGA TDS model from WFS 3.0.



Please input the landing page url

Go

Shortcuts to TIE components

	Interactive Instruments	Cubewerx	Geosolutions
Daraa	Open | Secured	Open | Secured	Open | Secured
Zaatari	Open | Secured	Open | Secured	Open | Secured

collections conformance api

parameters

limit 20 time time

Lazy Loading

bbox

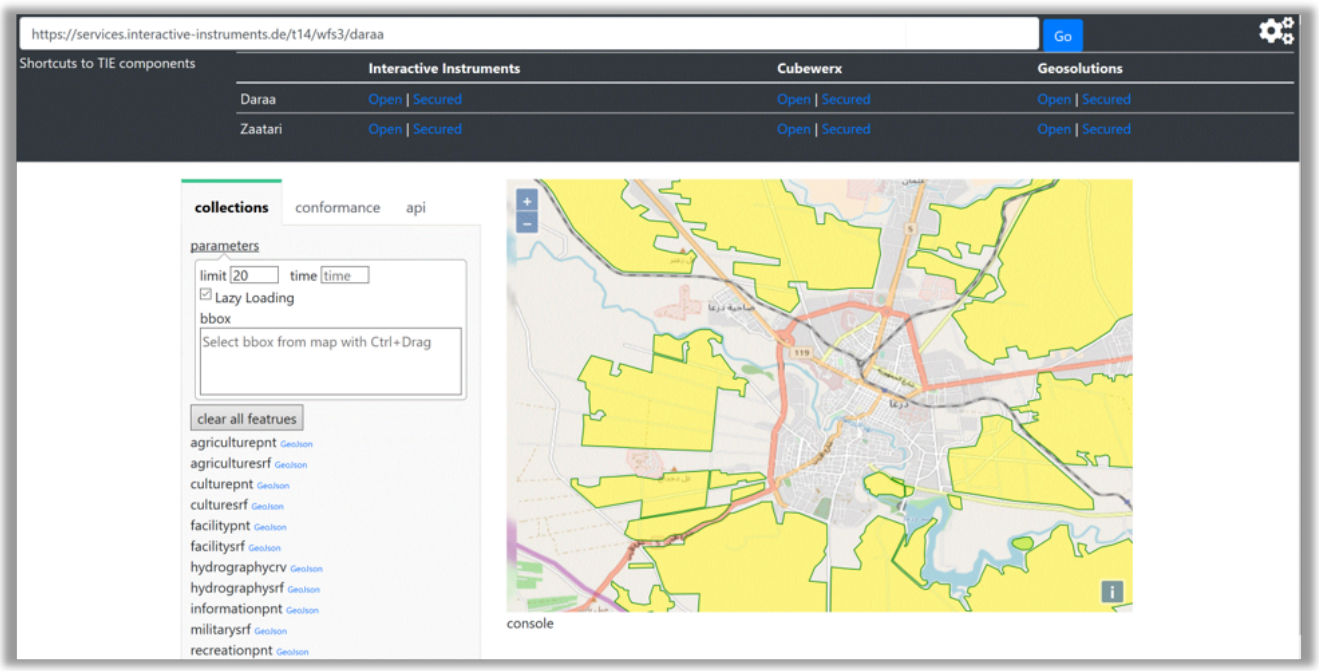
Select bbox from map with Ctrl+Drag

clear all featrues

console

Figure 24. GIS FCU Testbed-14 WFS 3.0 client application

In the example below the agriculture feature collection in GeoJSON from the Interactive Instruments collections API is accessed and displayed.


https://services.interactive-instruments.de/t14/wfs3/daraa

Go

Shortcuts to TIE components

	Interactive Instruments	Cubewerx	Geosolutions
Daraa	Open | Secured	Open | Secured	Open | Secured
Zaatari	Open | Secured	Open | Secured	Open | Secured

collections conformance api

parameters

limit 20 time time

☑ Lazy Loading

bbox

Select bbox from map with Ctrl+Drag

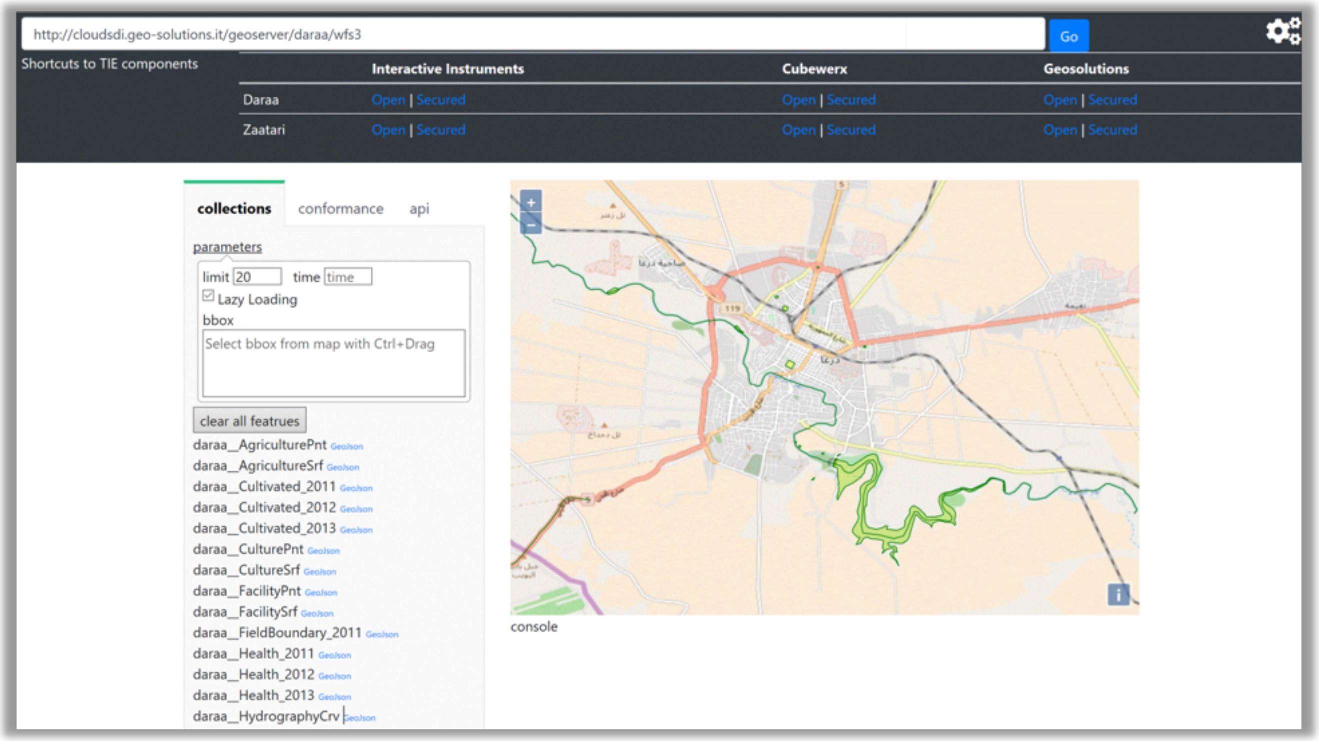
clear all featrues

- agriculturepnt GeoJson
- agriculturesrf GeoJson
- culturepnt GeoJson
- culturesrf GeoJson
- facilitypnt GeoJson
- facilitysrf GeoJson
- hydrographycrv GeoJson
- hydrographysrf GeoJson
- informationpnt GeoJson
- militarysrf GeoJson
- recreationpnt GeoJson

console

Figure 25. GIS FCU Testbed-14 WFS 3.0 client application accessing the Interactive Instruments collections API

In the example below the hydrography feature collection in GeoJSON from the GeoSolutions collections API is accessed and displayed.


http://cloudsdi.geo-solutions.it/geoserver/daraa/wfs3

Go

Shortcuts to TIE components

	Interactive Instruments	Cubewerx	Geosolutions
Daraa	Open | Secured	Open | Secured	Open | Secured
Zaatari	Open | Secured	Open | Secured	Open | Secured

collections conformance api

parameters

limit 20 time time

☑ Lazy Loading

bbox

Select bbox from map with Ctrl+Drag

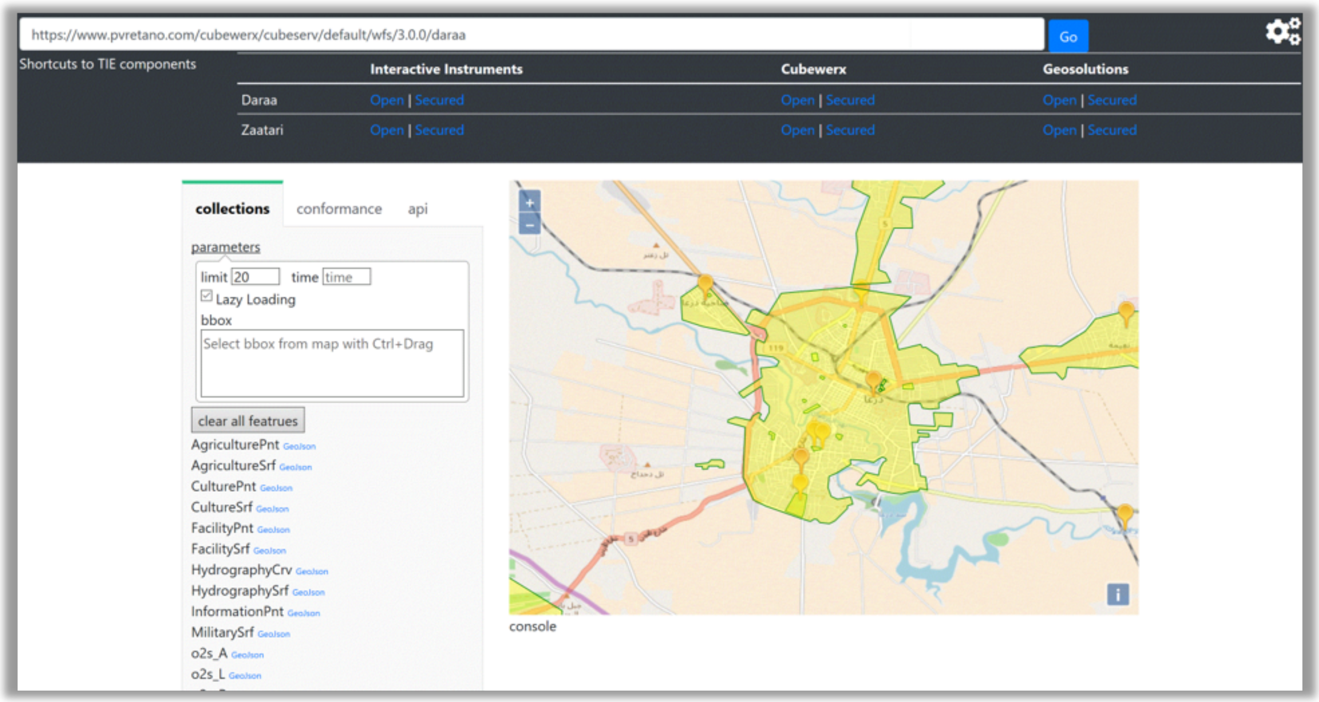
clear all featrues

- daraa_AgriculturePnt GeoJson
- daraa_AgricultureSrf GeoJson
- daraa_Cultivated_2011 GeoJson
- daraa_Cultivated_2012 GeoJson
- daraa_Cultivated_2013 GeoJson
- daraa_CulturePnt GeoJson
- daraa_CultureSrf GeoJson
- daraa_FacilityPnt GeoJson
- daraa_FacilitySrf GeoJson
- daraa_FieldBoundary_2011 GeoJson
- daraa_Health_2011 GeoJson
- daraa_Health_2012 GeoJson
- daraa_Health_2013 GeoJson
- daraa_HydrographyCrv GeoJson

console

Figure 26. GIS FCU Testbed-14 WFS 3.0 client application accessing the GeoSolutions collections API

In the example below the culture feature collection in GeoJSON from the Cubewerx collections API is accessed and displayed.


https://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/daraa

Shortcuts to TIE components

	Interactive Instruments	Cubewerx	Geosolutions
Daraa	Open | Secured	Open | Secured	Open | Secured
Zaatari	Open | Secured	Open | Secured	Open | Secured

Figure 27. GIS FCU Testbed-14 WFS 3.0 client application accessing the Cubewerx collections API

6.3.2. Secured WFS 3.0

OpenAPI 3.0 uses the term Security Scheme for authentication and authorization schemes. A Security Scheme Object provides a detailed description of a type of security control. Currently the OpenAPI specification supports the following Security Schemes:

- HTTP authentication,
- API key (either as a header or as a query parameter),
- OAuth2's common flows (implicit, password, application and access code) as defined in RFC6749, and
- OpenID Connect Discovery.

OAuth 2.0 and OpenID Connect were the security schemes implemented in the Testbed-14 WFS 3.0 effort. OpenID Connect is an authentication scheme on top of OAuth 2.0, an authorization scheme.

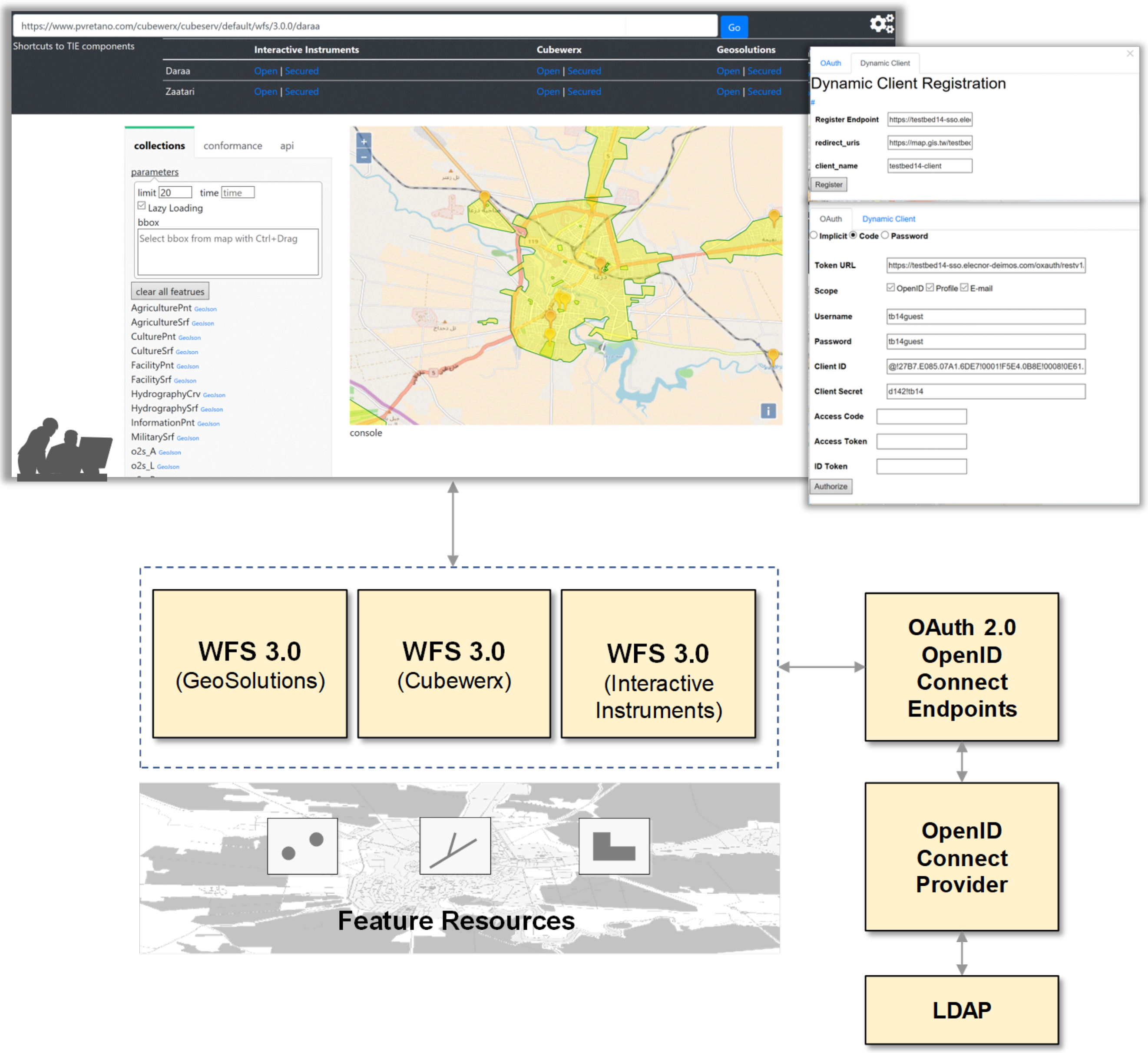
OAuth 2.0 provides a standard protocol for authorization. OAuth 2.0 supersedes the work done on the original OAuth protocol created in 2006. OAuth 2.0 focuses on client developer simplicity while providing specific authorization flows for web applications, desktop applications, mobile phones, and living room devices.

OpenID Connect (OIDC) provides a simple identity layer on top of the OAuth 2.0 protocol. It allows Clients to verify the identity of the End-User based on the authentication performed by an Authorization Server, as well as to obtain basic profile information about the End-User in an interoperable and REST-like manner. OpenID Connect allows clients of all types, including Web-based, mobile, and JavaScript clients, to request and receive information about authenticated sessions and end-users. The specification suite is extensible, allowing participants to use optional features such as encryption of identity data, discovery of OpenID Providers, and session management, when it makes sense for them.

For Testbed-14, participants implemented an Authorization Server, which was compliant with OAuth2.0 and OpenID Connect. The authorization server adds registration and management capabilities (that populate the LDAP service) on top of the authorization endpoints and makes information about the service available by means of an OpenID Connect Discovery document that lies on a commonly used relative path (/.well-known/openid-configuration).

The Basic Authentication Service is made available for potential use on profile management, and can either use LDAP credentials or utilize the authentication and authorization scheme. The end-users can then interact with clients to retrieve tokens and access services. Additional information on the Testbed-14 OpenID Connect and OAuth 2.0 security environment is provided in the Testbed-14 Security Engineering Report available at the following Security Engineering Report [https://github.com/opengeospatial/D024-Security_Engineering_Report/blob/master/6-auth.adoc].

The configuration of OAuth2.0 and OpenID Connect in the Next Generation APIs - WFS 3.0 component implementation design is shown below. The client application with security handling is provided by GIS.FCU. The WFS 3.0 are provided by Interactive Instruments, GeoSolutions and CubeWerx. The Authorization Server is provided by Deimos.


https://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/daraa

Shortcuts to TIE components

	Interactive Instruments	Cubewerx	Geosolutions
Daraa	Open | Secured	Open | Secured	Open | Secured
Zaatari	Open | Secured	Open | Secured	Open | Secured

limit 20 time time
☑ Lazy Loading
bbox
Select bbox from map with Ctrl+Drag

AgriculturePnt GeoJson
AgricultureSrf GeoJson
CulturePnt GeoJson
CultureSrf GeoJson
FacilityPnt GeoJson
FacilitySrf GeoJson
HydrographyCrv GeoJson
HydrographySrf GeoJson
InformationPnt GeoJson
MilitarySrf GeoJson
o2s_A GeoJson
o2s_L GeoJson

Dynamic Client Registration

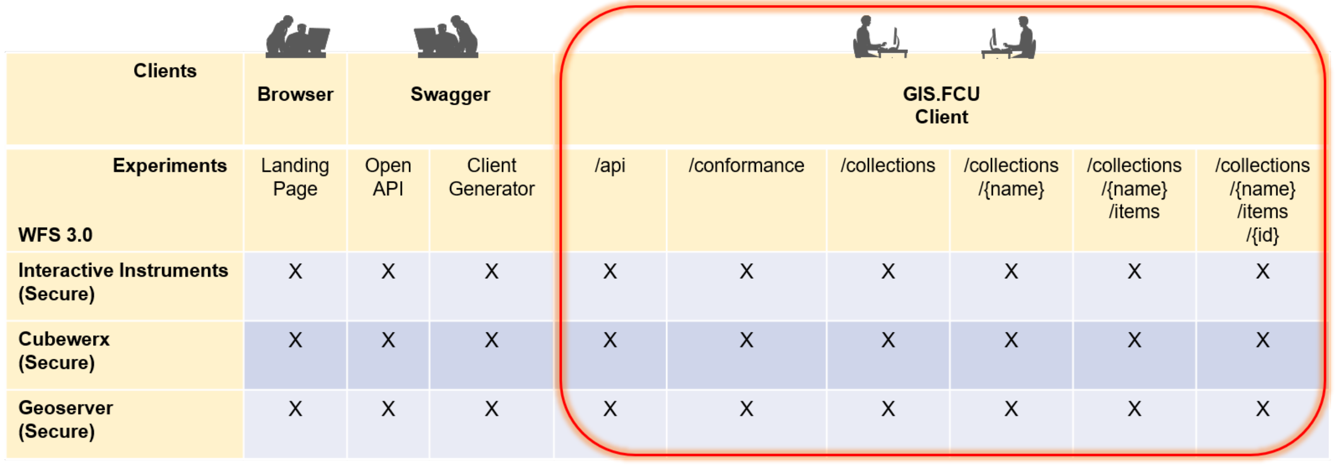
Register Endpoint	https://testbed14-sso.ele
redirect_uris	https://map.gis.tw/testbe
client_name	testbed14-client

Implicit [✓ Code] Password

Token URL	https://testbed14-sso.elecnor-deimos.com/oxauth/restv1.
Scope	☑ OpenID ☑ Profile ☑ E-mail
Username	tb14guest
Password	tb14guest
Client ID	@!27B7.E085.07A1.6DE7!0001!F5E4.0B8E!0008!0E61.
Client Secret	d142!tb14
Access Code	
Access Token	
ID Token	

Figure 28. Configuration of OAuth2.0 and OpenID Connect in the WFS 3.0 implementation design

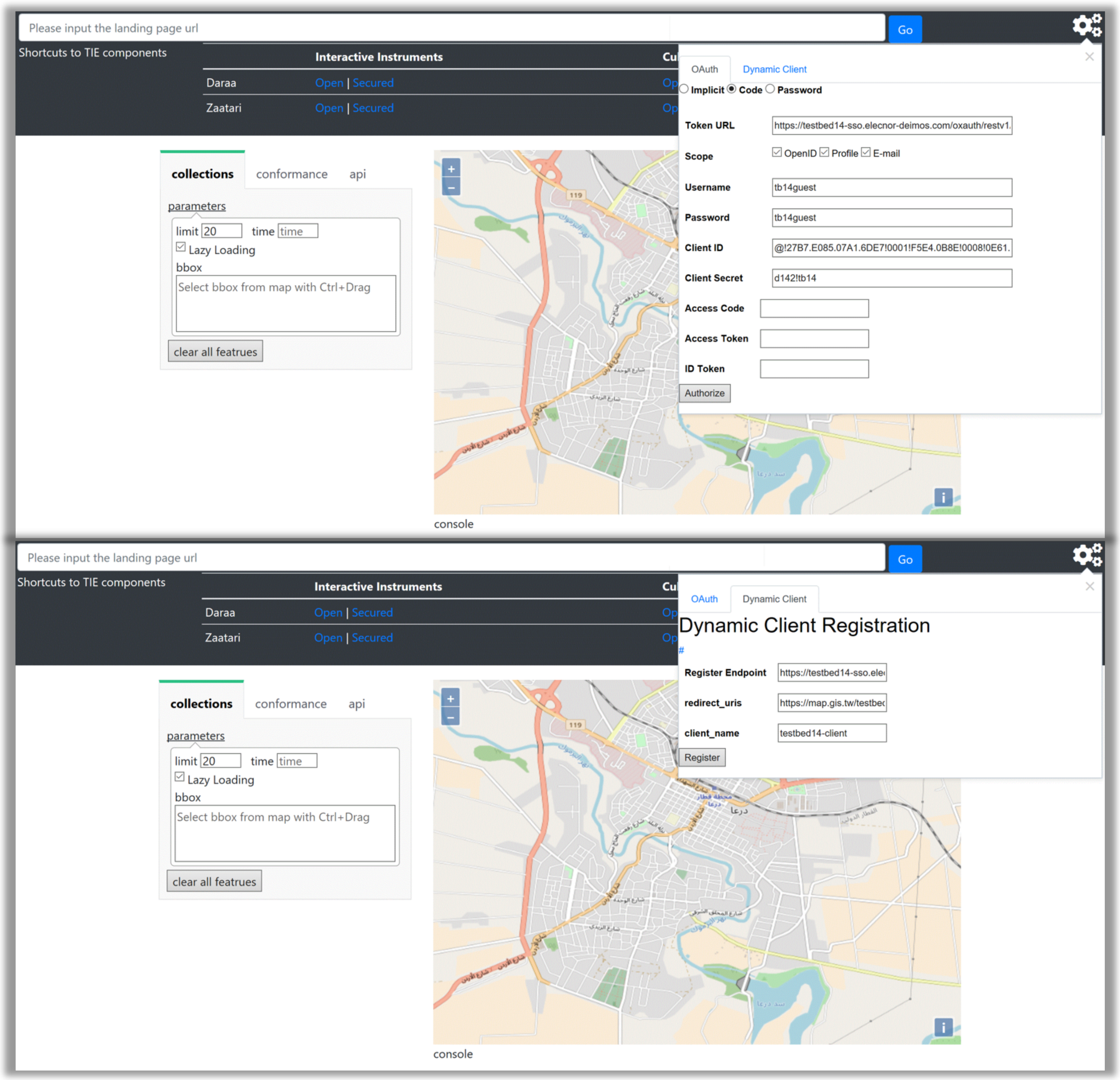
Testing of security for WFS 3.0 focused on access control for WFS 3.0 Core APIs including the API Definition (path /api), Conformance statements (path /conformance), and the Dataset Distribution (path /collections) resources as outlined in the following TIE table -



Clients	Browser	Swagger		GIS.FCU Client					
Experiments WFS 3.0	Landing Page	Open API	Client Generator	/api	/conformance	/collections	/collections /{name}	/collections /{name} /items	/collections /{name} /items /{id}
Interactive Instruments (Secure)	X	X	X	X	X	X	X	X	X
Cubewerx (Secure)	X	X	X	X	X	X	X	X	X
Geoserver (Secure)	X	X	X	X	X	X	X	X	X

Figure 29. Client Technology Integration Experiments with secure WFS 3.0 implementations

The GIS FCU example below shows the Testbed-14 client application. Along the top of the application is a text box to enter the URL of the WFS 3.0 API landing page, and shortcuts to WFS 3.0 TIE components from Interactive Instruments, Cubewerx and GeoSolutions. At the top right of the application are a set of panels to register the client on the AUthorization Server for proper permissions.



○ Implicit [✓ Code] ○ Password

Token URL	https://testbed14-sso.elecnor-deimos.com/oxauth/restv1.
Scope	☑ OpenID ☑ Profile ☑ E-mail
Username	tb14guest
Password	tb14guest
Client ID	@!27B7.E085.07A1.6DE7!0001!F5E4.0B8E!0008!0E61.
Client Secret	d142!tb14
Access Code	
Access Token	
ID Token	

Dynamic Client Registration

#

Register Endpoint	https://testbed14-sso.ele
redirect_uris	https://map.gis.tw/testbe
client_name	testbed14-client

Figure 30. Security panels of the GIS FCU Testbed-14 client application

In the client application, users can choose the following OAuth 2.0 permission flows -

- **Implicit Grant flow** - The Implicit Grant type is a simplified flow that can be used by public clients, where the access token is returned immediately without an extra authorization code exchange step [2]. It is generally not recommended to use the implicit flow, as industry best practice has changed to recommend that public clients should use the Authorization Code Grant flow without the client secret.
- **Authorization Code Grant flow** - The Authorization Code Grant flow is used by confidential and public clients to exchange an authorization code for an access token [3].
- **Password Grant flow** - The Password Grant flow can grant an access token based on Resource Owner Password Credentials (ROPC) grant type.
- **Dynamic Client Registration flow** - Users can also select Dynamic Client Registration of OAuth 2.0 if the Authorization Server supports it, and the client can skip this registration.

Additional information on these flows is provided in the Implementations section of this report.

Technology Integration Experiments

The TIEs for secured WFS 3.0 included three major component definitions - Service Type, Dataset, and Providers. Service Type defines what services are covered, such as WFS 3.0 with OAuth. Dataset defines what data source was been used, such as Daraa or Zaatari. Providers defines which Participant hosted the services. These three components are described in the following tables.

Service Types

Name	Abbre.	Description
WFS 3.0 with Security	WFS_Sec	Service Endpoint of WFS 3.0 with OAuth
OAuth Password Grant	OA_Pwd	The flow that grants an access token based on Resource Owner Password Credentials (ROPC) grant type
OAuth Implicit Grant	OA_Imp	The flow that is used by public clients where the access token is returned immediately without an extra authorization code exchange step.
OAuth Code Grant	OA_Code	The flow that is used by confidential and public clients to exchange an authorization code for an access token.

Dataset

Name	Abbre.	Description
Daraa	Da	NGA Topographic Data Store (TDS) OpenStreetMap data for Daraa area
Zaatari	Za	NGA Topographic Data Store (TDS) OpenStreetMap data for Zaatari area

Providers

Name	Abbre.	Description
Interactive Instruments	II	WFS 3.0 provider
CubeWerx	CW	WFS 3.0 provider
GeoSolutions	GS	WFS 3.0 provider

TIE Results

Results of TIE testing for OAuth are summarized in the following table, with an entry of "Yes" indicating a successful test

TIE for OAuth

Name	Success
OA_Pwd	Yes
OA_Imp	Yes
OA_Code	Yes

TIE for WFS

Each secured WFS 3.0 API endpoint is described using the abbreviation of the components in the following pattern

```
{Service Type}-{Dataset}-{Provider}
```

Results of TIE testing for secured WFS 3.0 API endpoints are summarized in the following table, with an entry of "Yes" indicating a successful test

Endpoint	landing page	/api	/conformance	/collections	/collections/{name}	/collections/{name}/items	/collections/{name}/items/{id}
WFS_Sec-Da-II	Yes	Yes	Yes	Yes	Yes	Yes	Yes
WFS_Sec-Za-II	Yes	Yes	Yes	Yes	Yes	Yes	Yes
WFS_Sec-Da-CW	Yes	Yes	Yes	Yes	Yes	Yes	Yes
WFS_Sec-Za-CW	Yes	Yes	Yes	Yes	Yes	Yes	Yes
WFS_Sec-Za-GS	Yes	Yes	Yes	Yes	Yes	Yes	Yes
WFS_Sec-Za-GS	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Additional information is provided in the Implementations section of this report.

Chapter 7. Implementations

The objective of the next generation APIs effort was to develop and test Web Feature Services (WFS) version 3.0. The initiative assessed OpenAPI, security based on OpenID Connect and OAuth 2.0 and WFS 3.0 extensions. The effort also began to assess methods to ease geospatial enterprise transition to next generation APIs.

This section describes the components implemented by Testbed-14 participants using WFS 3.0 with OpenAPI, and secured by OpenID Connect and OAuth 2.0.

7.1. Component Implementation Design

The WFS 3.0 Component Implementation Design for this part of Testbed-14 is shown below for reference -

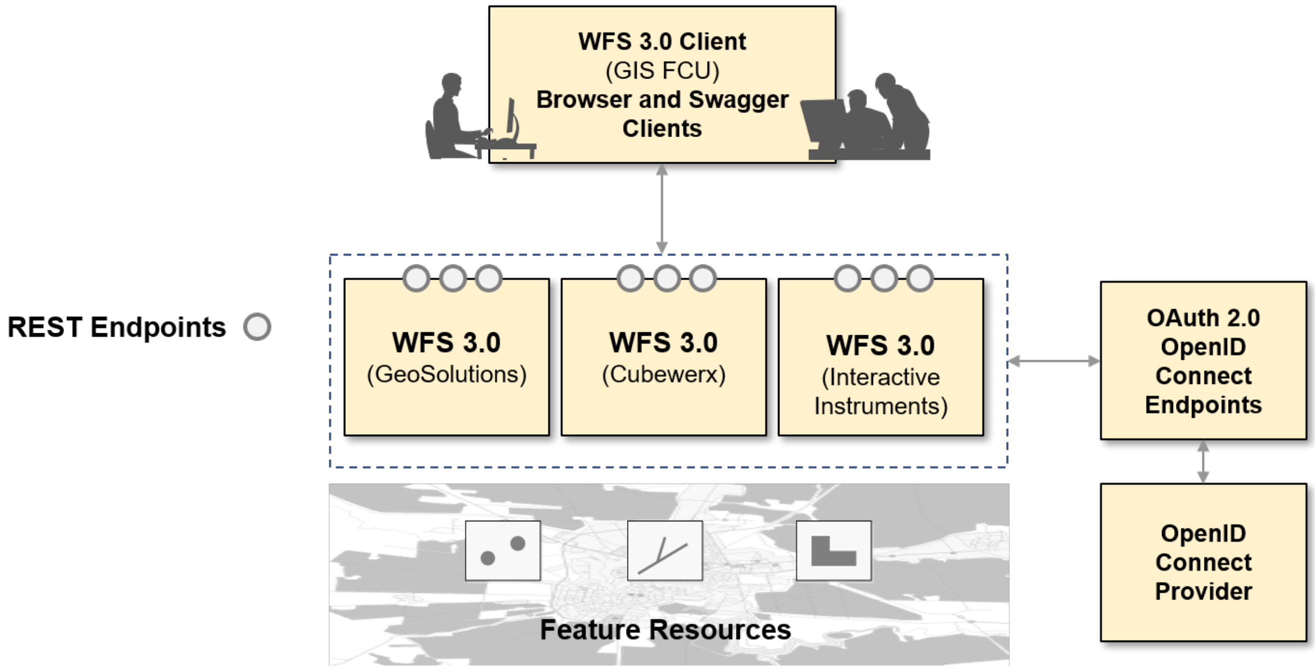


Figure 31. WFS 3.0 Component Implementation Design for this part of Testbed-14

7.2. Participant Implementations

This subsection describes the WFS 3.0 components implemented by Testbed-14 participants.

7.2.1. D113 - Next Generation API Implementation (GeoSolutions)

This deliverable consists of a single component, a WFS 3.0 server implementation developed by GeoSolutions [https://www.geo-solutions.it/] as a community plugin to the GeoServer [http://geoserver.org/] open source software [4]. The plugin has been made available to the GeoServer community from the start on the public GitHub repository [https://github.com/geoserver/geoserver/tree/master/src/community/wfs3], while its binary form can be found among the nightly builds of the development series [https://build.geoserver.org/geoserver/master/community-latest/geoserver-2.15-SNAPSHOT-wfs3-plugin.zip].

This component is published as four separate endpoints, exposing the OSM data for Daraa and

Zaatari areas, both as unsecured and secured services:

- Daraa Refugee Camp, unsecured [http://cloudsdi.geo-solutions.it/geoserver/daraa/wfs3]
- Daraa Refugee Camp, secured [https://cloudsdi.geo-solutions.it/geoserver/daraas/wfs3]
- Zaatari Refugee Camp, unsecured [http://cloudsdi.geo-solutions.it/geoserver/zaatari/wfs3]
- ZaatariRefugee Camp, secured [https://cloudsdi.geo-solutions.it/geoserver/zaataris/wfs3]

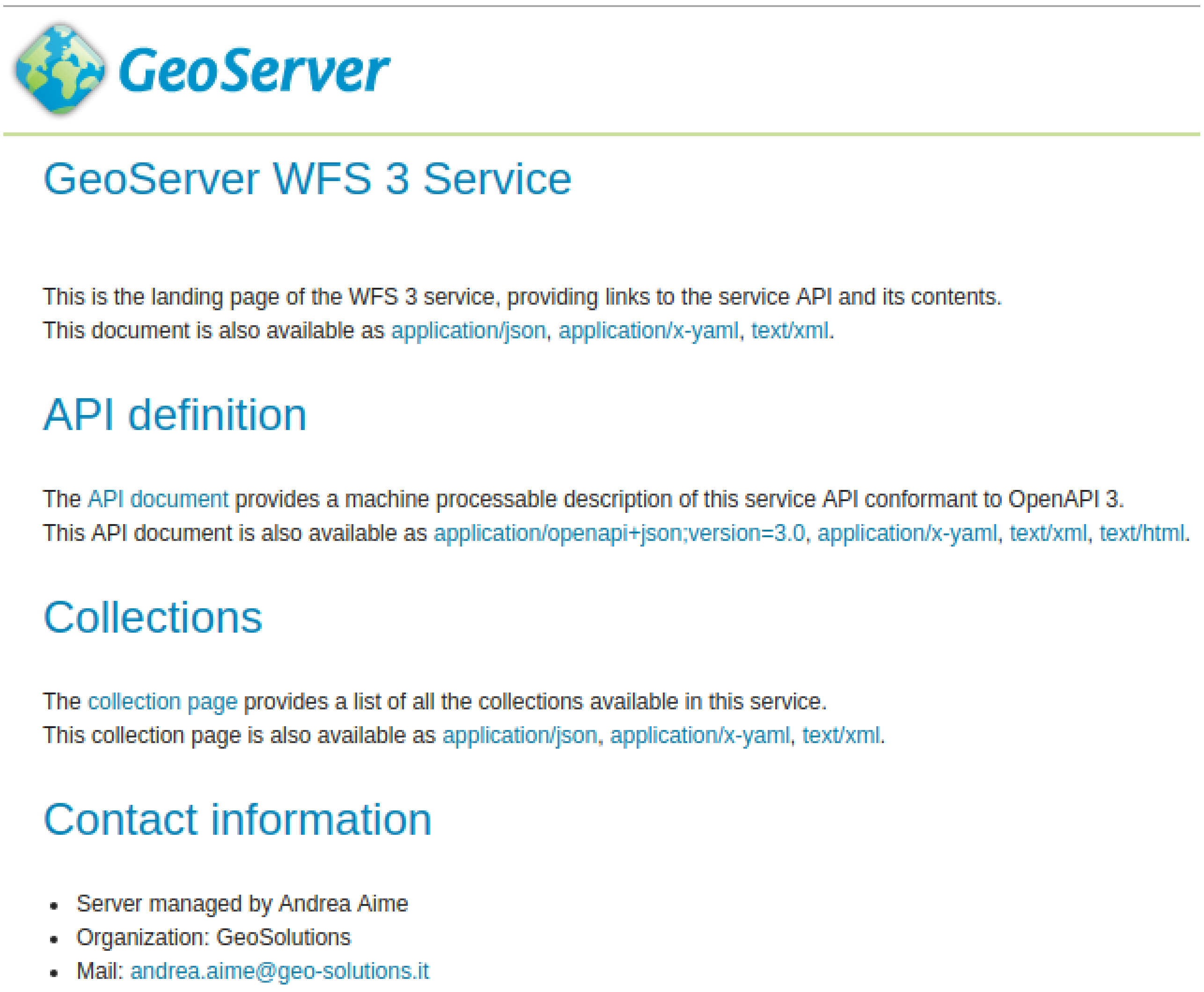
The module can be thought of as a protocol proxy layer running over the WFS 2.0 internal engine. It adds the following output formats to the WFS 3.0 implementation:

- RFC compliant GeoJSON outputs, with paging links (the current GeoJSON implementation, still available under a different media type, is the classic, pre-RFC GeoJSON output format)
- HTML output with paging links

Specific to WFS 3.0 are the implementations of the landing page, conformance and OpenAPI responses, as well as collection descriptions.

Exploring the HTML output is a good way to showcase the server abilities, especially since GeoServer's current HTML implementation is mimicking the other output formats (JSON, XML).

The landing page provides access to the API document and the collections, while including contact information should one need to get in touch with the server maintainers. When possible, links to alternate representations of the same resource are also provided.



GeoServer

GeoServer WFS 3 Service

This is the landing page of the WFS 3 service, providing links to the service API and its contents.
This document is also available as application/json, application/x-yaml, text/xml.

API definition

The API document provides a machine processable description of this service API conformant to OpenAPI 3.
This API document is also available as application/openapi+json;version=3.0, application/x-yaml, text/xml, text/html.

Collections

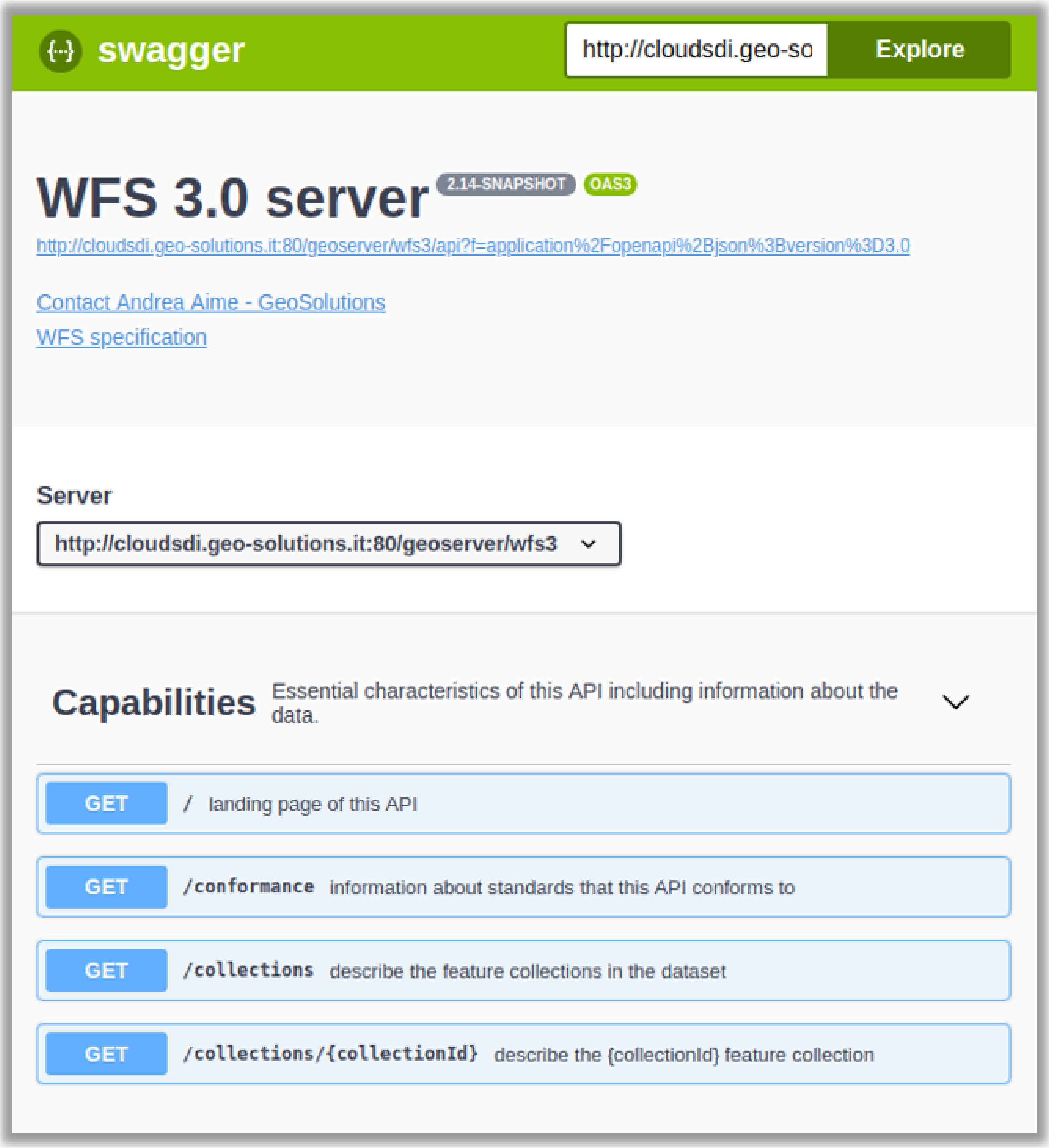
The collection page provides a list of all the collections available in this service.
This collection page is also available as application/json, application/x-yaml, text/xml.

Contact information

- Server managed by Andrea Aime
- Organization: GeoSolutions
- Mail: andrea.aime@geo-solutions.it

Figure 32. GeoServer landing page

The API page provides the familiar Swagger UI rendering. In particular, GeoServer generates a YAML document that the Swagger UI JavaScript toolkit turns into an API description, with the extra bonus of interactive request tests. An example is shown below:



WFS 3.0 server 2.14-SNAPSHOT OAS3

http://cloudsdi.geo-solutions.it:80/geoserver/wfs3/api?f=application%2Fopenapi%2Bjson%3Bversion%3D3.0

Contact Andrea Aime - GeoSolutions

WFS specification

Server

http://cloudsdi.geo-solutions.it:80/geoserver/wfs3

Capabilities Essential characteristics of this API including information about the data.

GET `/` landing page of this API

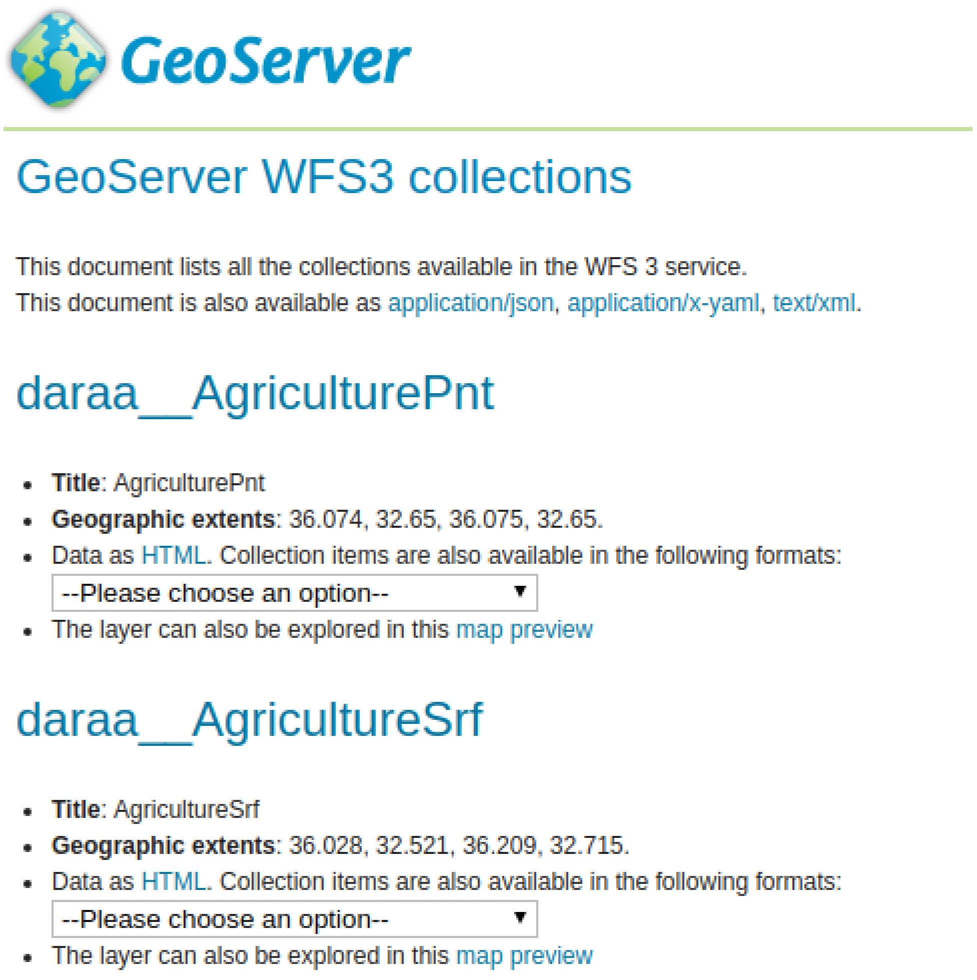
GET `/conformance` information about standards that this API conforms to

GET `/collections` describe the feature collections in the dataset

GET `/collections/{collectionId}` describe the {collectionId} feature collection

Figure 33. API page

The collections page lists the available collections, providing for each a direct HTML output on the "collections/{collectionId}/items" resource, as well as access to all available output formats in a dropdown.


GeoServer

GeoServer WFS3 collections

This document lists all the collections available in the WFS 3 service.
This document is also available as application/json, application/x-yaml, text/xml.

daraa__AgriculturePnt

- **Title**: AgriculturePnt
- **Geographic extents**: 36.074, 32.65, 36.075, 32.65.
- Data as HTML. Collection items are also available in the following formats:
  --Please choose an option--
- The layer can also be explored in this map preview

daraa__AgricultureSrf

- **Title**: AgricultureSrf
- **Geographic extents**: 36.028, 32.521, 36.209, 32.715.
- Data as HTML. Collection items are also available in the following formats:
  --Please choose an option--
- The layer can also be explored in this map preview

Figure 34. Collections page

As a result of the current architecture, GeoServer WFS 3.0 shares all the available output formats with the other versions of WFS, thus providing not only HTML, GeoJSON and Geography Markup Language (GML), but also KML, CSV, zipped shapefile, as well as others depending on what plugins have been installed (e.g. GeoPackage, DXF, Excel and OpenDocument spreadsheets, MIF, and more).

The collections page also links to a "map preview", which is the standard GeoServer preview, implemented as a GetMap with the special "application/openlayers" output format. An example is shown below:

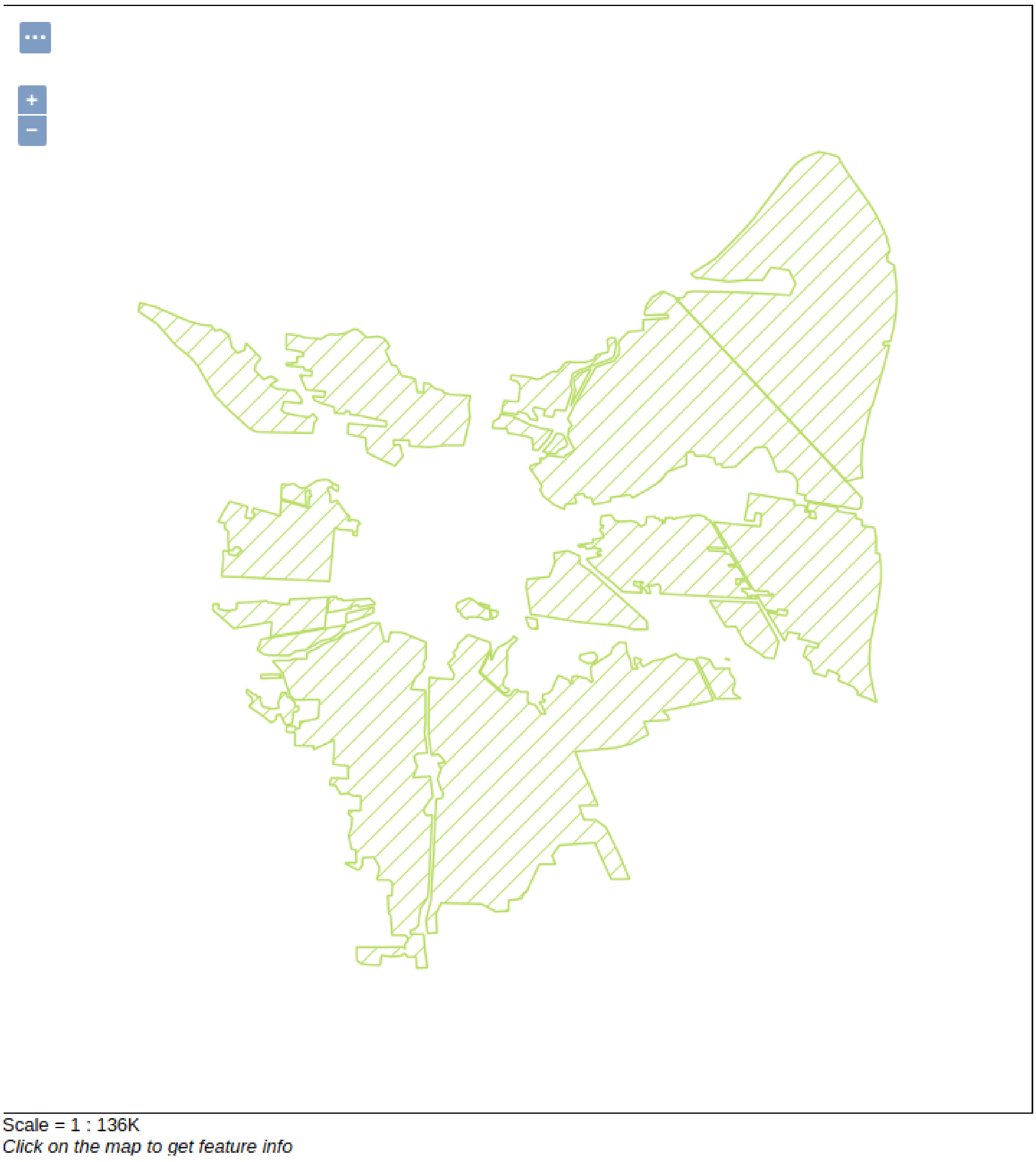
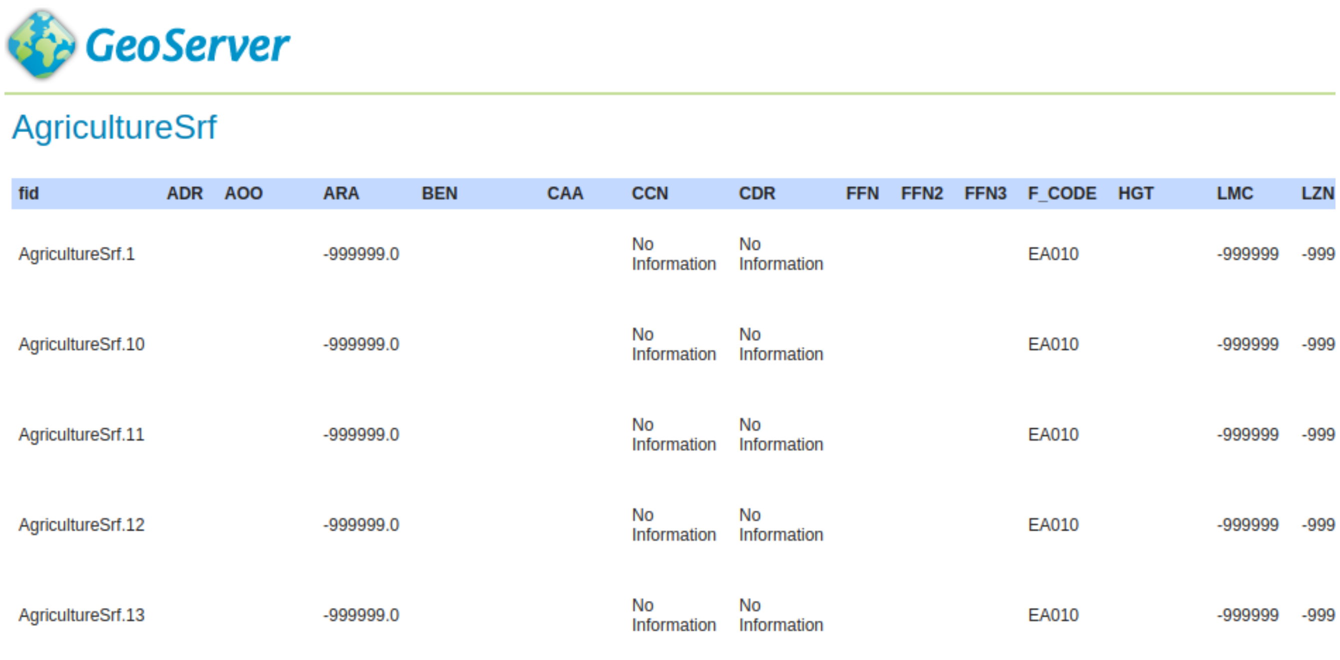


Figure 35. Map preview page

Finally, following the HTML output links to HTML output representation of the collection items:


GeoServer

AgricultureSrf

fid	ADR	AOO	ARA	BEN	CAA	CCN	CDR	FFN	FFN2	FFN3	F_CODE	HGT	LMC	LZN
AgricultureSrf.1			-999999.0			No Information	No Information				EA010		-999999	-999
AgricultureSrf.10			-999999.0			No Information	No Information				EA010		-999999	-999
AgricultureSrf.11			-999999.0			No Information	No Information				EA010		-999999	-999
AgricultureSrf.12			-999999.0			No Information	No Information				EA010		-999999	-999
AgricultureSrf.13			-999999.0			No Information	No Information				EA010		-999999	-999

Figure 36. Collection items

The current output is a simple table, with paging links at the bottom (omitted in the screenshot).

It is also worth noting that all HTML outputs are default implementations which can be replaced by the administrator on a workspace or collection basis with their own, using custom representations and their choice of page style and logos, thanks to Freemarker templating language [https://freemarker.apache.org/].

Conformance Classes

The current implementation of the WFS 3.0 GeoServer plugin supports the following conformance classes:

- Core
- OpenAPI 3.0
- HTML
- GeoJSON
- GML Simple Feature Profile, Level 2

The supported conformance classes are available at the path /conformance of each server.

Extensions

The WFS 3.0 protocol has presently no official extensions (CRS is currently in the works), GeoServer will likely start implementing extensions as they reach maturity.

However, given the nature of the internal proxy to the WFS 2.0 implementation, most WFS 2.0 KVP parameter can be seamlessly used in the WFS 3.0 protocol as well. For example:

- Random paging via "offset/limit", e.g. http://cloudsdi.geo-solutions.it/geoserver/daraa/wfs3/collections/daraa__Cultivated_2012/items?f=application%2Fgeo%2Bjson&limit=5&offset=10

- Filter via CQL_FILTER, e.g. http://cloudsdi.geo-solutions.it/geoserver/daraa/wfs3/collections/daraa__Cultivated_2012/items?f=application%2Fgeo%2Bjson&limit=5&offset=10&cql_filter=Area_ha%3E10
- Property selection via "propertyname", e.g. http://cloudsdi.geo-solutions.it/geoserver/daraa/wfs3/collections/daraa__Cultivated_2012/items?f=application%2Fgeo%2Bjson&limit=5&offset=10&propertyName=ClassName,Orchard

Security considerations

Before Testbed-14, GeoServer did support a few OAuth2 based variants, such as Google, GitHub and GeoNode ones, but not OpenID Connect. To integrate the server into the security environment of the Testbed, a new OAuth2/OpenId Connect plugin was developed and made available to the user community:

- Source code [https://github.com/geoserver/geoserver/tree/master/src/community/security/oauth2-openid-connect]
- Binary package [https://build.geoserver.org/geoserver/master/community-latest/geoserver-2.15-SNAPSHOT-sec-oauth2-openid-connect-plugin.zip]

The server endpoints provided for testing support both HTTP and HTTPS (the HTTPS one is preferred for authenticated requests).

IANA considerations

The WFS 3.0 implementation only added support for media types included in the specification itself. Non-IANA compliant media types might be present in the components as a result of previous development efforts (e.g., plugins, older versions of WFS also present in the server).

7.2.2. D140 - Next Generation API Implementation (interactive instruments)

The deliverable consists of two components, a WFS 2.0 component and a WFS 3.0 component that is a facade providing access to the WFS 2.0 services.

The data consists of two datasets used previously in OGC Testbed-13: Open Street Map data for two refugee camps that has been transformed to Shapefiles using feature and attribute codes from the NSG Application Schema (NAS) of NGA.

The WFS 2.0 server uses the XtraServer product of interactive instruments, an OGC Reference Implementation for WFS 2.0.

The endpoints are:

- Daraa Refugee Camp [https://services.interactive-instruments.de/t14/wfs2/daraa?SERVICE=WFS&REQUEST=GetCapabilities]
- Zaatari Refugee Camp [https://services.interactive-instruments.de/t14/wfs2/zaatari?SERVICE=WFS&REQUEST=GetCapabilities]

As no official schema for the datasets exist, the GML application schema is derived from the datasets.

The WFS 3.0 server uses the ldproxy software, developed by interactive instruments.

The ldproxy tool has been developed as a proxy for WFS endpoints to explore how WFS could be better aligned with today’s Web and the expectations of developers and users. See, for example, the report titled "Spatial Data on the Web using the current SDI" [5]. The results have been input to the development of the W3C/OGC Spatial Data on the Web Best Practice [6] and the OGC Change Request for WFS 3.0 [7].

From the beginning of the WFS 3.0 process, ldproxy has been updated to assess the usability of design proposals for the API building blocks of WFS 3.0.

The WFS 3.0 landing pages are:

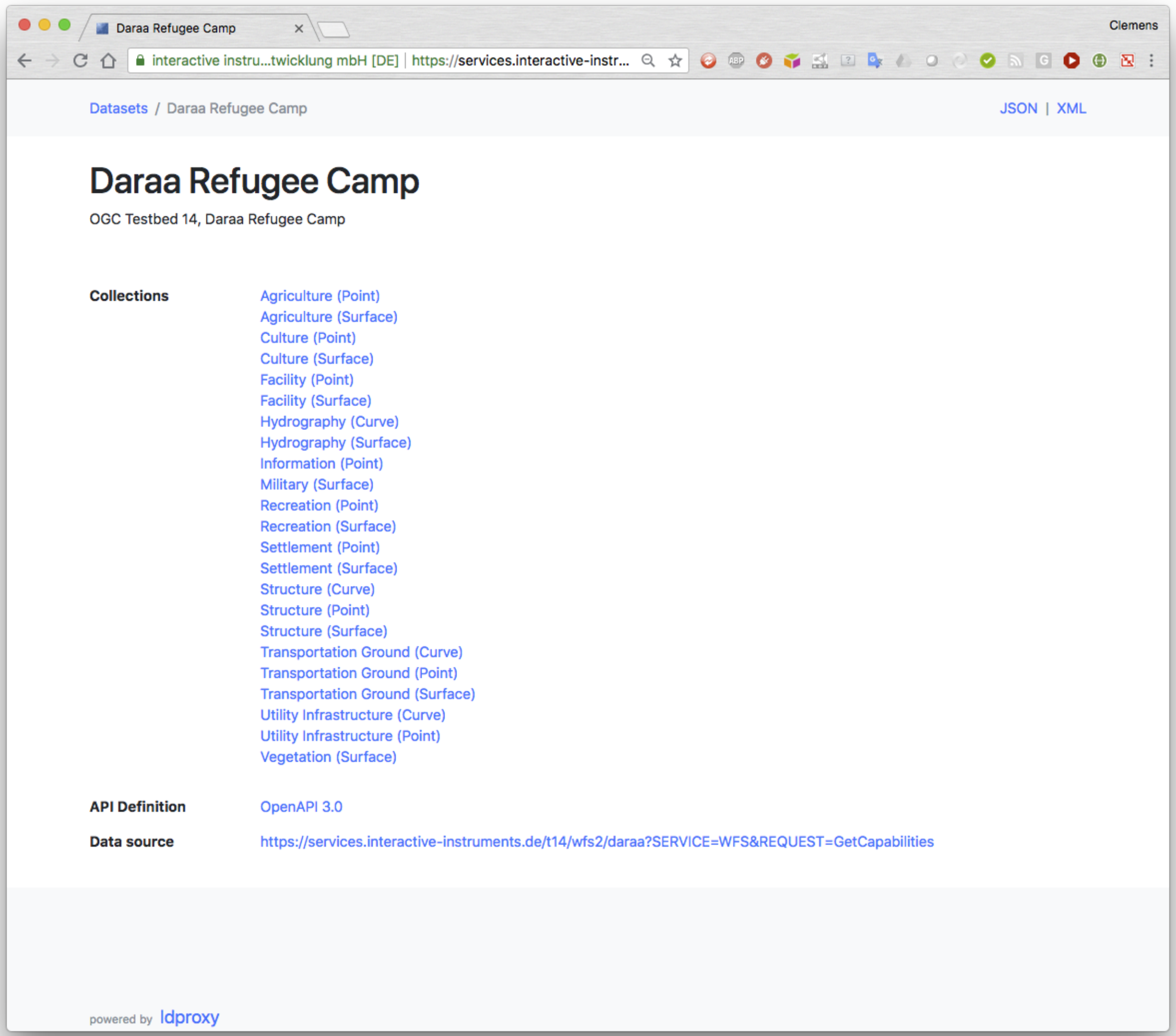
- Daraa Refugee Camp [https://services.interactive-instruments.de/t14/wfs3/daraa]
- Zaatari Refugee Camp [https://services.interactive-instruments.de/t14/wfs3/zaatari]

In addition, secured services have been set up using the OpenID Connect/OAuth2.0 Server deployed by Deimos Space for the Testbed.

- Daraa dataset:
    - Landing page: https://services.interactive-instruments.de/t14/wfs3/daraa-sec/ (requires token to access)
    - OpenAPI definition: https://services.interactive-instruments.de/t14/wfs3/daraa-sec/api/
- Zaatari dataset:
    - Landing page: https://services.interactive-instruments.de/t14/wfs3/zaatari-sec/ (requires token to access)
    - OpenAPI definition: https://services.interactive-instruments.de/t14/wfs3/zaatari-sec/api/

The configuration of the facade services for the Daraa dataset has been tweaked to improve the usability of the services. For example, feature and attribute names have be changed so that they can be understood more easily (in HTML and GeoJSON), and attributes useful for filtering have been configured.

The following screenshots illustrate the current HTML representation of selected resources.


Daara Refugee Camp
Clemens
interactive instru...twicklung mbH [DE] | https://services.interactive-instr...
Datasets / Daraa Refugee Camp
JSON | XML

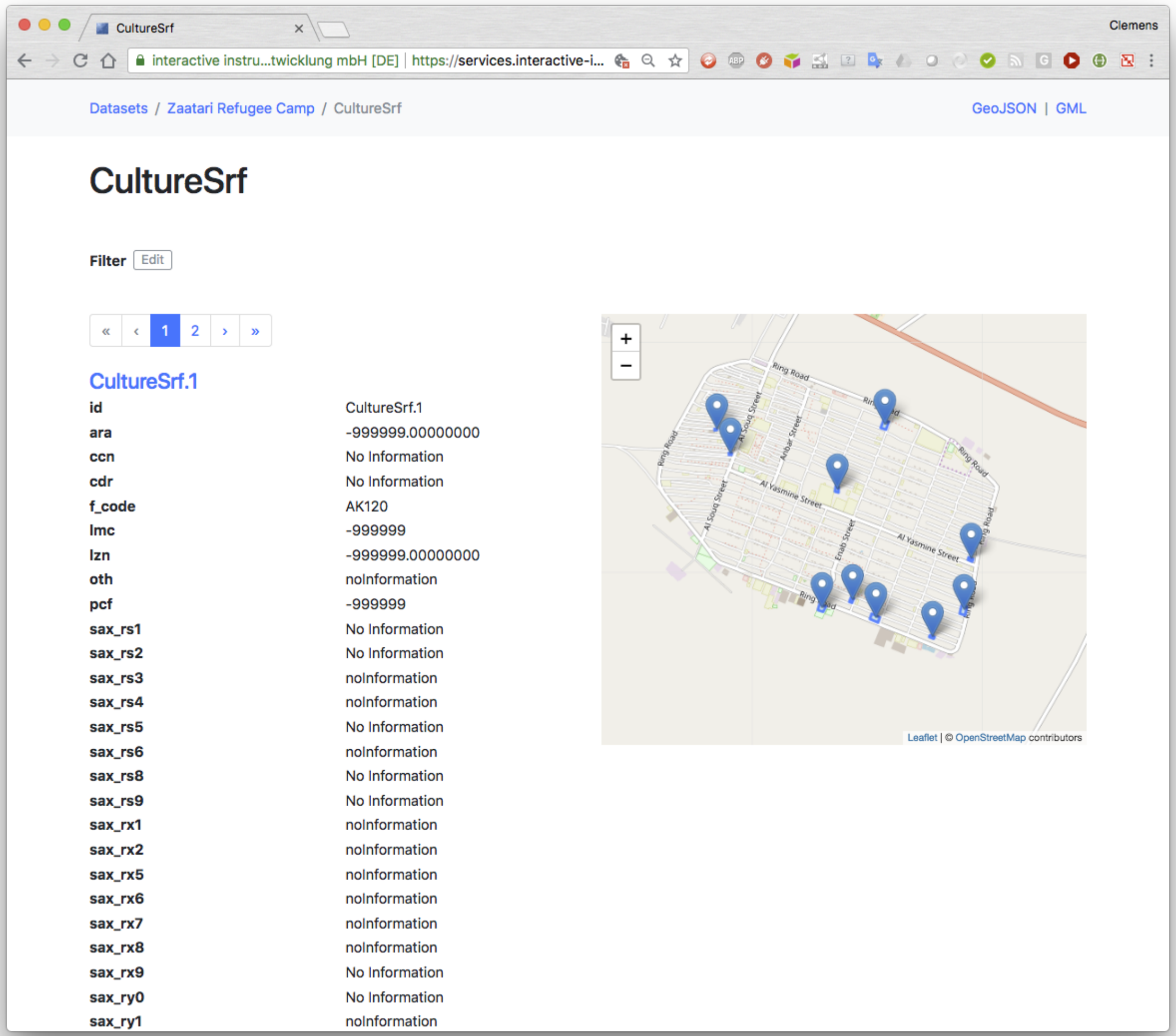
Daraa Refugee Camp

OGC Testbed 14, Daraa Refugee Camp

Collections	Agriculture (Point) Agriculture (Surface) Culture (Point) Culture (Surface) Facility (Point) Facility (Surface) Hydrography (Curve) Hydrography (Surface) Information (Point) Military (Surface) Recreation (Point) Recreation (Surface) Settlement (Point) Settlement (Surface) Structure (Curve) Structure (Point) Structure (Surface) Transportation Ground (Curve) Transportation Ground (Point) Transportation Ground (Surface) Utility Infrastructure (Curve) Utility Infrastructure (Point) Vegetation (Surface)
API Definition	OpenAPI 3.0
Data source	https://services.interactive-instruments.de/t14/wfs2/daraa?SERVICE=WFS&REQUEST=GetCapabilities

Figure 37. ldproxy landing page in HTML

Since ldproxy is a WFS 2.0 facade, it includes not only the standard WFS 3.0 links to the API description and to each collection, it also includes a link to the WFS 2.0 server.


CultureSrf
Clemens
interactive instru...twicklung mbH [DE] | https://services.interactive-i...
Datasets / Zaatari Refugee Camp / CultureSrf
GeoJSON | GML
Filter
Edit
« ‹ 1 2 › »
Ring Road
Al Soud Street
Anbar Street
Al Yasmine Street
Enab Street
Leaflet | © OpenStreetMap contributors

CultureSrf

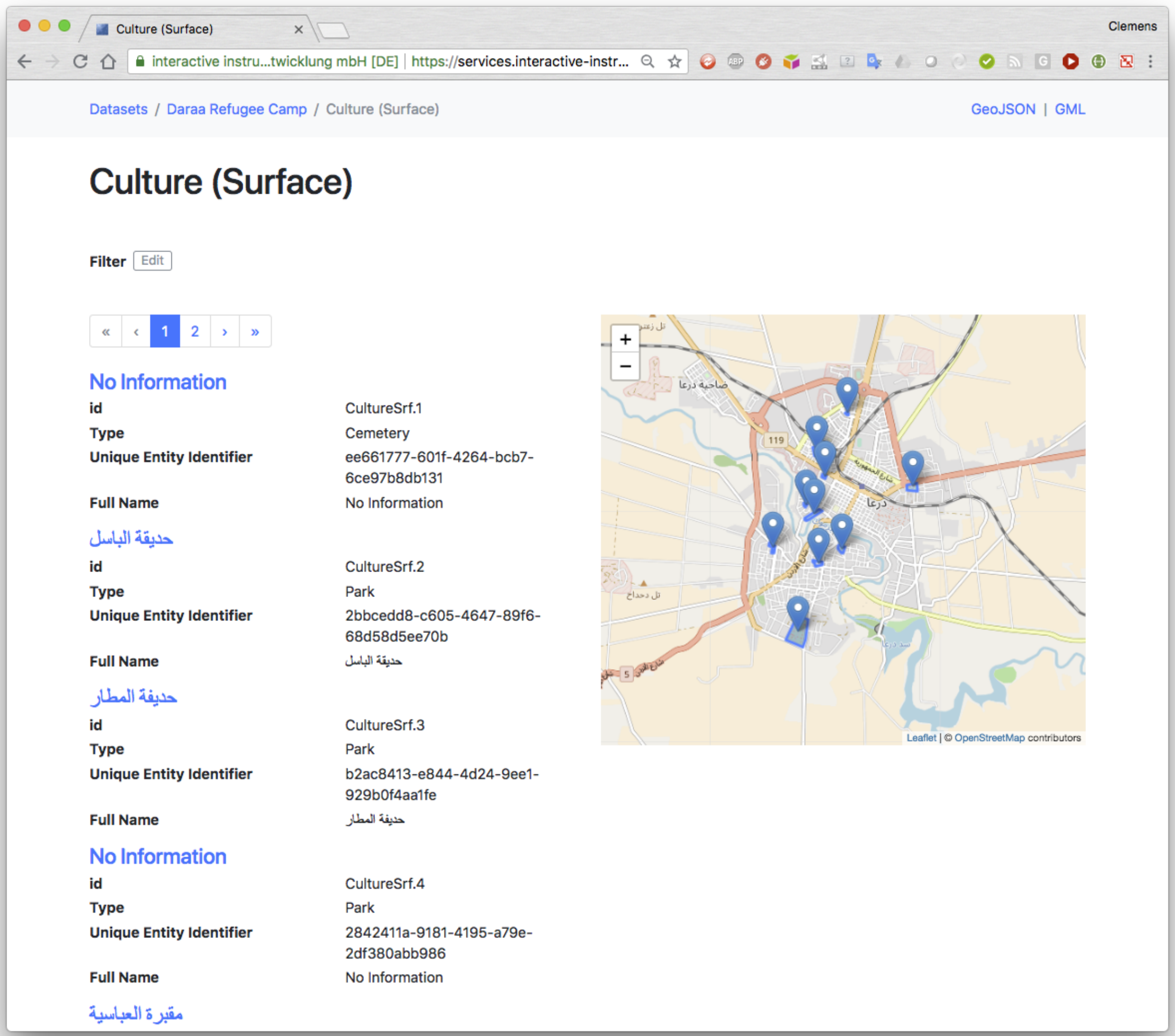
CultureSrf.1

id	CultureSrf.1
ara	-999999.00000000
ccn	No Information
cdr	No Information
f_code	AK120
lmc	-999999
lzn	-999999.00000000
oth	noInformation
pcf	-999999
sax_rs1	No Information
sax_rs2	No Information
sax_rs3	noInformation
sax_rs4	noInformation
sax_rs5	No Information
sax_rs6	noInformation
sax_rs8	No Information
sax_rs9	No Information
sax_rx1	noInformation
sax_rx2	noInformation
sax_rx5	noInformation
sax_rx6	noInformation
sax_rx7	noInformation
sax_rx8	noInformation
sax_rx9	No Information
sax_ry0	No Information
sax_ry1	noInformation

Figure 38. ldproxy items of a collection (cultural surfaces) in HTML, before configuration

The page displays the features with selected attributes (by default all attributes, this will be changed in the service configuration) the prev and next links, a map with all features from that page, links to alternate representations, and options to enter bbox, time and other filter parameters.

The screenshot is from the Zaatari dataset, without additional configuration i.e., the data is presented as published in the WFS 2.0 server.


Culture (Surface)

interactive instru...twicklung mbH [DE] | https://services.interactive-instr...

Datasets / Daraa Refugee Camp / Culture (Surface) GeoJSON | GML

Culture (Surface)

Filter Edit

« ‹ 1 2 › »

No Information

id	CultureSrf.1
Type	Cemetery
Unique Entity Identifier	ee661777-601f-4264-bcb7-6ce97b8db131
Full Name	No Information

حديقة الباسل

id	CultureSrf.2
Type	Park
Unique Entity Identifier	2bbcedd8-c605-4647-89f6-68d58d5ee70b
Full Name	حديقة الباسل

حديقة المطار

id	CultureSrf.3
Type	Park
Unique Entity Identifier	b2ac8413-e844-4d24-9ee1-929b0f4aa1fe
Full Name	حديقة المطار

No Information

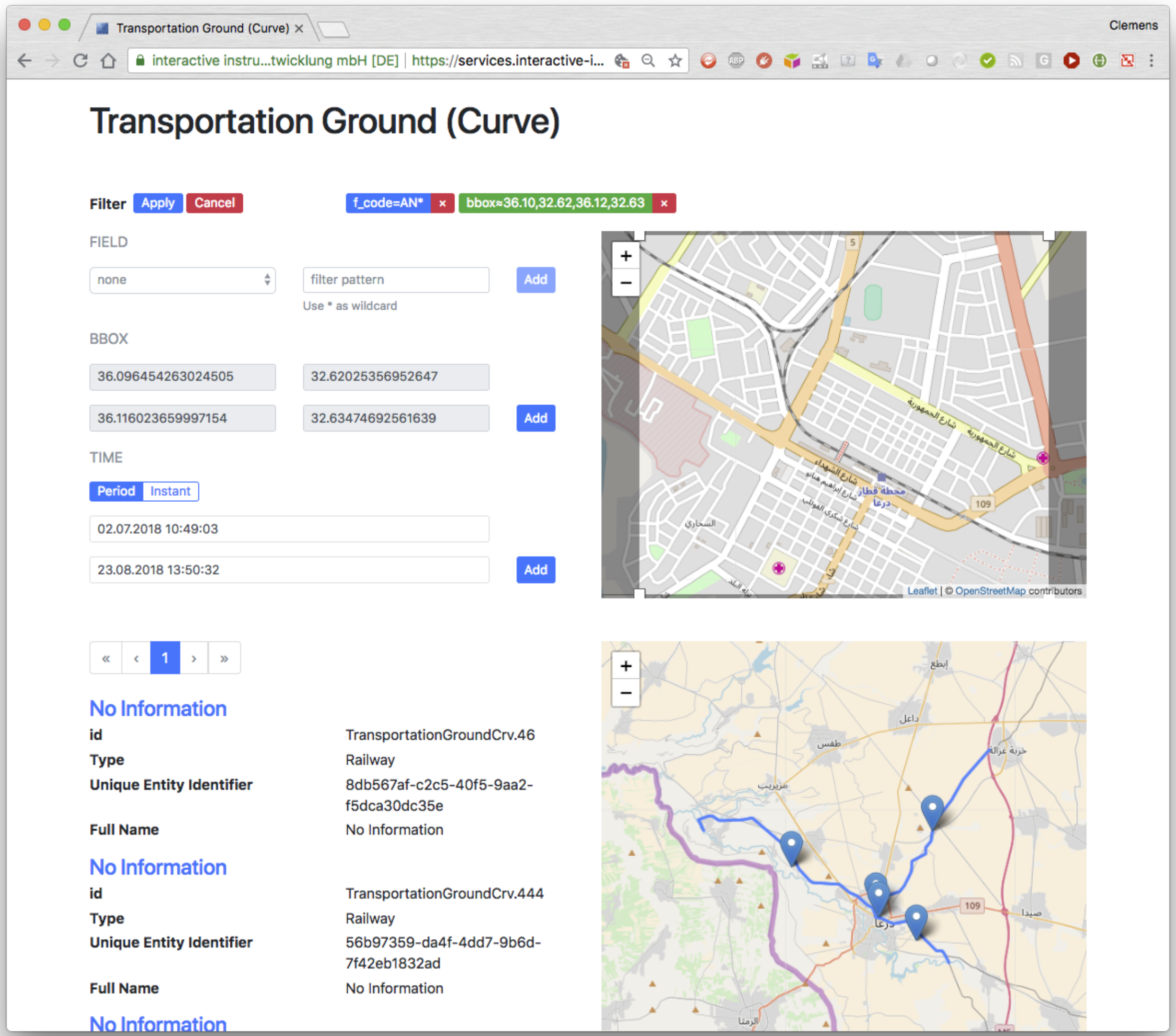
id	CultureSrf.4
Type	Park
Unique Entity Identifier	2842411a-9181-4195-a79e-2df380abb986
Full Name	No Information

مقبرة العباسية

Figure 39. ldproxy items of a collection (cultural surfaces) in HTML, after configuration

This figure shows the same feature type in the Daraa dataset after the configuration of the WFS 3.0 proxy service to improve the usability of the data. Attributes that are never present are not shown, labels have been changed to more human-friendly text, code values have been translated and the name has been set up as the label of a feature.

The form for filtering is shown in the next figure:


Transportation Ground (Curve)
interactive instru...twicklung mbH [DE] | https://services.interactive-i...
Clemens
Leaflet | © OpenStreetMap contributors

Transportation Ground (Curve)

Filter Apply Cancel f_code=AN* × bbox=36.10,32.62,36.12,32.63 ×

FIELD

none | filter pattern | Add

Use * as wildcard

BBOX

36.096454263024505 | 32.62025356952647

36.116023659997154 | 32.63474692561639 | Add

TIME

Period Instant

02.07.2018 10:49:03

23.08.2018 13:50:32 | Add

« ‹ 1 › »

No Information

id	TransportationGroundCrv.46
Type	Railway
Unique Entity Identifier	8db567af-c2c5-40f5-9aa2-f5dca30dc35e
Full Name	No Information

No Information

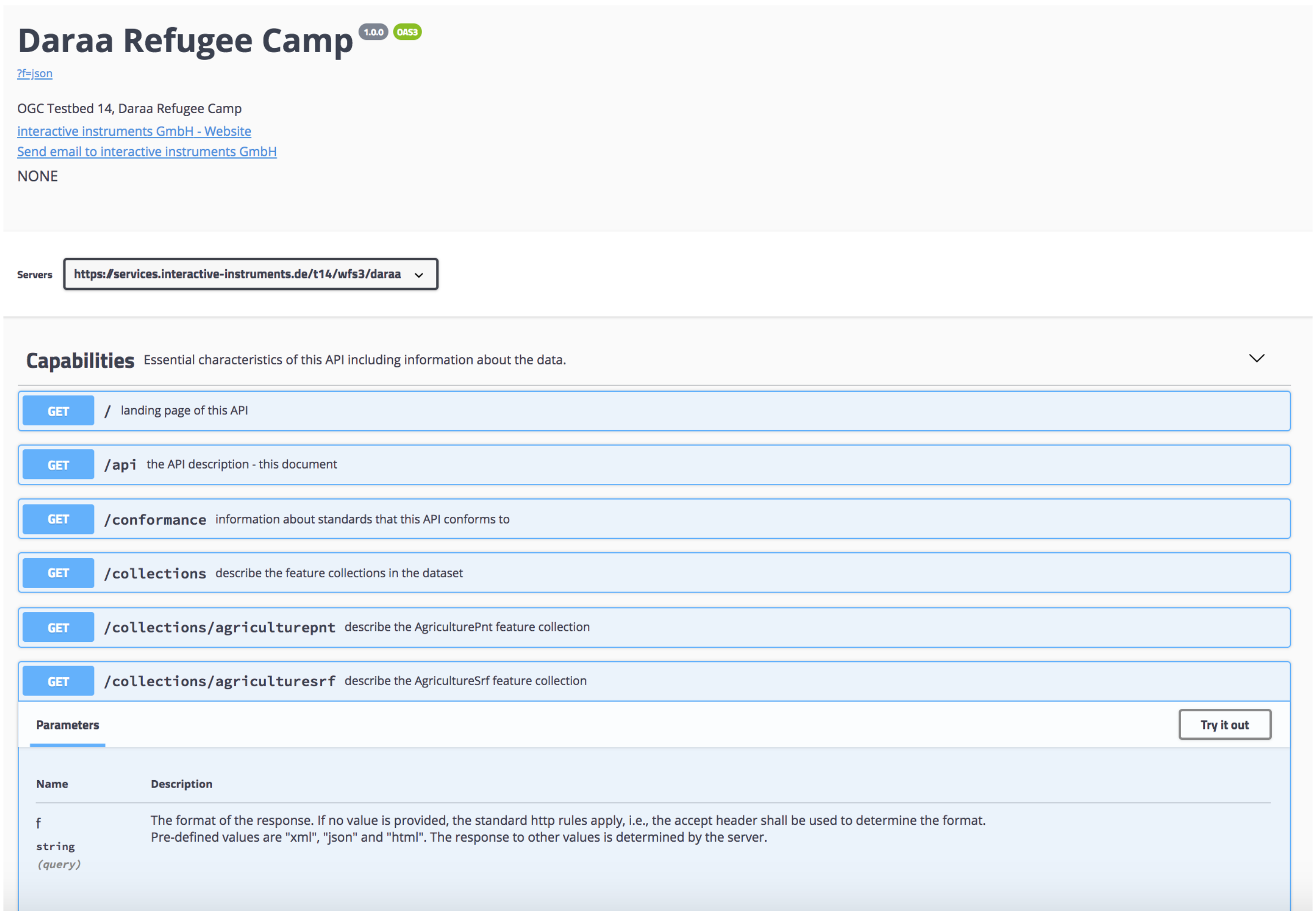
id	TransportationGroundCrv.444
Type	Railway
Unique Entity Identifier	56b97359-da4f-4dd7-9b6d-7f42eb1832ad
Full Name	No Information

No Information

Figure 40. ldproxy filtering items of a collection in HTML

Filters have been added to select railway features near the railway station.

Finally, for displaying the OpenAPI definition, ldproxy uses Swagger UI.


Daraa Refugee Camp 1.0.0 OAS3

?f=json

OGC Testbed 14, Daraa Refugee Camp

interactive instruments GmbH - Website

Send email to interactive instruments GmbH

NONE

Servers https://services.interactive-instruments.de/t14/wfs3/daraa

Capabilities Essential characteristics of this API including information about the data.

GET / landing page of this API

GET /api the API description - this document

GET /conformance information about standards that this API conforms to

GET /collections describe the feature collections in the dataset

GET /collections/agriculturepnt describe the AgriculturePnt feature collection

GET /collections/agriculturesrf describe the AgricultureSrf feature collection

Parameters Try it out

Name	Description
f string (query)	The format of the response. If no value is provided, the standard http rules apply, i.e., the accept header shall be used to determine the format. Pre-defined values are "xml", "json" and "html". The response to other values is determined by the server.

Figure 41. ldproxy OpenAPI definition in HTML

ldproxy exposes each collection explicitly so that feature attributes can be used for filtering.

Conformance Classes

All of the existing WFS 3.0 conformance classes are supported by the services:

- Core
- OpenAPI specification 3.0
- HTML
- GeoJSON
- Geography Markup Language (GML), Simple Features Profile, Level 0 / 2

The supported conformance classes are available at the path `/conformance` of each server.

Extensions

This section describes the extensions that have been included in the server.

CRS Support

The draft CRS extension [https://github.com/opengeospatial/WFS_FES/tree/master/extensions/crs] with the parameters `crs` and `bbox-crs` on `items` / `crs` on `items/{featureId}` is supported.

The following CRSs have been enabled in addition to the default CRS, WGS84 lon/lat [http://www.opengis.net/def/crs/OGC/1.3/CRS84]:

- WGS84 lat/lon [http://www.opengis.net/def/crs/EPSG/0/4326]
- WGS84 / Web Mercator [http://www.opengis.net/def/crs/EPSG/0/3857]
- WGS84 / World Mercator [http://www.opengis.net/def/crs/EPSG/0/3395]
- WGS84 / World Equidistant Cylindrical [http://www.opengis.net/def/crs/EPSG/0/4087]

Examples:

- Metadata for a feature collection with CRS information [https://services.interactive-instruments.de/t14/wfs3/daraa/collections/culturepnt?f=json]
- A feature in the Web Mercator projection [https://services.interactive-instruments.de/t14/wfs3/daraa/collections/culturepnt/items/CulturePnt.1?f=json&crs=http://www.opengis.net/def/crs/EPSG/0/3857]
- A feature in the World Mercator projection [https://services.interactive-instruments.de/t14/wfs3/daraa/collections/culturepnt/items/CulturePnt.1?f=json&crs=http://www.opengis.net/def/crs/EPSG/0/3395]
- Select features in a World Mercator bounding box [https://services.interactive-instruments.de/t14/wfs3/daraa/collections/culturepnt/items?f=json&crs=http://www.opengis.net/def/crs/EPSG/0/3395&bbox=4010000,3820000,4020000,3830000&bbox-crs=http://www.opengis.net/def/crs/EPSG/0/3395]

Geometry Simplification

A parameter `maxAllowableOffset` on `items` and `items/{featureId}` has been added. This simplifies geometries using the Douglas-Peucker algorithm and the parameter value is the maximum distance between the original line string and the simplified line string in the coordinate reference systems of the response.

NOTE This parameter follows the design in the GeoServices REST API. For developers and users it is probably easier to just use a zoom level. This is under discussion in Testbed-14 and will likely be changed to a different parameter definition. That parameter would then be converted internally to the distance value for the Douglas-Peucker algorithm.

The two datasets used are large scale data, so the benefits of using simplification for larger scale map representations are not so important in this case. In other cases the size of the data can be reduced significantly.

Example:

- a feature without simplification in HTML [https://services.interactive-instruments.de/t14/wfs3/daraa/collections/agriculturesrf/items/AgricultureSrf.7?f=html];
- the same feature with a slightly simplified geometry in HTML [https://services.interactive-instruments.de/t14/wfs3/daraa/collections/agriculturesrf/items/AgricultureSrf.7?f=html&maxAllowableOffset=0.003]

Extensions based on WFS 2.0 Capabilities

- A parameter `offset` on `items`: The optional offset parameter indicates the index within the result set from which the server shall begin presenting results in the response document. The first element has an index of 0. Minimum = 0. Default = 0.

  - In WFS 2.0, the parameter was called `startIndex`.

- A parameter `resultType` on `items`: This service will respond to a query in one of two ways (excluding an exception response). It may either generate a complete response document containing resources that satisfy the operation or it may simply generate an empty response container that indicates the count of the total number of resources that the operation would return. Which of these two responses is generated is determined by the value of the optional resultType parameter. The allowed values for this parameter are "results" and "hits". If the value of the resultType parameter is set to "results", the server will generate a complete response document containing resources that satisfy the operation. If the value of the resultType attribute is set to "hits", the server will generate an empty response document containing no resource instances. Default = "results".

  - An example: The number of Ground Transportation features with line geometries in the Daraa dataset (941) [https://services.interactive-instruments.de/t14/wfs3/daraa/collections/transportationgroundcrv/items?f=json&resultType=hits]

Security considerations

Before Testbed-14, the ldproxy software did not support any of the OpenAPI supported security schemes. To integrate the server into the security environment of the Testbed, the following implementation tasks were required:

- implementing the OAuth2/OpenID Connect workflow in the server;
- publish the details in the OpenAPI definition.

In addition, it was identified that a WFS 3.0 with HTML support is basically also a client and the client workflow needs to be implemented as well.

Another aspect that Participants learned is that the information about the OpenID Connect security scheme in the OpenAPI definition is not visible/supported in the HTML generated by Swagger UI.

This is different for other schemes. If JWT bearer authentication is used, the generated HTML has a capability to provide bearer tokens to access the API via the HTML client generated with Swagger UI.

The server uses HTTPS only.

IANA considerations

The standard media types and link relations are all supported.

7.2.3. D113 - Next Generation API Implementation (CubeWerx)

The CubeWerx D113 deliverables consist of servers deployed at the following endpoints:

- Unsecured, development, servers:
  - USGS Framework Data [http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/framework/api]

    - VMAP Level 0 Foundation Data [http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/foundation/api]
    - Daraa Refugee Camp [http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/daraa/api]
    - Zaztari Refugee Camp [http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/zaatari/api]
    - US Building Footprints [http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/buildings/api]
- Secured, stable, servers:
    - VMAP Level 0 Foundation Data [https://tb14.cubewerx.com/cubewerx/cubeserv/default/wfs/3.0.0/Foundation/api]
    - Daraa Refugee Camp [https://tb14.cubewerx.com/cubewerx/cubeserv/default/wfs/3.0.0/Daraa/api]
    - Zaatari Refugee Camp [https://tb14.cubewerx.com/cubewerx/cubeserv/default/wfs/3.0.0/Zaatari/api]

The usernames/password for the secured servers is **tb14guest/tb14guest**. The security protocols supported by the server are described in the Security Considerations clause.

Conformance Classes

All of the existing WFS 3.0 conformance classes are supported by the CubeWerx WFS 3.0 server deployed for Testbed-14:

- Core
- OpenAPI specification 3.0
- HTML
- GeoJSON
- Geography Markup Language (GML), Simple Features Profile, Level 0 / 2

The supported conformance classes are available at the /conformance path.

Extensions

The following extensions have been implemented by the CubeWerx WFS 3.0 server (D113) for Testbed-14:

- Coordinate reference systems (by reference) extension
- Geometry simplification extension
- Collections selection extension
- Property selection extension
- Asynchronous request extension
- Hierarchical path extension
- Map extension
- Tile extension
- OpenSearch query extension
- Advanced adhoc query extension

* Transaction extension

Security considerations

In Testbed-14 CubeWerx implemented methods for a WFS 3.0 to advertise security requirements objects and security scheme objects in its OpenAPI document.

As defined in OGC Web Feature Service 3.0: Path 1 - Core [https://rawgit.com/opengeospatial/WFS_FES/master/docs/17-069.html#_api_definition_2] a server's OpenAPI document can be accessed via the "/api" path.

Security Requirements Object

The security requirements object [https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.0.md#securityRequirementObject] is a declaration of which security mechanisms can be used to authorize resource access across the API.

The list of values includes alternative security requirement objects that can be used. Only one of the security requirement objects needs to be satisfied to authorize resource access. The field name of the security requirements object is "security". The following JSON fragment illustrates the top level security requirements objects:

```
...
"servers": [
  ...
],
"security": [
   { "apiKey": [ ] },
   { "cubewerxApiKey": [ ] },
   { "httpBasic": [ ] },
   { "httpBearer": [ ] },
   { "oauth2": ["profile", "openid", "email"] }
],
"paths": {
   ...
},
...
```

Individual operations can override this definition using a locally scoped security requirements object [https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.0.md#operation-object].

```
 "/collections": {
   "get": {
     "summary": "Undefined message",
     "description": "Undefined message",
     "operationId": "describeCollections",
     "tags": [
       "FeatureSets"
     ],
        "security": [
          { "apiKey": [ ] },
          { "oauth2": ["profile", "openid", "email"] }
        ],
        "parameters": [
          {
            "name": "f",
            "in": "query",
            "description": "A MIME type indicating the representation of the resources
to be presented (e.g. application/gml+xml; version=3.2 for GML 3.2).",
            "required": false,
            "allowEmptyValue": true,
            "schema": {
              "type": "string",
              "enum": [
                "json",
                "xml",
                "html"
              ]
            },
            "style": "form",
            "explode": true
          }
        ],
        "responses": {
          "200": {
            "description": "Undefined message",
            "content": {
              "application/json": {
                "schema": {
                  "$ref": "#/components/schemas/content-json"
                }
              },
              "application/xml": {
                "schema": {
                  "$ref": "#/components/schemas/content-xml"
                }
              },
              "text/html": {
                "schema": {
                  "type": "string"
                }
              }
            }
          },
          "default": {
            "description": "Undefined message",
            "content": {
              "application/json": {
                "schema": {
                  "$ref": "#/components/schemas/exception-json"
                }
              },
              "application/xml": {
                "schema": {
                  "$ref": "#/components/schemas/exception-xml"
                }
              },
              "text/html": {
                "schema": {
                  "type": "string"
                }
              }
            }
          }
        }
      }
    }
```

Security scheme object

Each security requirement (i.e. "apiKey", "cubewerxApiKey", etc.) specified at the API or operation level must reference a security scheme object [https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.0.md#securitySchemeObject] defined in the components section [https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.0.md#componentsObject] of a server's OpenAPI document.

Supported schemes are HTTP authentication, an API key (either as a header or as a query parameter), OAuth2's common flows (implicit, password, application and access code) as defined in RFC 6749 [https://tools.ietf.org/html/rfc6749], and OpenID Connect Discovery [https://tools.ietf.org/html/draft-ietf-oauth-discovery-06].

The following JSON fragment illustrates a security schemes object in the components section of a WFS's OpenAPI document:

```
 ...
"components": {
  "parameters": {
  ...
  },
  "securitySchemes": {
    "apiKey": {
      "type": "apiKey",
      "name": "apiKey",
      "in": "query"
    },
    "cubewerxApiKey": {
      "type": "apiKey",
      "name": "CubeWerx-API-Key",
      "in": "header"
    },
    "httpBasic": {
        "scheme": "Basic",
        "type": "http"
      },
      "httpBearer": {
        "scheme": "Bearer",
        "type": "http"
      },
      "cwauth": {
        "schema": "CwAuth",
        "type": "http",
        "description": "CubeWerx-specific scheme.  Pass value of CW_CREDENTIALS
cookie."
      },
      "oauth2": {
        "type": "oauth2",
        "flows": {
          "implicit": {
            "authorizationUrl": "https://tb14.cubewerx.com/cubewerx/oauth/authorize",
            "scopes": {
              "profile": "requests access to the end-user's profile",
              "openid": "OpenID Connect scope",
              "email": "requests access to the end-user's e-mail address"
            }
          },
          "password": {
            "tokenUrl": "https://tb14.cubewerx.com/cubewerx/oauth/token",
            "scopes": {
              "openid": "OpenID Connect scope",
              "profile": "requests access to the end-user's profile",
              "email": "requests access to the end-user's e-mail address"
            }
          },
          "authorizationCode": {
            "authorizationUrl": "https://tb14.cubewerx.com/cubewerx/oauth/authorize",
            "tokenUrl": "https://tb14.cubewerx.com/cubewerx/oauth/token",
            "scopes": {
              "openid": "OpenID Connect scope",
              "profile": "requests access to the end-user's profile",
              "email": "requests access to the end-user's e-mail address"
            }
          }
        }
      }
    }
  },
  ...
```

Issues

Access to a secured WFS requires that a client reads a server’s OpenAPI document at `/api` in an unsecured way in order to understand the security controls that the server supports. This view of the OpenAPI cannot be the **normal** document that the server generates because it would introduce a covert channel likely disclosing access controlled information such as the list of collections that the server offers.

This issue is under discussion in the WFS/FES SWG and the current thinking is that in the anonymous case, the server's OpenAPI document could report support for two Security Requirement Objects. One would indicate support for anonymous access, the other for your selected authentication method. According to the OpenAPI specification [https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.2.md], "The Paths MAY be empty, due to ACL constraints." So under anonymous access you would not see any of the details of the Paths objects. The complete discussion can be found here [https://github.com/opengeospatial/WFS_FES/issues/135].

IANA considerations

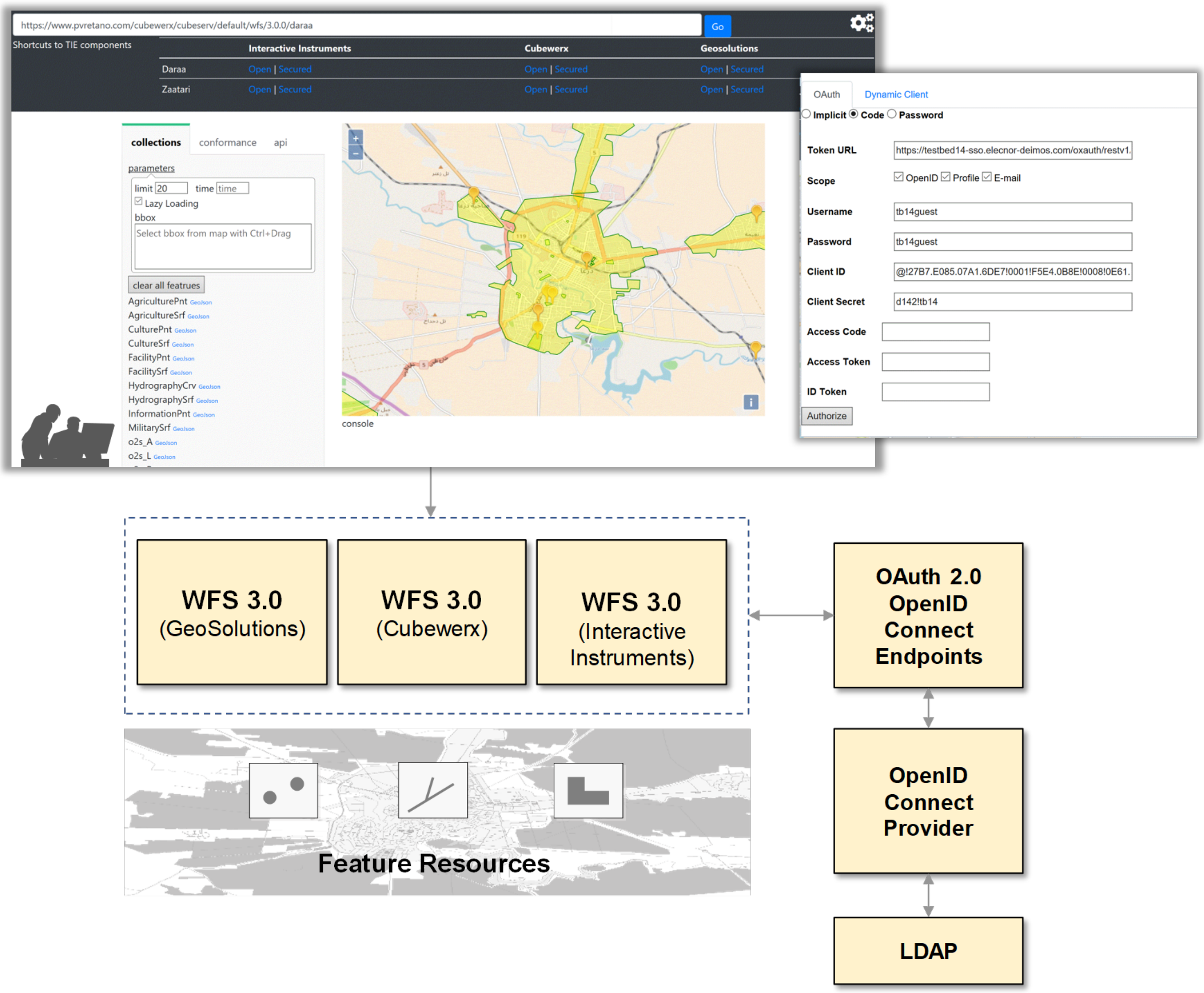
Standard media types and link relations are supported.

CORS considerations

CubeWerx has implemented the Cross-Origin Resource Sharing (CORS) mechanism as detailed in "http://www.w3.org/TR/cors/". Specifically, we support the "Access-Control-Request-Method" and "Origin" HTTP request headers, and generate the "Access-Control-Allow-Origin", "Access-Control-Allow-Credentials", "Access-Control-Max-Age", "Access-Control-Allow-Methods", "Access-Control-Allow-Header" and "Access-Control-Expose-Headers" HTTP response headers in accordance with the CORS mechanism. This includes full support for preflight requests. The list of Origins for from which credentialed access is allowed is fully configurable, and the entire mechanism can be disabled if necessary (e.g., to support older environments that are not compatible with CORS).

7.2.4. D142 - Next Generation API Client Implementation (GIS FCU)

This subsection describes the Testbed-14 client application for secure WFS 3.0.



https://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/daraa

	Interactive Instruments	Cubewerx	Geosolutions
Daraa	Open | Secured	Open | Secured	Open | Secured
Zaatari	Open | Secured	Open | Secured	Open | Secured

limit 20 time time

Select bbox from map with Ctrl+Drag

AgriculturePnt GeoJson
AgricultureSrf GeoJson
CulturePnt GeoJson
CultureSrf GeoJson
FacilityPnt GeoJson
FacilitySrf GeoJson
HydrographyCrv GeoJson
HydrographySrf GeoJson
InformationPnt GeoJson
MilitarySrf GeoJson
o2s_A GeoJson
o2s_L GeoJson

Implicit [✓ Code] Password

Token URL	https://testbed14-sso.elecnor-deimos.com/oxauth/restv1.
Scope	✓ OpenID ✓ Profile ✓ E-mail
Username	tb14guest
Password	tb14guest
Client ID	@!27B7.E085.07A1.6DE7!0001!F5E4.0B8E!0008!0E61.
Client Secret	d142!tb14
Access Code	
Access Token	
ID Token	

Figure 42. Next Generation API Client Implementation

To align with Testbed-14 objectives this deliverable implements at least the following interfaces defined in WFS 3.0 Core.

- the API Definition (path /api),
- the Conformance statements (path /conformance),
- the Dataset Distribution metadata (path /collections).

For security, the client uses the same logic as when requesting unprotected WFS 3.0 resources, but just "appends" the security requirements before requests. An overview of the security integration is described the following illustration:

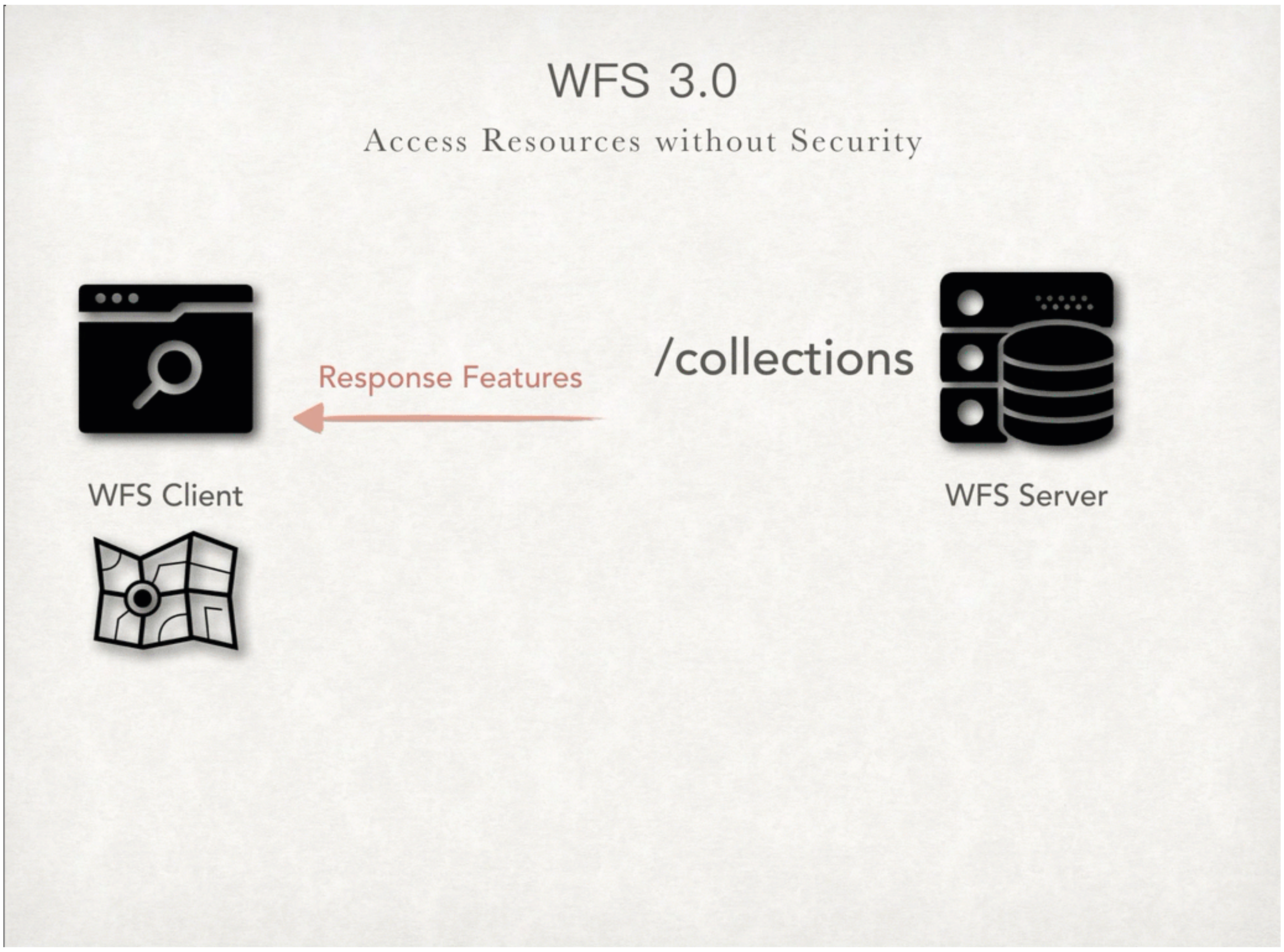


Figure 43. WFS 3.0 Security Overview (without security)

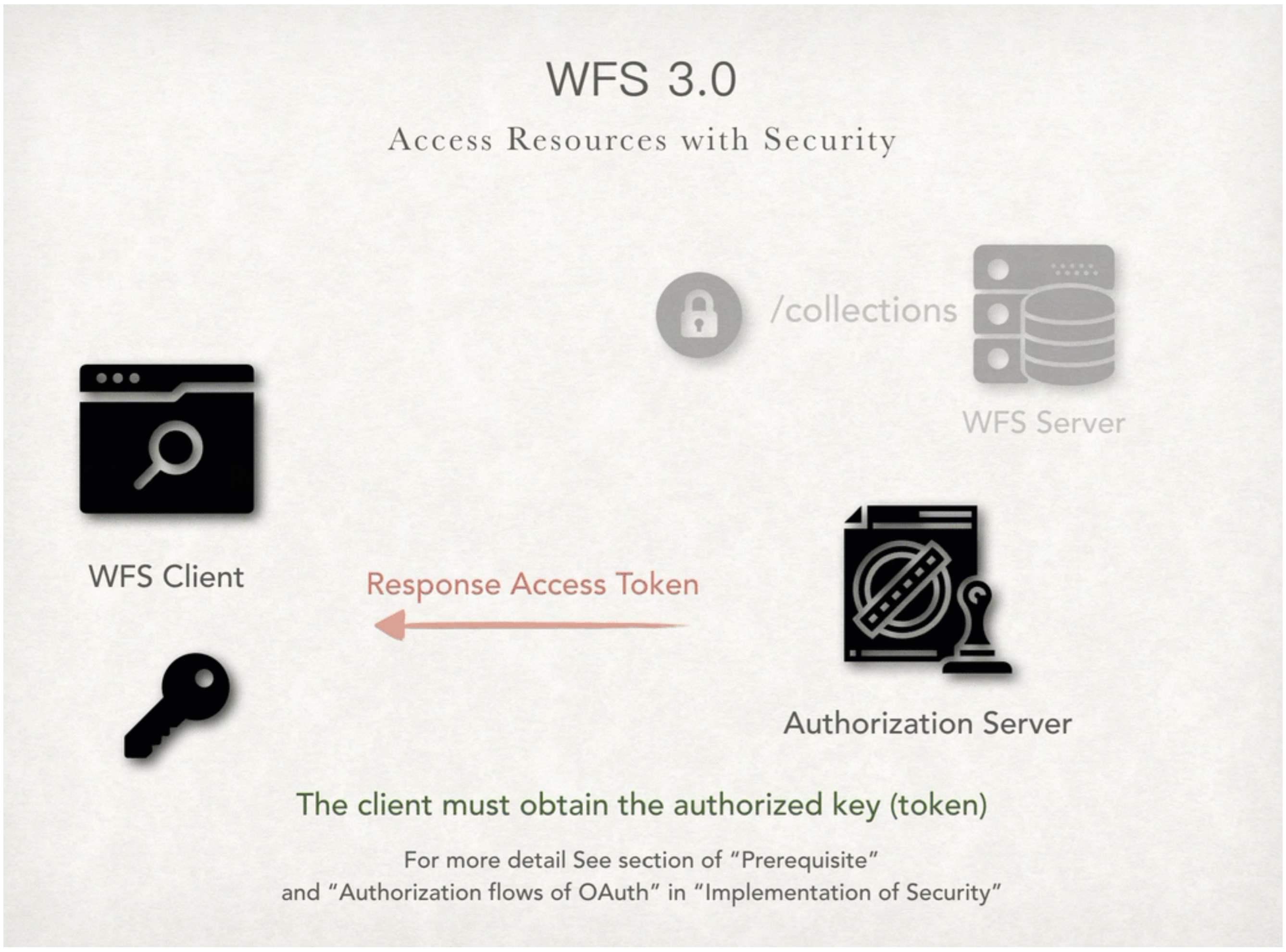

WFS 3.0
Access Resources with Security
/collections
WFS Server
WFS Client
Response Access Token
Authorization Server
The client must obtain the authorized key (token)
For more detail See section of "Prerequisite"
and "Authorization flows of OAuth" in "Implementation of Security"

Figure 44. WFS 3.0 Security Overview (with security)

NOTE An animated image illustrating the security overview can be found here [http://www.pvretano.com/Projects/tb14/WebAPIs/EngineeringReports/D021/FCU_animated_GIFs/fcu_wfsSecurityOverview.gif].

Implementing Security in a Client Application

OAuth 2.0 and OpenID Connect are the security mechanisms implemented in the WFS 3.0 Client. For details on the security aspects, please see Security Engineering Report [https://github.com/opengeospatial/D024-Security_Engineering_Report/blob/master/6-auth.adoc].

Implementation Questions

If an application needs to consume secured WFS, what should it do?

Before presenting the implementation discussion, there is information that needs to be collected. This section will use a Q&A method to introduce several questions that need to be answered, and the expected output of each answer. Such as:

- Is the application registered on the Authorization Server?
- What kind of client application is going to be implemented?

These questions are described in the following subsections.

Is the application registered on the Authorization Server?

The client application should be registered on the Authorization Server for proper permissions. After the registration, please keep the following mandatory parameters in mind:

a. Expected parameters gathered in this step
    i. client id
    ii. client secret
    iii. authorization URL
        - The endpoint on the Authorization Server to validate client credentials and grant the Authorization Code or Access Token.
    iv. token URL (only for Authorization Code Grant)
        - The endpoint on the Authorization Server to grant the Access Token.

NOTE - If the Authorization Server supports Dynamic Client Registration of OAuth2, the client can skip this registration. For more information, please refer to the section on Considerations with Dynamic Client Registration and OAuth 2.0

What kind of client application is going to be implemented?

For public client applications, we can choose OAuth Implicit Grant Flow, but it is also recommended to use Authorization Code Grant Flow without client secret for better security. For server-side apps, it is always recommended to use Authorization Code Grant Flow.

a. Expected parameters gathered in this step
    i. redirect uri
        - The URL where you are redirected back, and where you perform the callback() function.
    ii. nonce
        - client can send a random "nonce" value during authentication and check this value again after server response to avoid replay attack
    iii. scope
        - a set of claims (space separated) about the End-User information, only required if the client app needs the End-User information.
    iv. response type
        - To determine the authorization processing flow to be used, including what parameters are returned from the endpoints used

response_type	Flow
code	Authorization Code Flow
id_token	Implicit Flow
id_token token	Implicit Flow
code id_token	Hybrid Flow

response_type	Flow
code token	Hybrid Flow
code id_token token	Hybrid Flow

NOTE - OpenID Connect also defines a Hybrid Flow to retrieve OAuth 2.0 Multiple Response Type [https://openid.net/specs/oauth-v2-multiple-response-types-1_0.html], the procedures in the Hybrid Flow are almost the same as Authorization Code Grant except for the "response type" parameters. To simplify this discussion, we use Authorization Code Grant as the Hybrid Flow instead.

Example of Configuration for Implementations Questions

Given the Implementation Questions an example configuration is shown below.

```
{
    providerID: "Deimos",
    response_type: "id_token code",
    client_id: "xxx",
    client_secret: "yyy", //only for implicit grant
    redirect_uri: "https://map.gis.tw/testbed14/",
    authorization_url: "https://testbed14-sso.elecnor-
deimos.com/oxauth/restv1/authorize",
    token_url: "https://testbed14-sso.elecnor-deimos.com/oxauth/restv1/token",
    scope: ["openid","profile","email"],
    nonce: "123456"
}
```

The Authorization Flows of OAuth

This section describes the flows that grant a proper Access Token to the client prior to accessing resources.

Implicit Grant Flow

1. Client sends the request to the Authorization Server
2. Authorization Server Authenticates the End-User
3. Authorization Server obtains End-User Consent/Authorization
4. Authorization Server sends the End-User back to the Client with an ID Token and Access Token
5. Client validates the ID token and retrieves the End-User’s profile
6. Client uses Access Token to request WFS endpoints

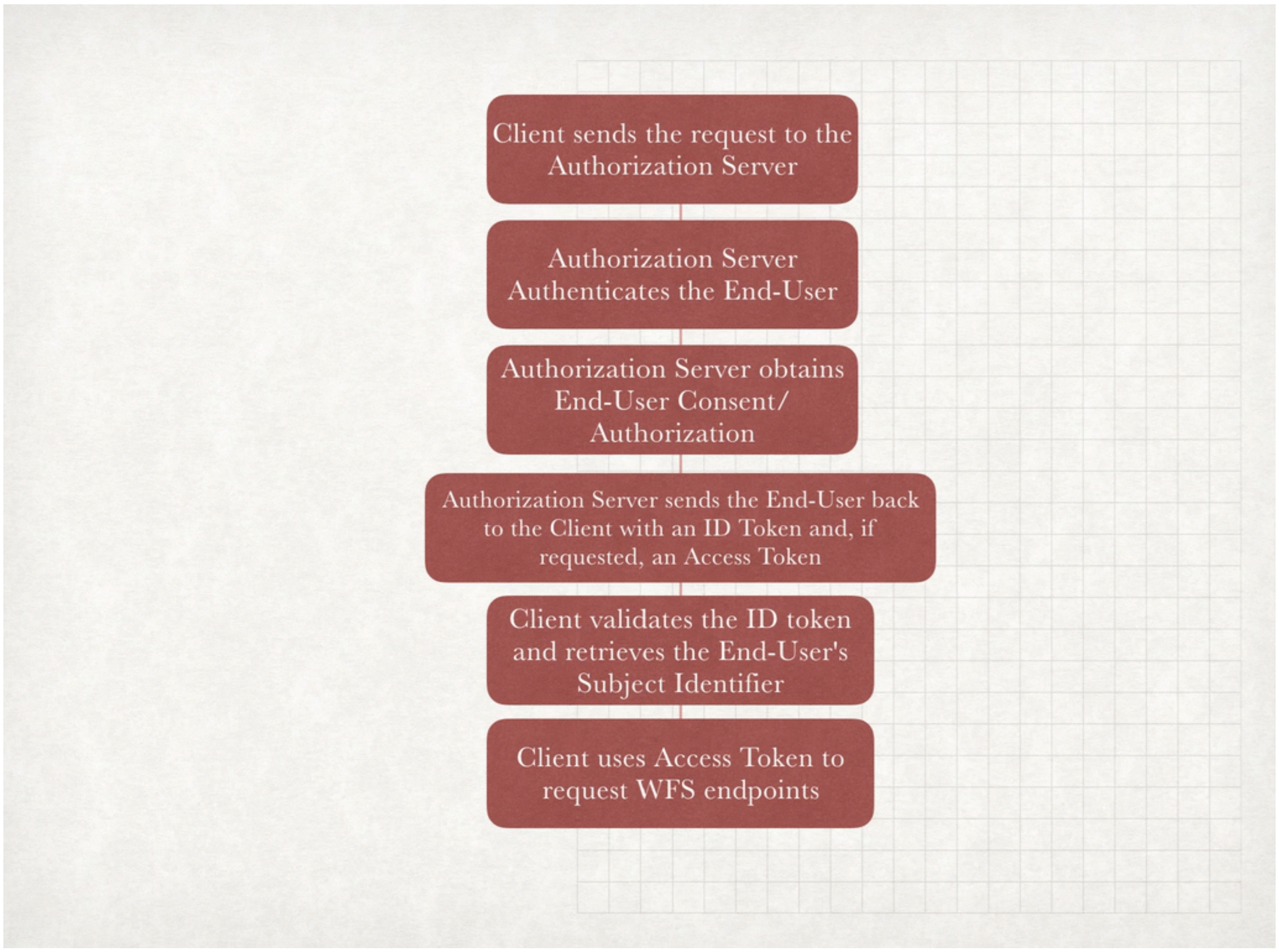

Client sends the request to the Authorization Server
Authorization Server Authenticates the End-User
Authorization Server obtains End-User Consent/ Authorization
Authorization Server sends the End-User back to the Client with an ID Token and, if requested, an Access Token
Client validates the ID token and retrieves the End-User's Subject Identifier
Client uses Access Token to request WFS endpoints

Figure 45. WFS with Implicit Grant Flow

NOTE An animated image illustrating the WFS with Implicit Grant Flow can be found here [http://www.pvretano.com/Projects/tb14/WebAPIs/EngineeringReports/D021/FCU_annimated_GIFs/fcu_implicit_grant.gif].

Authorization Code Grant Flow

1. Client sends the request to the Authorization Server
2. Authorization Server authenticates the End-User and obtains End-User Consent
3. Authorization Server sends the End-User back to the Client with an Authorization Code and, depending on the Response Type, one or more additional parameters
4. Client requests a response using the Authorization Code at the Token Endpoint
5. Client receives a response that contains an ID Token and Access Token in the response body
6. Client validates the ID Token and retrieves the End-User’s Subject Identifier
7. Client uses Access Token to request WFS endpoints

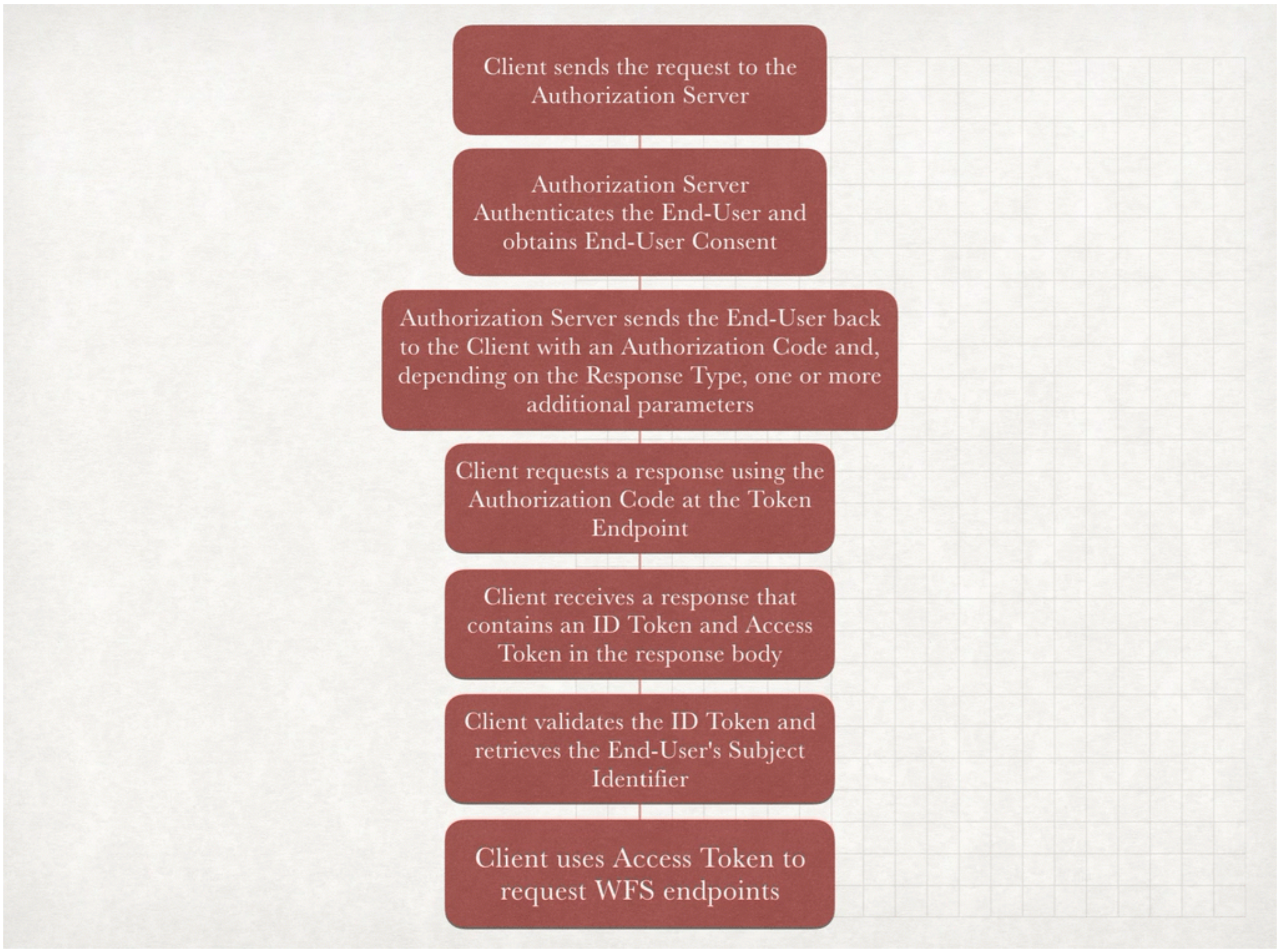

Client sends the request to the Authorization Server
Authorization Server Authenticates the End-User and obtains End-User Consent
Authorization Server sends the End-User back to the Client with an Authorization Code and, depending on the Response Type, one or more additional parameters
Client requests a response using the Authorization Code at the Token Endpoint
Client receives a response that contains an ID Token and Access Token in the response body
Client validates the ID Token and retrieves the End-User's Subject Identifier
Client uses Access Token to request WFS endpoints

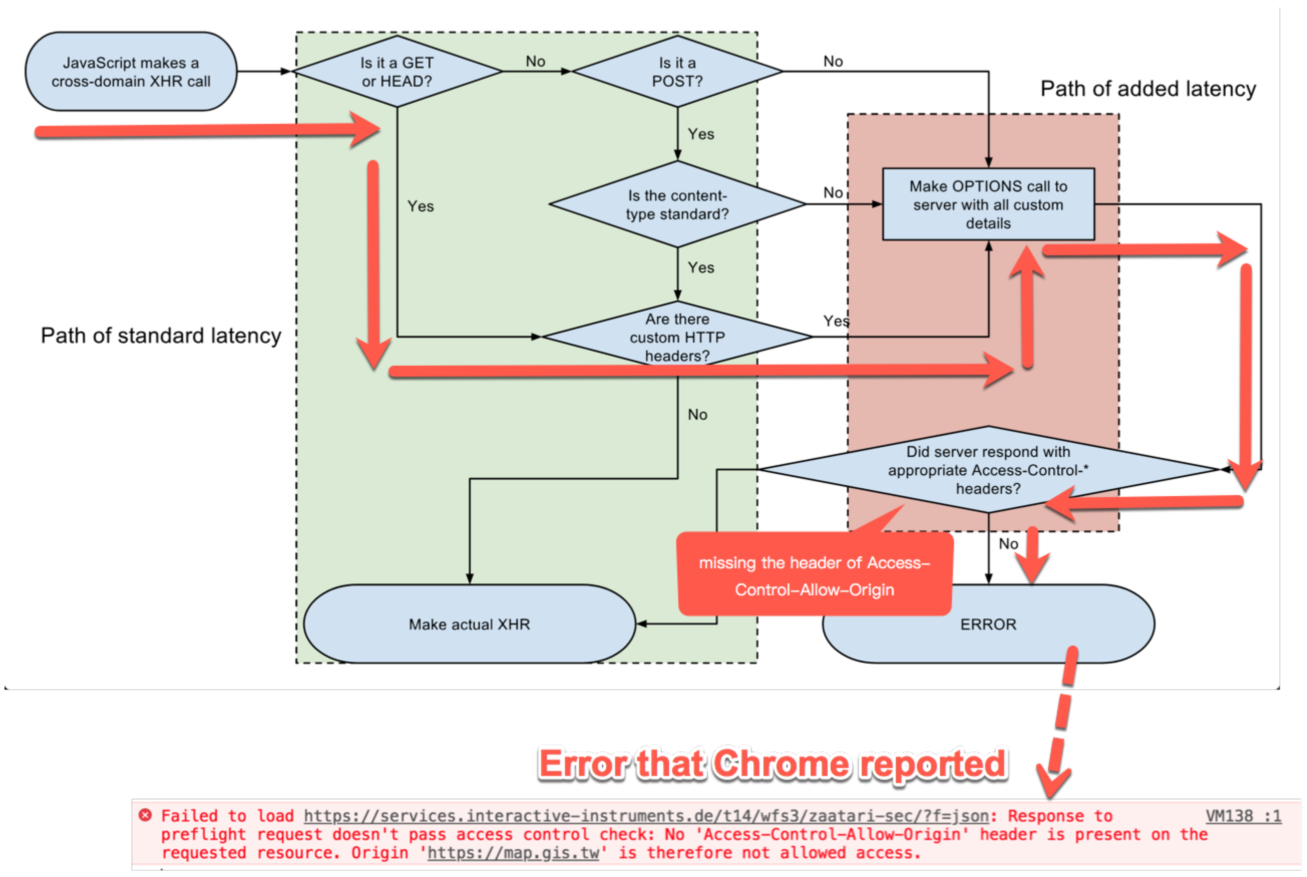
Figure 46. WFS with Authorization Code Grant Flow

NOTE An animated image illustrating the WFS with Authorization Code Grant Flow can be found here [http://www.pvretano.com/Projects/tb14/WebAPIs/EngineeringReports/D021/FCU_annimated_GIFs/fcu_authorization_code_grant.gif].

The Problem of CORS

For security reasons, modern browsers such as Chrome and Firefox send an OPTIONS request (aka. preflight [https://www.w3.org/TR/cors/#preflight-request]) to the API before sending the actual requests (GET/POST/PUT/DELETE) during Cross-Origin Resource Sharing (CORS) scenarios.

The CORS procedures for modern browsers are described in the following image. The red arrows are the flow that makes the problem occur.


JavaScript makes a cross-domain XHR call
Is it a GET or HEAD?
No
Is it a POST?
No
Path of added latency
Yes
Yes
Is the content-type standard?
No
Make OPTIONS call to server with all custom details
Path of standard latency
Yes
Are there custom HTTP headers?
Yes
No
Did server respond with appropriate Access-Control-* headers?
missing the header of Access–Control–Allow–Origin
No
Make actual XHR
ERROR
Error that Chrome reported

```
Failed to load https://services.interactive-instruments.de/t14/wfs3/zaatari-sec/?f=json: Response to          VM138 :1
preflight request doesn't pass access control check: No 'Access-Control-Allow-Origin' header is present on the
requested resource. Origin 'https://map.gis.tw' is therefore not allowed access.
```

Figure 47. The CORS procedures for modern browsers

Possible Solutions to the Problem of CORS

Browsers require the following headers that make the pre-API request work. Either server-side and client-side should consider supporting these headers with proper settings.

- Access-Control-Allow-Origin
  - the domain that makes the request, like https://map.gis.tw
- Access-Control-Allow-Headers
  - value of “Authorization”
- Access-Control-Allow-Methods
  - at least of “GET”, “OPTIONS”

Pagination

To enable pagination-like functions, the client should implement the "limit" parameter in the WFS Core, and the parameter of "offset" in the WFS Extensions.

- Example of the query string to retrieve features on the page 2

```
/collections/{collection_id}/items?f=json&limit=100&offset=100
```

If the WFS offset extension is not supported, using "next" and "prev" links defined in the Core are another alternative to load pages one by one. The "lazy-loading" mechanism, which loads the pages in the background, may be considered as an implementation option in the client as well.

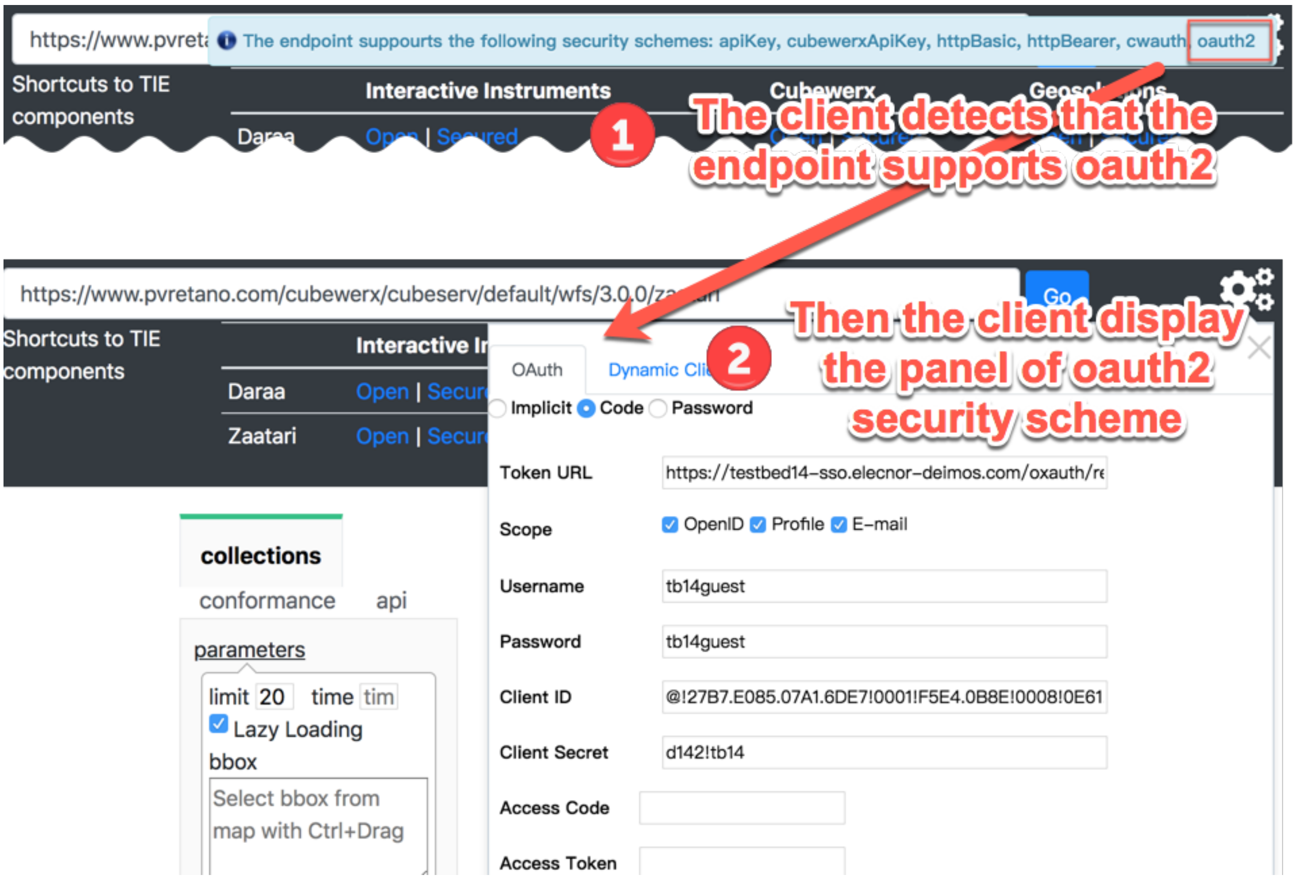
Security Scheme Object of OpenAPI 3.0

While a variety of security schemes [https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.0.md#securitySchemeObject] are possible, OAuth is not the only option for OpenAPI.

Both server and client sides should negotiate the security scheme.

pseudo code of how client determine which Security Schemes to negotiate

```
if the client detects the API metadata contains Security Scheme Objects {
  if the security schemes are supported by the client {
    show security scheme
  } else {
    hide the security schemes that were not supported
  }
}
if the client cannot detects the Security Scheme Objects {
  the client will show all the security schemes that the client supports by default.
}
```


The endpoint suppourts the following security schemes: apiKey, cubewerxApiKey, httpBasic, httpBearer, cwauth, oauth2
1
The client detects that the endpoint supports oauth2
2
Then the client display the panel of oauth2 security scheme
Shortcuts to TIE components
Interactive Instruments
Cubewerx
Geosolutions
Daraa
Zaatari
Open | Secured
Go
collections
conformance
api
parameters
OAuth
Dynamic Cli

https://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/za

limit 20 time tim
☑ Lazy Loading
bbox
Select bbox from map with Ctrl+Drag

○ Implicit ● Code ○ Password

Token URL	https://testbed14-sso.elecnor-deimos.com/oxauth/re
Scope	☑ OpenID ☑ Profile ☑ E-mail
Username	tb14guest
Password	tb14guest
Client ID	@!27B7.E085.07A1.6DE7!0001!F5E4.0B8E!0008!0E61
Client Secret	d142!tb14
Access Code	
Access Token	

Figure 48. Illustrating how to display which Security Schemes that the endpoint supports

Considerations when using 3rd party libraries to implement security

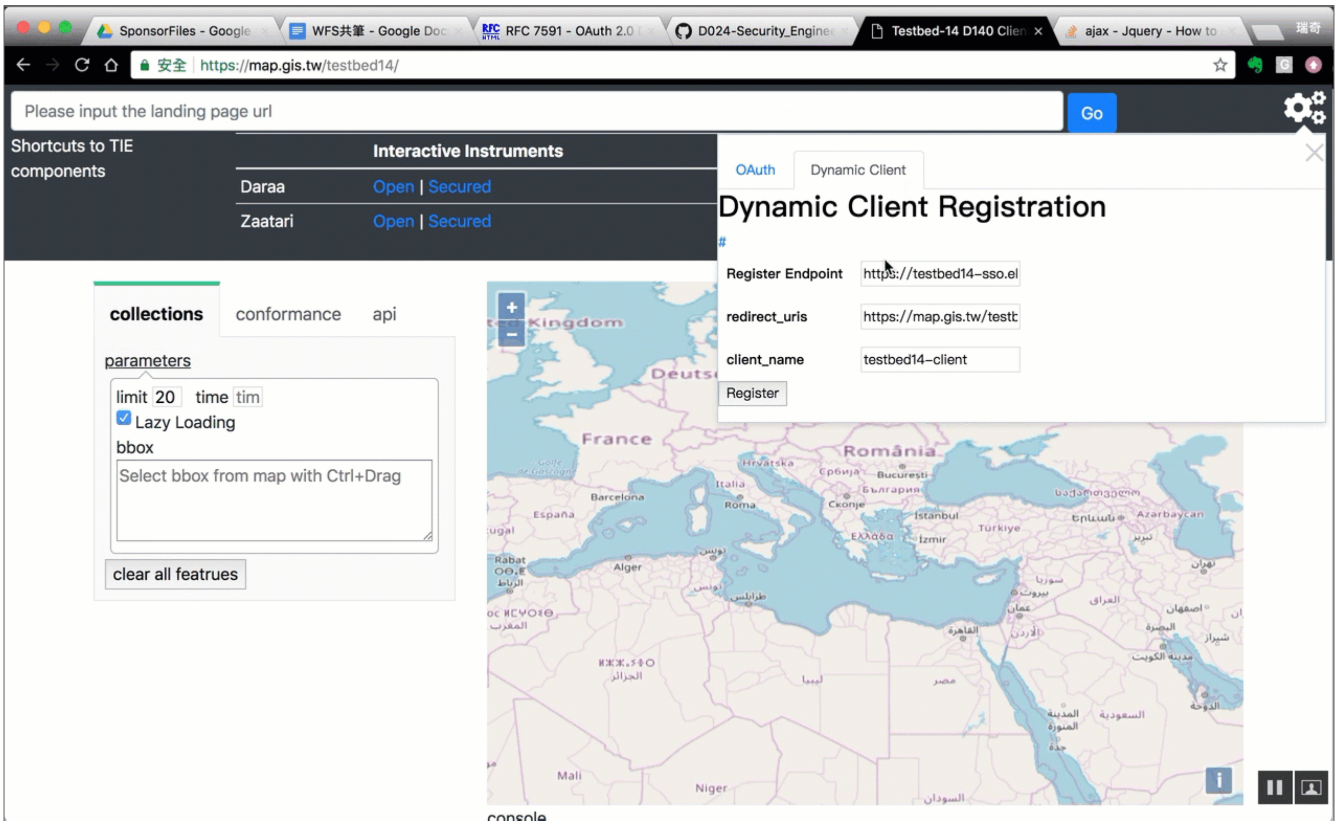
- Pros: Reduce learning curve on securities.
- Cons: For clients using 3rd party library to implement OAuth, it’s hard to debug if the communication between server and client has some problems.

Considerations with Dynamic Client Registration and OAuth 2.0

If the Authorization Server supports Dynamic Client Registration of OAuth 2.0, the client does not need to register itself before the authorization process.

Steps of Integrating Dynamic Client Registration

- 1) Register the client on the Authorization server
- 2) Use the returned credential (replace the default one) to authorize the OpenAPI.


Dynamic Client Registration

Register Endpoint: https://testbed14-sso.el

redirect_uris: https://map.gis.tw/testb

client_name: testbed14-client

Figure 49. Illustrating the flow of Dynamic Client Registration

NOTE An animated image illustrating the flow of Dynamic Client Registration can be found here [http://www.pvretano.com/Projects/tb14/WebAPIs/EngineeringReports/D021/FCU_annimated_GIFs/fcu_dynamicClientRegistration.gif].

Conformance Classes

- WFS 3.0 Core
- OpenAPI specification 3.0
- HTML
- GeoJSON

Extensions

- Parameter `offset` on `items` (see more detail in Pagination)

Security considerations

- SSL Support
- Logout Mechanism
    - In order to release authorized status (cookie, session, etc), a logout function should be considered.

IANA considerations

- For GeoJSON format, either legacy "application/json" or IETF "application/geo+json" should be considered to be supported.

Chapter 8. Extensions

During Testbed-14, a number of experiments in extending WFS 3.0 Core were implemented and tested. The specific list of extensions is:

- Coordinate reference systems (by reference) extension
- Geometry simplification extension
- Collections selection extension
- Property selection extension
- Asynchronous request extension
- Hierarchical path extension
- Map extension
- Tile extension
- OpenSearch query extension
- Advanced adhoc query extension
- Transaction extension

The implementation details for each extension can be found in Annex A.

Chapter 9. Findings

The objective of the Next Generation Web APIs effort in Testbed-14 was to develop and test WFS 3.0. The goal was to experiment with the new WFS 3.0 specification, OpenAPI and to add security mechanisms based on OpenID Connect and OAuth 2.0.

The evidence obtained through the Testbed-14 Next Generation Web APIs effort supports the following main findings:

- It is possible to develop and deploy multiple OGC WFS 3.0 with a simple core specified as OpenAPI components.
- It is possible to implement multiple OGC WFS 3.0 with modular extensions for complex functions.
- It is possible to use well-known resource types, and deploy geospatial feature resources on multiple OGC WFS 3.0.
- It is possible to deploy facades on selected WFS 3.0 to assist legacy service transition to next generation APIs.
- It is possible to develop and deploy client applications able to access geospatial feature resources using HTTP methods. Leverage openly available, browser-based tools such as Swagger to exercise the OpenAPI components of WFS 3.0.
- It is possible to secure the WFS 3.0 clients and services via an Authorization Server using OpenID Connect and OAuth 2.0.

9.1. Recommendations for Future Work

As a result of the work performed during this Testbed, several future work points have been identified:

- Assess the ability of secure WFS 3.0 extensions to support transactions against geospatial feature resources, geometry simplification and other functions.
- Experiment with OpenAPI and security mechanisms based on OAuth 2.0 as a Profile of a WMTS implemented as reusable OpenAPI components.
- Assess the ability of secure WFS 3.0 extensions to support access control and security metadata, optionally enclosed within dissemination formats for binding assertion metadata with data resources.
- Assess the ability of secure WFS 3.0 extensions to describe and deliver emerging forms of geospatial resources such as Vector Tiles.
- Assess how security specifications, access control and dissemination may further enable JSON, HTML and Vector Tiles-based information exchange.

Appendix A: Extensions

This annex describes the various experiments in extending WFS 3.0 that were implemented and tested during Testbed-14.

The following list of extension are described:

- Coordinate reference systems (by reference) extension
- Geometry simplification extension
- Collections selection extension
- Property selection extension
- Asynchronous request extension
- Hierarchical path extension
- Map extension
- Tile extension
- OpenSearch query extension
- Advanced adhoc query extension
- Transaction extension

A.1. Coordinate reference systems (by reference) extension

See https://github.com/opengeospatial/WFS_FES/tree/master/extensions/crs.

A.2. Geometry simplification extension

A common usage pattern for a WFS is as a data source for visualization. An issue that arises when used in this way is that the server has no information about the display scale and thus may, in certain situations, provide much more information in the response than is necessary to visually render the information.

The optional `resolution` parameter provides the server with the information necessary to suitably generalize geometries and features in the response for a specified resolution. This document does not describe a specific method or methods of generalization that a server might use. However, the CubeWerx server uses the Douglas-Peucker [https://en.wikipedia.org/wiki/Ramer-Douglas-Peucker_algorithm] algorithm.

The `resolution` parameter may be added to the following WFS paths:

- /collections/{collectionId}/items
- /collections/{collectionId}/items/{fid}

Clients making a `getFeature` request using the `resolution` parameter should be aware that geometries in the response document may be modified and have non-visible segments removed or that even sub-pixel (and thus not visible) features may be removed altogether from the response document.

The value of the resolution shall be either a single value denoting the resolution along all display dimensions or a list of double values corresponding to the resolution along each display dimension of the data. If a list of values is specified, the order of values shall be as specified by the definition of the nearest-to-scope coordinate reference system for the geometry or feature being processed.

Servers implementing this document shall understand the presence of the `resolution` parameter but are not obliged to use the information in generating a response document. In other words, the `resolution` parameter is a rendering hint that a sever may or may not use.

NOTE One possible outcome of generalization is dimensional collapse. This occurs when a subset of dimensions in a geometry are rendered non-visible by the generalization process. In this case, the entire geometry shall be considered non-visible.

Example: The following examples illustrate responses without and with geometry simplification.

- without simplified geometries: http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/framework/collections/NHDLIHI/items?f=application%2Fgeo%2Bjson&bbox=38.7952,-77.1336,39.0062,-76.9102
- with simplified geometries: http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/framework/collections/NHDLIHI/items?f=application%2Fgeo%2Bjson&bbox=38.7952,-77.1336,39.0062,-76.9102&resolution=13

The following image illustrates the effects of simplification graphically.

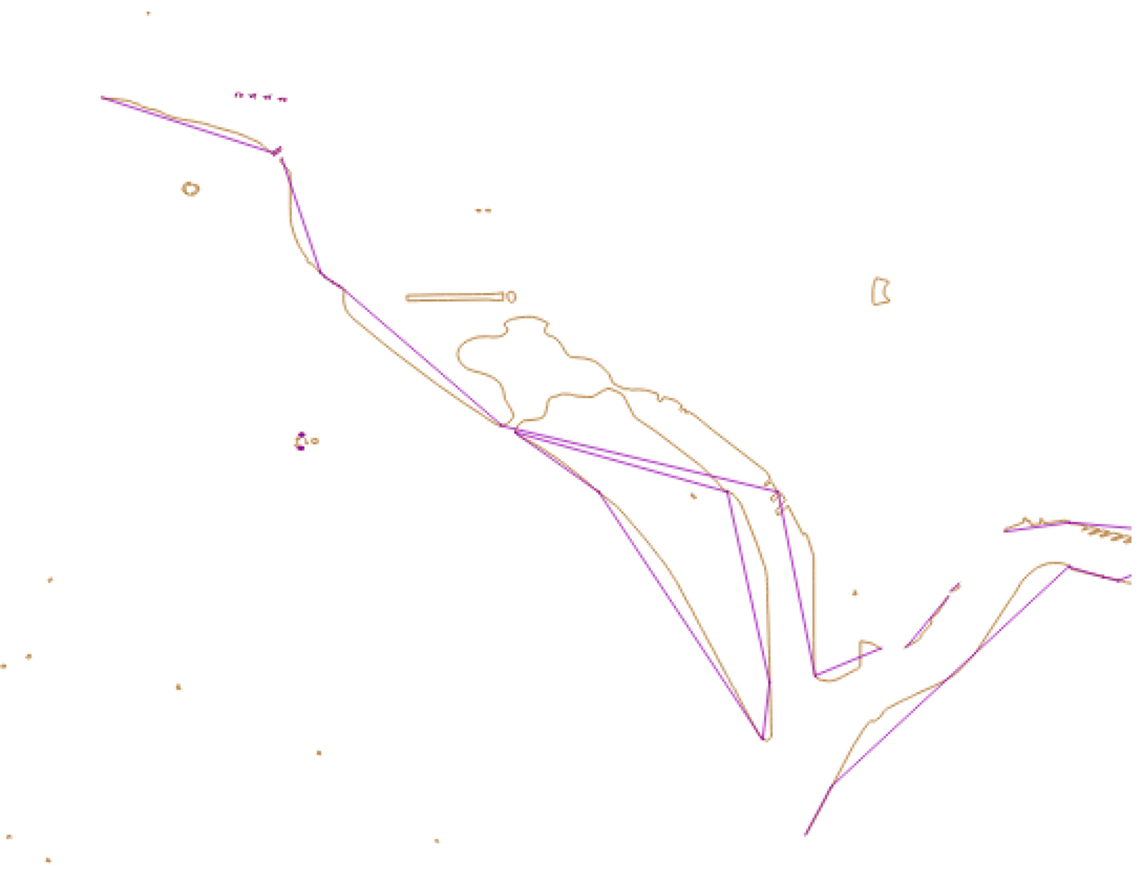


Figure 50. Simplified versus un-simplified WFS response

A.3. Collections selection extension

The collections selection extension defines a query parameter, named `collections`, that may be used to set the scope of a particular resource or operation to an enumerated subset of collections offered by a WFS. The following YAML fragment defines the parameter:

```
name: collections
in: query
description: A list of collection identifiers.
required: false
allowEmptyValue: false
schema:
  type: array
  items:
    type: string
    enum: {list of valid collection id's}
style: form
explode: false
```

The value of the `collections` parameter is a comma-separated list of collection identifiers. The domain of collection identifiers is from the set of collections offered by a WFS.

The `collections` parameter may be appended to the following set of resources defined in the OGC Web Feature Service 3.0: Part 1 - Core [https://github.com/opengeospatial/WFS_FES] specification:

- landing page (path = /)
- API definition (path = /api)
- Feature collections metadata (path = /collections)

The effect of appending this parameter to any of these paths is to limit the scope of the collection-specific components of the retrieved resource(s) to those collections specified as the value of the parameter. For example, the `/collections` resource from the CubeWerx [http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/daraa/collections] Testbed-14 server:

- http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/daraa/collections?f=json

lists over thirty collections. If, however, the `collections` parameter is appended to the resource:

- http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/daraa/collections?collections=AgricultureSrf,HydrographySrf,StructureSrf&f=json

the server only lists metadata for the 3 specified collections in the response.

In addition to the `collections` parameter, this extension defines a new top-level resource, `/items` that aggregates all the features offered by a server under a single path. Thus, for example, a client can fetch all the features from all collections within a particular AOI using the URL:

- http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/daraa/items?bbox=32.6228,36.0998,32.6304,36.1070.

Appending the `collections` parameter to this URL would further limit the scope of the operation by returning only features from the specified collections (i.e. AgricultureSrf, HydrographySrf, StructureSrf):

- http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/daraa/items?collections=AgricultureSrf,HydrographySrf,StructureSrf&bbox=32.6228,36.0998,32.6304,36.1070

The WFS 3.0 Core specification defines the following filtering parameters for features:

- limit
- bbox
- time
- filter parameters for feature properties

All of these parameters may be used with the `/limits` resources.

Filter parameters for feature properties merit particular attention because not all features are required to have the same schema. If a particular property filtering parameter is specified that does not exist in the schema of a particular feature then that predicate test shall evaluate to false while evaluating the overall query predicate.

NOTE

If and how a schema is defined for the `/items` resource was not formally investigated during the testbed. The CubeWerx WFS for Testbed-14, however, generates a schema for all collections aggregated under the `/items` resource via the `/items/schema` path.

A.4. Property selection extension

The default behaviour of a WFS is to present all feature properties in a response document. The property selection extension defines a parameter named `properties` that may be used to control which subset of optional properties are presented in a server's response. The following JSON fragment defines the `properties` parameter:

```
name: properties
in: query
description: A list of properties names.
required: false
allowEmptyValue: false
schema:
  type: array
  items:
    type: string
    enum: {list of valid property names}
style: form
explode: false
```

The value of the `properties` parameter is a comma-separated list of optional feature property names.

Attention is drawn to the use of the word "optional". Some output formats, such as GML, are bound by strict schema validation requirements that define certain properties as mandatory and others as optional. For such output formats, mandatory properties are always presented in the response regardless of the value of the `properties` parameter. Since mandatory properties will, for schema validation reasons, always be presented in the response, only the names of optional properties need be specified.

The list of possible optional feature properties that may be specified as elements of the value of the `properties` parameter may be determined by inspecting the schema of a feature which, as per Recommendation 7 in the WFS 3.0 Core specification, may be determined by resolving the link with 'rel=describedBy' in the feature collection metadata.

Example: http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/daraa/collections/AgricultureSrf/items?count=10&outputFormat=application%2Fgml%2Bxml&properties=f_code

A.5. Asynchronous request extension

This clause describes a light-weight protocol extension for WFS 3.0 for processing long-running operations asynchronously. Rather than the standard request-response invocation pattern – where a client submits a request and then waits to get the response from the server – an asynchronous request is executed in the background by the server and either a notification message is sent when operation processing has completed or (depending on the conformance classes implemented) the server may be polled periodically to determine the execution status of the request.

A.5.1. Conformance classes

Two conformance classes are defined for asynchronous request processing:

1. Asynchronous Processing
2. Asynchronous Polling

For the Asynchronous Processing conformance class, a server implements the ability to accept an asynchronous request, acknowledge that the request was successfully accepted, process the request in the background and finally send a notification — using the specified response handler — when the request's processing has been completed. Servers that implement this class may optionally provide a hypermedia control that may be used to cancel the asynchronous request after it has been invoked.

For the Asynchronous Polling conformance class, a server implements the ability to accept an asynchronous request, acknowledge that the request was successfully accepted, process the request in the background and allow the server to be polled periodically to obtain the execution status and progress of the request's execution. Once request processing has been completed a hypermedia control, provided by the server, will allow the request's response to be retrieved.

A.5.2. responseHandler parameter

Asynchronous request processing shall be triggered by the presence of the **responseHandler** parameter.

The **responseHandler** parameter may be added to the following WFS paths:

- /collections/{collectionId}/items
- /collections/{collectionId}/map

If the **responseHandler** parameter is present, the server shall immediately respond with an acknowledgement message that indicates that the request has been successfully accepted or an exception messages if there was a problem. At this point communication with the server shall terminate.

The server shall then proceed to process the request in the background, taking as much time as necessary to completed its processing.

For servers that implement the Asynchronous Processing class, when request processing has completed a notification message shall be sent using the scheme(s) specified as the value of the **responseHandler** parameter. The content of the notification message is discussed in clause Notification message content.

For servers that implement the Asynchronous Polling class, the acknowledgement message shall contain a means by which the server may be polled to determine the execution status of the request. When request processing has been completed, the acknowledgement message shall contain a means by which the request's response may be retrieved.

The value of the **responseHandler** parameter shall be a list of one or more values. The value of each element of the list shall either be a URI or the token "poll".

If a list element value of the **responseHandler** parameter is a URI, then its form shall be a valid expression of the one the notification schemes that the server claims to support in its API document.

This standard does not define a normative set of notification schemes but possible schemes include:

- Email scheme as per RFC 2368
  - Example — mailto:tb12@pvretano.com
- Sms scheme as per Apple Inc. (or perhaps RFC 5724)
  - Example – sms:1-555-555-5555
- Webhook as per http://www.webhooks.org
  - Example — http://www.myserver.com/posthandler

The specific set of schemes supported for each operation shall be advertised in the server's API document.

If a list element value of the **responseHandler** parameter is the token "poll", the server shall, in its acknowledgement message, provide a hypermedia control that may be used to poll the server periodically to determine the execution status of the request.

If more than one instance of the "poll" token appears as a list element value of the **responseHandler** parameter, the extraneous instances of the token shall be ignored. The "poll" token need only appear once to trigger the inclusion of the status, progress and control elements within acknowledgement messages.

The following YAML fragment defines the **responseHandler** parameter.

```
name: responseHandler
in: query
description: A list of response handlers.
required: false
allowEmptyValue: false
schema:
  type: array
  items:
    type: string
    format: url
style: form
explode: false
```

Example: The following example fetches NHD flow lines asynchronously from the CubeWerx Testbed-14 WFS server (D113).

```
http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/framework/
     collections/NHDFLHI/items?
     count=100&
     f=application%2Fgeo%2Bjson&
     bbox=37.709077,-122.513476,37.839064,-122.351771&
     responseHandler=pvretano@pvretano.com
```

NOTE Examples in this clause are formatted to facilitate readability and to highlight the various query parameters.

A.5.3. Acknowledgement schema

This clause defines the XML-Schema and JSON-schema of the acknowledgement message that is used to signal that an asynchronous request has been successfully accepted. The same message schema is also used in response to a polling request to indicate the execution status of an asynchronous request.

The following JSON-Schema fragment define the JSON encoding of the acknowledgement message.

```
    "ackMessageType" : {
       "type" : "object",
       "properties" : {
          "links" : {
             "type" : "array",
             "items" : {
                "$ref" : "#/components/schemas/jsonLink"
             }
          },
          "executionStatus": {
             "type": "string",
             "enum": [
                "cancelled",
                "completed",
                "executing",
                "pending"
             ]
          },
          "precentCompleted": {
             "type": "integer"
          }
       }
    },
    "jsonLink" : {
       "type" : "object",
       "required" : [
          "href"
       ],
       "properties" : {
          "href" : {
             "type" : "string",
             "format" : "uri"
          },
          "rel" : {
             "type" : "string",
             "example" : "next"
          },
          "type" : {
             "type" : "string",
             "example" : "application/gml+xml;version=3.2"
          },
          "hreflang" : {
             "type" : "string",
             "example" : "el"
          },
          "title" : {
             "type" : "string",
             "example" : "Trierer Strasse 70, 53115 Bonn"
          }
       }
    }
```

The following XML-Schema fragment defines the ows:Acknowledgement element:

```
<xsd:element name="Acknowledgment"
             type="ows:Acknowledgement" id="Acknowledgement"/>
<xsd:complexType name="Acknowledgement" id="AcknowledgementType">
   <xsd:sequence>
      <xsd:element ref="atom:link" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="Status"
                   type="wfs:ExecutionStatusType" minOccurs="0"/>
      <xsd:element name="PercentCompleted"
                   type="xsd:nonNegativeInteger" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="ExecutionStatusType">
   <xsd:union>
      <xsd:simpleType>
         <xsd:restriction base="xsd:string">
            <xsd:enumeration value="cancelled"/>
            <xsd:enumeration value="completed"/>
            <xsd:enumeration value="executing"/>
            <xsd:enumeration value="pending"/>
         </xsd:restriction>
      </xsd:simpleType>
      <xsd:simpleType>
         <xsd:restriction base="xsd:string">
            <xsd:pattern value="other:\w{2,}"/>
         </xsd:restriction>
      </xsd:simpleType>
   </xsd:union>
</xsd:simpleType>
```

When an operation is invoked asynchronously, the server shall respond immediately with an acknowledgement message indicating that the server has successfully accepted the request or an OGC exception message indicating an error. If successfully accepted, the HTTP status code shall be set to "202 Accepted".

A.5.4. Asynchronous Processing class

For servers that implement the Asynchronous Processing conformance class, the acknowledgement message may contain an link, with rel="cancel", that may be used to cancel the asynchronously invoked operation.

The response to resolving the rel="cancel" link shall be an acknowledgement message that shall contains the wfs:Status element with its value set to "cancelled". The HTTP status code in this case shall be set to "200 OK".

NOTE	The "cancel" link may also be included in the response's HTTP header using the Link field (see RFC 5988 [https://tools.ietf.org/html/rfc5988]).

NOTE This document does not define a specific template, form or encoding of any link that appears in an acknowledgement message. Server implementations are free to encode the URI value of the href attribute of a link element in whatever way they deem suitable.

NOTE When resolving links, this should be done with the same credentials as the original asynchronous request.

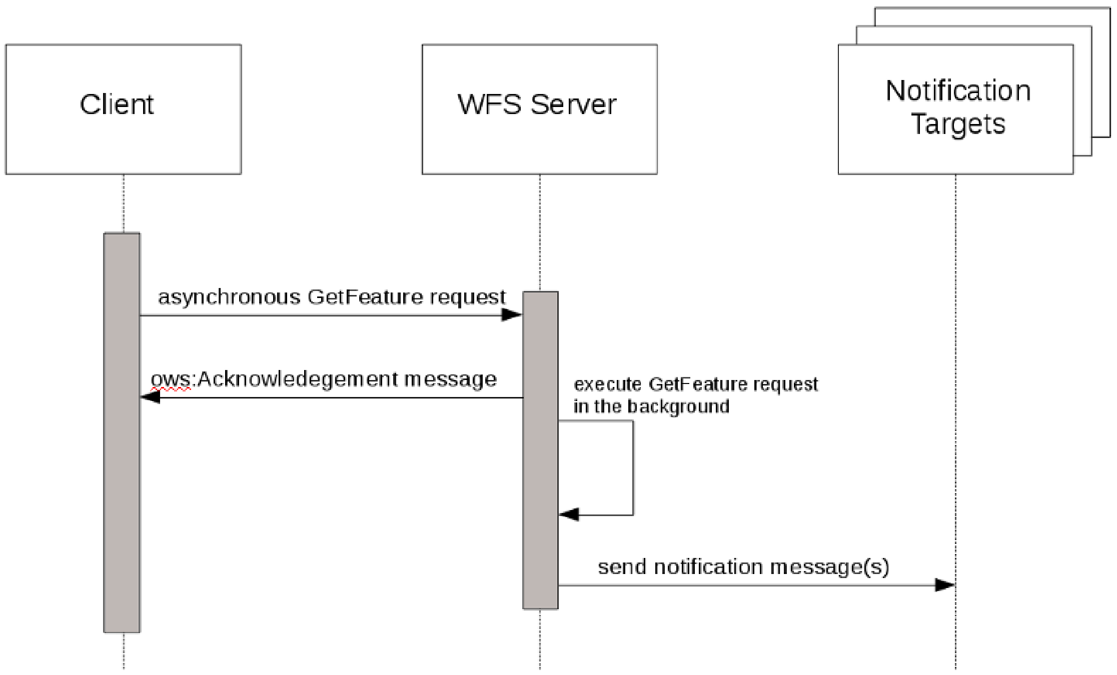

Client
WFS Server
Notification Targets
asynchronous GetFeature request
ows:Acknowledegement message
execute GetFeature request in the background
send notification message(s)

Figure 51. Sequence diagram for the Asynchronous Process Class

Example: Simple JSON acknowledgement with a hypermedia control to cancel the request.

```
{
   "links": [
      {"rel": "cancel",
       "href": "http://www.someserver.com/jobs/cancel/1013"}
   ]
}
```

A.5.5. Asynchronous Polling class

For servers that implement the Asynchronous Polling conformance class, the acknowledgement message shall include a link element, with rel="monitor" that may be periodically resolved to determine the execution status of an asynchronous request.

The response to resolving the rel="monitor" link shall be an acknowledgement message that shall contain the a status component indicating the execution status of the asynchronous request and may include a percent completed component with a percentage value indicating how much of the request has been completed. The HTTP status code in this case shall be set to "200 OK".

Requesting the execution status of an asynchronous request after its processing has been completed — and the operation's response is still available — shall result in an acknowledgement message that shall contain a status component with its value set to "completed" and shall also include a link component, with rel="http://www.opengis.net/def/rel/ogc/1.0/operationResponse", that provides a URI that may be used to retrieve the response.

Requesting the execution status of an asynchronous request after its processing has been completed — and the operation's response is no longer available (e.g. is has expired) — shall result in an OGC exception message and the HTTP status code shall be set to "404 Not Found".

NOTE The rel "http://www.opengis.net/def/rel/ogc/1.0/operationResponse" is an extension relation type (see RFC 5988 [https://tools.ietf.org/html/rfc5988], Section 4.2) and shall, in due course, be defined with OGC Naming Authority.

NOTE The "http://www.opengis.net/def/rel/ogc/1.0/operationResponse", "monitor" and "cancel" links may also be included in the response's HTTP header using the Link field (see RFC 5988 [https://tools.ietf.org/html/rfc5988]).

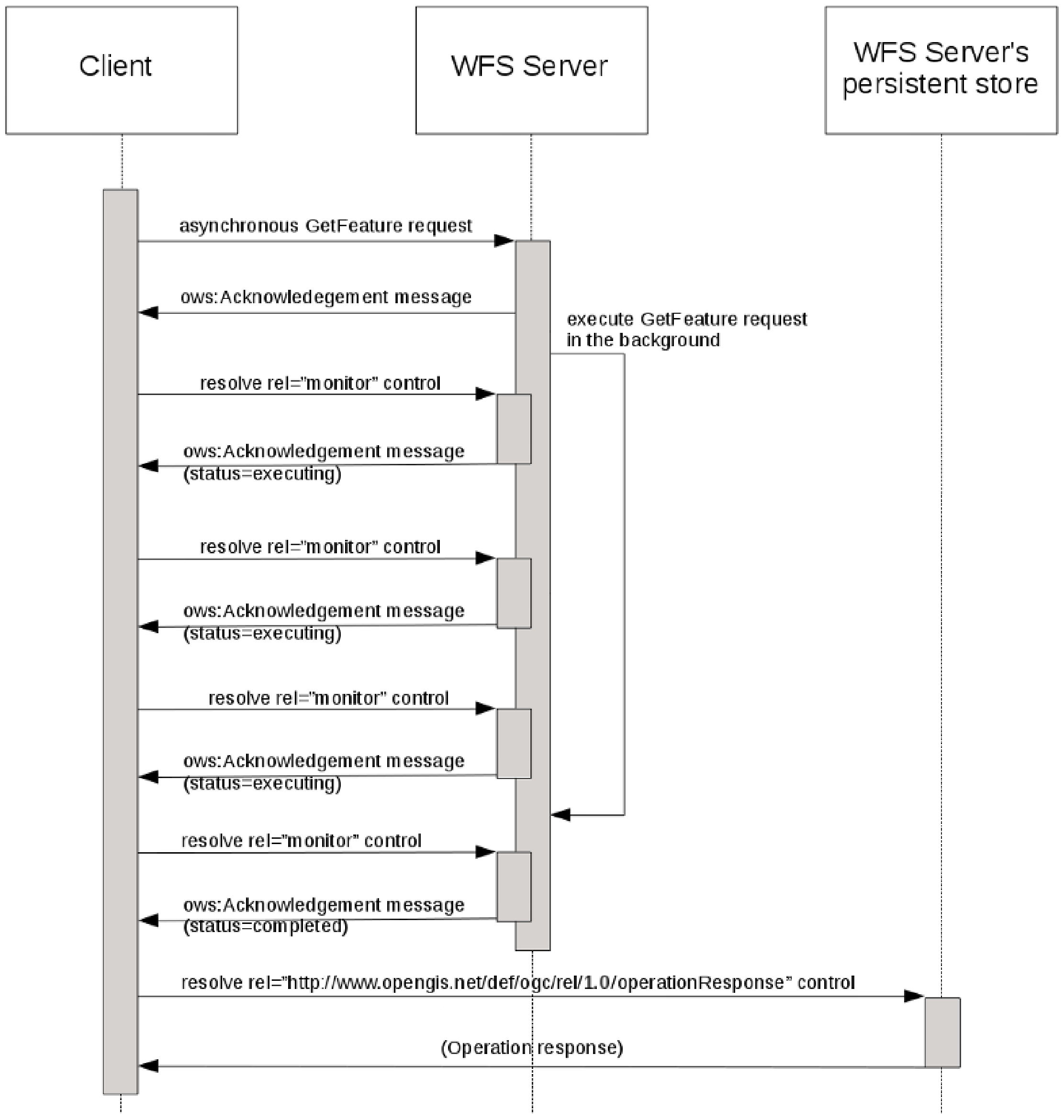

Client
WFS Server
WFS Server's persistent store
asynchronous GetFeature request
ows:Acknowledegement message
execute GetFeature request in the background
resolve rel="monitor" control
ows:Acknowledgement message (status=executing)
resolve rel="monitor" control
ows:Acknowledgement message (status=executing)
resolve rel="monitor" control
ows:Acknowledgement message (status=executing)
resolve rel="monitor" control
ows:Acknowledgement message (status=completed)
resolve rel="http://www.opengis.net/def/ogc/rel/1.0/operationResponse" control
(Operation response)

Figure 52. Sequence diagram for the Asynchronous Polling Class

A.5.6. Examples

EXAMPLE: The following example shows the response that a server that implements the Asynchronous Polling conformance class might initially generate in response to an asynchronously invoked operation. The acknowledgement message contains hypermedia controls to monitor the execution status of the request and to cancel the request. The execution status at this time is *pending* indicating that the request has been queued for execution.

```
{
   "links": [
      {"rel": "monitor",
       "href": "http://www.someserver.com/jobs/1013"},
      {"rel": "cancel",
       "href": "http://www.someserver.com/jobs/cancel/1013"},
   "status": "pending"
}
```

EXAMPLE: The following example shows a polling response some time after an operation was invoked asynchronously. The acknowledgement message contains hypermedia controls to monitor the execution status of the request and to cancel the request.

```
{
   "links": [
      {"rel": "monitor",
       "href": "http://www.someserver.com/jobs/1013"},
      {"rel": "cancel",
       "href": "http://www.someserver.com/jobs/cancel/1013"},
   "status": "executing",
   "percentCompleted": 47
}
```

EXAMPLE: This following example shows the polling response after request processing has been completed. Resolving the hypermedia control with rel="http://www.opengis.net/def/rel/ogc/1.0/operationResponse" will retrieve the request's response if it is still available; if the response is not available (e.g. it has expired from the cache) resolving the control would result in an OGC exception message and a "404 Not Found".

```
{
   "links": [
      {"rel": "http://www.opengis.net/def/rel/ogc/1.0/operationResponse",
       "href": "http://www.someserver.com/jobs/results/1013"}
   ],
   "status": "completed",
}
```

A.5.7. Notification message content

For servers that implement the Asynchronous Processing conformance class, an operation’s response shall be accessible via the notification message sent by the server using the specified response handler(s) (see responseHandler parameter) to signal that request processing has been completed.

In general the content of a notification message shall either be the operation’s complete response, or a reference to it, or an exception message.

The specific content of a notification message is not defined in this document because it is dependent on the scheme(s) specified as the value of the **responseHandler** parameter (see responseHandler parameter). For size-limited schemes, such as sms, a URL reference to the response would seem to be most appropriate since the entire response content is unlikely to fit into the message space. For other schemes, such as webhooks, the content of the notification message can be the complete response of the operation (e.g. the response to a GetFeature request). The following table contains informative recommendations for the content of notification messages based on the scheme being used:

Table 1. Recommended notification content based on scheme

Notification scheme	Recommended content (good response)	Recommended content (exception)
mailto:	An email message containing a URL for retrieving the operation’s response.	An email message containing a narrative that describes the exception; an optional attachment with the server’s actual OGC exception message may also be included
sms:	A URL for retrieving the operation’s response; tiny URLs may be used if the retrieval URL is particularly long	A URL for retrieving the server’s OGC exception message; tiny URLs may be used if the retrieval URL is particularly long
http: (webhook)	The operation’s complete response	The complete OGC exception message

A.6. Hierarchical path extension (i.e. theme extension)

A.6.1. Introduction

The theme extension defines a mechanism that allows WFS providers to organize the collections their WFS offers into one or more hierarchies or themes. A theme is a hierarchy of connected nodes where intermediate nodes can be references to other themes or collections and terminal (leaf) nodes are references to collections.

A service can offer any number of themes, and themes or collections can be members of zero or more themes.

NOTE	The same capability could also be achieved under the '/collections' path by allowing additional path elements between the '/collections' element and the '{collectionId}' element. However, as per issue https://github.com/opengeospatial/WFS_FES/issues/90, it was decided (for Testbed-14) to preserve the simplicity of the '/collections' path and define an external mechanism for hierarchies under the '/themes' path.

A.6.2. Operations

Requirement 1	/req/extensions/themes/themes-root The server SHALL support the HTTP GET operation at the path `/themes`.

Requirement 2	/req/extensions/themes/themes-node The server SHALL support the HTTP GET operation on a path that starts with the '/themes' path element, is followed by zero or more '/{themeId}' path elements and is terminated by the '/items' path element or the pseudo-terminal '/{collectionId}' path elements. If the terminal path element is the '/collectionId' path element, it may be followed by the other elements defined in the WFS 3.0 Core specification (i.e. the '/items' path element or the '/items/{fid}' path element.

A.6.3. Response

Requirement 3	/req/extension/themes/themes-description A successful execution of the operation SHALL be reported as a response with a HTTP status code `200`. The content of the response is defined in table X.

Path	Response
/themes	A "themes-root" document as per the schema below.
/themes/{themeId}[/{themeId}...]	A "theme-node" document as per the schema below.
/themes[/{themeId}...]/items	Same behaviour as /{collectionId}/items except that the collection is the union of all terminal leaf {collectionId} nodes.
/themes[/{themeId}...]/{collectionId}	Same behaviour as /{collectionId}

/themes[/{themeId}…]/{collectionId}/items	Same behaviour a /{collectionId}/items

```
    "themes-root": {
       "type": "array",
       "items": {
          "$ref": "#/components/schemas/theme-node"
       }
    }

    "theme-node": {
       "type": "object",
       "required": ["name","themes","collections"],
       "properties": {
          "name": {
             "type": "string"
          },
          "title": {
             "type": "string"
          },
          "description": {
             "type": "string"
          },
          "themes": {
             "type": "array",
             "items": {
                "$ref": "#/components/schemas/theme-node"
             }
          },
          "collections": {
             "type": "array",
             "items": {
                "$ref": "#/components/schemas/collectionInfo"
             },
          }
       }
    }

    "collectionInfo": {
       "type": "object",
       "required": ["name","title","links"],
       "properties": {
          "name": {
             "type": "string"
          },
          "title": {
             "type": "string"
          },
          "description": {
             "type": "string"
```

```
        },
        "links": {
           "type": "array",
           "items": {
              "$ref": "#/components/schemas/link
           }
        },
        "extent": {
           "$ref": "#/components/schemas/bbox"
        },
        "crs": {
           "type": "array",
           "default": "http://www.opengis.net/def/crs/OGC/1.3/CRS84"
           "items": {
              "type": "string",
              "format": "uri"
           }
        }
     }
  }
```

A.6.4. Parameters

For paths that terminate with a '/{collectionId}' or '/items' path element, the parameters defined in the WFS 3.0 Core specification may be used.

For paths that terminate with '/{collectionId}/items', the behaviour shall be the same as that defined for the '/collections/{collectionId}/items' path in the WFS 3.0 Core.

For paths that terminate with '/{themeId}/items', the behaviour shall be the same as accessing a collection of features composed of the union of all {collectionId} leaf nodes for the sub-tree anchored by the specified {themeId} node.

NOTE If parameter filtering [https://rawgit.com/opengeospatial/WFS_FES/master/docs/17-069.html#_parameters_for_filtering_on_feature_properties] is used it is likely that not all filters will apply for all collections aggregated under a theme. In this case, the filter shall be applied to those collections that contain the specified parameter and ignored otherwise.

NOTE Say there is a theme X that aggregates collection A, B and C. Collection A has properties P1,P2 and collections B and C have properties P1,P3. In this example, for the following operation '/X/items?P1=value1&P3=value2', the 'P1=value1' filter shall be applied to collections A, B and C and the 'P3=value2' filter shall additionally be applied to collections B and C. In other words, it would be same as executing the following three queries:

```
/A?P1=value1
/B?P1=value1&P3=value2
/C?P1=value1&P3=value2
```

and then union-ing the results into a single response document.

EXAMPLE:

The following example was generated from the Testbed-14 Cubewerx server using the following URL: http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/foundation/themes?f=json

```
{
  "themes": [
    {
      "name": "BOUNDARIES",
      "title": "Boundaries",
      "themes": [
         {
           "name": "...",
           "title": "...",
           "collection":
         },

      ],
      "collections": [
        {
          "name": "barrierl_1m",
          "title": "Barrier Line Features",
          "links": [
            {
              "href":
"http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/foundation/collections/ba
rrierl_1m?f=application%2Fjson",
              "rel": "collection",
              "type": "application/json",
              "title": "Metadata about the barrierl_1m feature collection as JSON."
            },
            {
              "href":
"http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/foundation/collections/ba
rrierl_1m?f=text%2Fxml",
              "rel": "collection",
              "type": "text/xml",
              "title": "Metadata about the barrierl_1m feature collection as XML."
            },
            {
              "href":
"http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/foundation/collections/ba
rrierl_1m?f=text%2Fhtml",
              "rel": "collection",
              "type": "text/html",
              "title": "Metadata about the barrierl_1m feature collection as HTML."
            }
          ]
        },
        {
          "name": "coastl_1m",
          "title": "Coastlines",
          "description": "Coastline/shorelines (BA010) have been portrayed for coastal
islands but not inland islands.",
          "links": [
            {
              "href":
"http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/foundation/collections/co
astl_1m?f=application%2Fjson",
              "rel": "collection",
              "type": "application/json",
              "title": "Metadata about the coastl_1m feature collection as JSON."
            },
            {
              "href":
"http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/foundation/collections/co
astl_1m?f=text%2Fxml",
              "rel": "collection",
              "type": "text/xml",
              "title": "Metadata about the coastl_1m feature collection as XML."
            },
            {
              "href":
"http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/foundation/collections/co
astl_1m?f=text%2Fhtml",
              "rel": "collection",
              "type": "text/html",
              "title": "Metadata about the coastl_1m feature collection as HTML."
            }
          ]
        }
      ]
    },
    {
      "name": "HYDROGRAPHY",
      "title": "Hydrography",
      "collections": [
        {
          "name": "inwatera_1m",
          "title": "Inland Water Areas",
          "links": [
            {
              "href":
"http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/foundation/collections/in
```

```
watera_1m?f=application%2Fjson",
              "rel": "collection",
              "type": "application/json",
              "title": "Metadata about the inwatera_1m feature collection as JSON."
            },
            {
              "href":
"http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/foundation/collections/in
watera_1m?f=text%2Fxml",
              "rel": "collection",
              "type": "text/xml",
              "title": "Metadata about the inwatera_1m feature collection as XML."
            },
            {
              "href":
"http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/foundation/collections/in
watera_1m?f=text%2Fhtml",
              "rel": "collection",
              "type": "text/html",
              "title": "Metadata about the inwatera_1m feature collection as HTML."
            }
          ]
        },
        {
          "name": "miscl_hydro_1m",
          "title": "Miscellaneous Line Features",
          "links": [
            {
              "href":
"http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/foundation/collections/mi
scl_hydro_1m?f=application%2Fjson",
              "rel": "collection",
              "type": "application/json",
              "title": "Metadata about the miscl_hydro_1m feature collection as JSON."
            },
            {
              "href":
"http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/foundation/collections/mi
scl_hydro_1m?f=text%2Fxml",
              "rel": "collection",
              "type": "text/xml",
              "title": "Metadata about the miscl_hydro_1m feature collection as XML."
            },
            {
              "href":
"http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/foundation/collections/mi
scl_hydro_1m?f=text%2Fhtml",
              "rel": "collection",
              "type": "text/html",
              "title": "Metadata about the miscl_hydro_1m feature collection as HTML."
            }
          ]
        },
      ]
    },
    {
      "name": "POPULATION",
      "title": "Population",
      "collections": [
        {
          "name": "builtupa_1m",
          "title": "Built-Up Areas",
          "links": [
            {
              "href":
"http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/foundation/collections/bu
iltupa_1m?f=application%2Fjson",
              "rel": "collection",
              "type": "application/json",
              "title": "Metadata about the builtupa_1m feature collection as JSON."
            },
            {
              "href":
"http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/foundation/collections/bu
iltupa_1m?f=text%2Fxml",
              "rel": "collection",
              "type": "text/xml",
              "title": "Metadata about the builtupa_1m feature collection as XML."
            },
            {
              "href":
"http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/foundation/collections/bu
iltupa_1m?f=text%2Fhtml",
              "rel": "collection",
              "type": "text/html",
              "title": "Metadata about the builtupa_1m feature collection as HTML."
            }
          ]
        },
        {
          "name": "builtupp_1m",
          "title": "Built-Up Area Points",
          "links": [
            {
              "href":
"http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/foundation/collections/bu
iltupp_1m?f=application%2Fjson",
              "rel": "collection",
              "type": "application/json",
              "title": "Metadata about the builtupp_1m feature collection as JSON."
            },
            {
              "href":
"http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/foundation/collections/bu
iltupp_1m?f=text%2Fxml",
              "rel": "collection",
              "type": "text/xml",
              "title": "Metadata about the builtupp_1m feature collection as XML."
            },
            {
              "href":
"http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/foundation/collections/bu
iltupp_1m?f=text%2Fhtml",
              "rel": "collection",
              "type": "text/html",
              "title": "Metadata about the builtupp_1m feature collection as HTML."
            }
          ]
        },
      ]
    },
    {
      "name": "TRANSPORTATION",
      "title": "Transportation",
      "collections": [
        {
          "name": "aerofacp_1m",
          "title": "Airport Facilities Points",
          "links": [
            {
              "href":
"http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/foundation/collections/ae
rofacp_1m?f=application%2Fjson",
              "rel": "collection",
              "type": "application/json",
              "title": "Metadata about the aerofacp_1m feature collection as JSON."
            },
            {
              "href":
"http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/foundation/collections/ae
rofacp_1m?f=text%2Fxml",
              "rel": "collection",
              "type": "text/xml",
              "title": "Metadata about the aerofacp_1m feature collection as XML."
            },
            {
              "href":
"http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/foundation/collections/ae
rofacp_1m?f=text%2Fhtml",
              "rel": "collection",
              "type": "text/html",
              "title": "Metadata about the aerofacp_1m feature collection as HTML."
            }
          ]
        },
        {
          "name": "roadl_1m",
          "title": "Roads",
          "links": [
            {
              "href":
"http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/foundation/collections/ro
adl_1m?f=application%2Fjson",
              "rel": "collection",
              "type": "application/json",
              "title": "Metadata about the roadl_1m feature collection as JSON."
            },
            {
              "href":
"http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/foundation/collections/ro
adl_1m?f=text%2Fxml",
              "rel": "collection",
              "type": "text/xml",
              "title": "Metadata about the roadl_1m feature collection as XML."
            },
            {
              "href":
"http://www.pvretano.com/cubewerx/cubeserv/default/wfs/3.0.0/foundation/collections/ro
adl_1m?f=text%2Fhtml",
              "rel": "collection",
              "type": "text/html",
              "title": "Metadata about the roadl_1m feature collection as HTML."
            }
          ]
        },
      ]
    }
  ]
}
```

A.7. Map extension

A.7.1. Introduction

This clause defines the behavior of a WFS that supports the diagrammatic representation of feature data in the form of a digital image or map. In this sense, this extension is similar to a Web Map Server (WMS). However, this extension differs from a WMS in that the contents of the map, with respect to the underlying feature data, can be dynamically specified at request time by the requester. In other words, filter expressions using the filtering capabilities defined in WFS 3.0 Core or using any other query extension may be used to define the set of features that shall be rendered into the map response.

This clause describes:

- an access path for the 'getMap' operation
- a set of query parameters that are specific to the 'getMap' operation
- a set of HTTP response headers to encode metadata about the response

A.7.2. Operation

For every feature collection identified in the metadata about the feature collection (path '/'), the server SHALL support the HTTP GET operation at the path '/collections/{name}/map'.

The parameter 'name' is each property of the same name in the feature collection metadata (JSONPath: '$.collections[*].name').

A.7.3. Parameter style

Each getMap operation SHALL support a parameter 'style' with the following characteristics (using an OpenAPI Specification 3.0 fragment):

```
name: style
in: query
required: false
schema:
  type: string
  description: the named style to use to render the features into the map
  enum:
    - {list of valid style names}
  default:
    - {the name of the default style}
style: form
explode: true
```

The optional 'style' parameter specifies the style in which the set of features in the response SHALL be rendered into the map.

The value of the 'style' parameter SHALL be from the set of style names advertised in the service description document of the server.

If the 'style' parameter is not specified, the set of feature in the response SHALL be rendered into the map using the style listed as the default style in the service description document of the server.

A.7.4. Parameter width,height

Each getMap operation SHALL support a 'width' and 'height' parameter with the following characteristics (using an OpenAPI Specification 3.0 fragment):

```
name: width
in: query
description: Width of output map in pixels.
required: false
allowEmptyValue: true
schema:
  type: integer
  minimum: {min width supported by the server}
  maximum: {max width supported by the server}
  default: {default width if the parameter is not specified}
style: form
explode: true

name: height
in: query
description: Height of output map in pixels.
required: false
allowEmptyValue: true
schema:
  type: integer
  minimum: {min height supported by the server}
  maximum: {max height supported by the server}
  default: {default height if the parameter is not specified}
style: form
explode: true
```

If the optional 'width' and/or 'height' parameters are specified then the server SHALL generate a map of the specified pixel width and height.

If either or both parameters are not specified on a getMap operation then the default values advertised in the service description document of the server SHALL be used.

The optional 'width' and 'height' parameters specify the size in integer pixels of the map to be produced.

If the request is for a picture format, the returned picture, regardless of its MIME type, SHALL have exactly the specified width and height in pixels.

In the case where the aspect ratio of the 'bbox' parameter [https://rawgit.com/opengeospatial/WFS_FES/master/docs/17-069.html#_parameter_bbox] and the ratio of the 'width'/'height' parameters are different, the server shall stretch the returned map so that the resulting pixels could themselves be rendered in the aspect ratio of the 'bbox' parameter. In other words, it SHALL be possible using this definition to request a map for a device whose output pixels are themselves non-square, or to stretch a map into an image area of a different aspect ratio.

Map distortions will be introduced if the aspect ratio 'width'/'height' is not commensurate with X, Y and the pixel aspect. Client developers should minimize the possibility that users will inadvertently request or unknowingly receive distorted maps.

If a request is for a graphic element format that does not have explicit width and height, the client shall include the 'width' and 'height' values in the request and a server may use them as helpful information in constructing the output map.

The optional <MaxWidth> and <MaxHeight> elements in the service metadata are integers indicating the maximum width and height values that a client is permitted to include in a single GetMap request. If either element is absent, the server imposes no limit on the corresponding parameter.

A.7.5. Parameter bgcolor

Each getMap operation SHALL support a 'bgcolor' parameter with the following characteristics (using an OpenAPI Specification 3.0 fragment):

```yaml
name: bgcolor
in: query
description: Background colour of map.
required: false
allowEmptyValue: true
schema:
  type: string
  default: {default background map color}
  example: 0xFFFFFF
style: form
explode: true
```

The optional 'bgcolor' parameter is a string that specifies the color to be used as the background (non-data) pixels of the map. The general format of 'bgcolor' is a hexadecimal encoding of an RGB value where two hexadecimal characters are used for each of red, green, and blue color values. The values can range between 00 and FF (0 and 255, base 10) for each. The format is 0xRRGGBB; either upper or lower case characters are allowed for RR, GG, and BB values. The “0x” prefix shall have a lower case “x”.

The default value of the 'bgcolor' parameter shall be specified in the service description document of the server.

When FORMAT is a picture format, a WFS shall set the background pixels to the color specified by 'bgcolor'.

When FORMAT is a graphic element format (which does not have an explicit background), or a picture format, a WFS should avoid use of the 'bgcolor' value for foreground elements because they would not be visible against a background picture of the same color.

When the value of the 'transparent' parameter is set to FALSE, non-data pixels shall be set to the value of 'bgcolor' parameter.

A.7.6. Parameter transparent

Each getMap operation SHALL support a 'transparent' parameter with the following characteristics (using an OpenAPI Specification 3.0 fragment):

```yaml
name: transparent
in: query
description: Flag indicating whether the map should be transparent or not.
required: false
allowEmptyValue: true
schema:
  type: boolean
  default: false
style: form
explode: true
```

The optional 'transparent' parameter specifies whether the map background is to be made transparent or not. The 'transparent' parameter can take on two values, "true" or "false".

The default value SHALL be FALSE if this parameter is absent from a getMap request.

The ability to return pictures drawn with transparent pixels allows results of different Map requests to be overlaid, producing a composite map.

It is strongly recommended that every WFS offer a format that provides transparency for maps that could sensibly be overlaid above others.

NOTE The image/gif format provides transparency and is properly displayed by common web clients. The image/png format provides a range of transparency options but support in viewing applications is less common. The image/jpeg format does not provide transparency at all.

When the value of the 'transparent' parameter is set to TRUE and the requested format is a picture format (e.g. image./gif), then the server SHALL return (when permitted by the requested format) a result where all of the pixels not representing feature data in the map are set to a transparent value.

When the value of the 'transparent' parameter is set to FALSE, non-data pixels shall be set to the value of `bgcolor`.

A.7.7. Response

A successful execution of the operation SHALL be reported as a response with a HTTP status code '200'.

The response SHALL only render features selected by the request into the map.

The number of features rendered into the map depends on the server and the parameter `limit`.

Unlike the case for text-based responses, the server is not required to include any additional hypermedia controls in the response.

If, however, a server wishes to do so, those hypermedia controls SHALL be specified using the HTTP response header 'Link' [https://www.w3.org/wiki/LinkHeader].

If hypermedia controls are provided in the header, they SHALL include the 'rel' and 'type' link parameters.

Additional response metadata such as a timestamp [https://rawgit.com/opengeospatial/WFS_FES/master/docs/17-069.html#_response_6], number of matched features [https://rawgit.com/opengeospatial/WFS_FES/master/docs/17-069.html#_response_6] and number of returned features [https://rawgit.com/opengeospatial/WFS_FES/master/docs/17-069.html#_response_6] may be included in the response as HTTP headers.

If timestamp information is included in the response, it SHALL be encoded using the HTTP header 'Date'.

If the number of matched features is included in the response, it SHALL be encoded using a HTTP header defined by this standard named 'OGC-NUMBER-MATCHED'.

If the number of returned features (i.e. the number of features rendered into the map) is included in the response, it SHALL be encoded using a HTTP header defined by this standard named 'OGC-NUMBER-RETURNED'.

NOTE The WFS core standard does not define HTTP headers for the 'numberMatched' and 'numberReturned' cases since their usefulness for text-based, and perhaps streamed, response formats is limited given that this information is usually known at the end of request. However, in the map case, such headers make sense and are thus defined by this standard.

A.8. Tile extension

The concurrently running Vector Tiles Pilot generated some details about a tile extensions for WFS. This is actually less an extension of WFS as much as a test to see if the architectural pattern established by WFS can be applied to objects other than features. However, the information is still relevant to this ER and so the reference is listed here.

See: https://portal.opengeospatial.org/files/?artifact_id=81698&version=1

A.9. OpenSearch query extension

The OpenSearch query extension defines a set of query parameters that can be mapped to OpenSearch [https://github.com/dewitt/opensearch/blob/master/opensearch-1-1-draft-6.md] and/or OpenSearch Geo [https://portal.opengeospatial.org/files/?artifact_id=56866] parameters.

NOTE The acronym "f.i.t." stands for "fixed in template". That is that the value is fixed in an OpenSearch query template.

The following query parameters may be used on the following paths:

- /collections/{collectionId}/items

to define a query predicate used to select a sub-set of features to be presented in a response document.

Table 2. OpenSearch query parameters

Query class	Query subclass	URL component	OpenSearch parameter(s)	Description	Datatype and Value
Text search	N/A	q	searchTerms	A space-separated list of search terms are used to search all text fields in a feature	Character string
Identity search	N/A	fids	uid	A comma-separated list of feature identifiers to retrieve	Character string

Spatial search	Bounding box search	bbox	box	The bounding box to be used as a spatial predicate	Array of number (lower,upper)
		bbox_crs	n/a	Asserts the crs of the **bbox** parameter	AnyURI (default=http://www.opengis.net/def/crs/OGC/1.3/CRS84)
	Geometry search	geometry	geometry	The geometry to be used as a spatial predicate	WKT String (see OGC 06-103r4)
		geometry_crs	N/A	Asserts the crs of the **geometry** parameter	AnyURI (default=http://www.opengis.net/def/crs/OGC/1.3/CRS84)
		relation	relation	The spatial operator to apply when comparing with the **geometry** parameter	Character string. One of: equals, disjoint, touches, within, overlaps, crosses, intersects, contains, dwithin, beyond (default=intersects)
		distance	N/A	Distance value for dwithin or beyond spatial operators	Number
		distance_uom	N/A	The units of measure used to express the value of the **distance** parameter	Number
	Proximity search	lat	lat	Latitude expressed in WGS84	Number
		lon	lon	Longitude expressed in WGS84	Number
		radius	radius	Search radius	Number
		radius_uom	N/A	The units of measure used to express the value of the **radius** parameter	Character string (see http://unitsofmeasure.org/ucum.html) (default=m)

Temporal search		time	start/end	A time instance or time period	Character string (see http://www.w3.org/TR/NOTE-datetime)
		time_rs	N/A	The temporal reference system for the **time** parameter	AnyURI (default=http://www.opengis.net/def/uom/ISO-8601/0/Gregorian)
		trelation	N/A	The temporal operator to apply using the value of the **time** parameter	Character string. One of:after, before, begins, begunBy,tContains, during, endedBy, ends, tEquals, meets, metBy, tOverlaps, overlappedBy, anyIntersects (default=anyIntersects)
Complex predicate	n/a	filter	n/a	A text fragment encoding a query predicate is some query language	Character string
		filter_language	n/a	Indicates the predicate language used to encode the filter expression that is the value of the **filter** parameter.	Character string (default=urn:ogc:def:queryLanguage:OGC-FES:Filter)

Notes:

1. The **geometry_crs**, **relation**, **distance** and **distance_uom** parameters are all modifiers of the **geometry** parameter and should only be specified if the **geometry** parameter is specified.
2. If specified, the parameters **lat**, **lon** and **radius** must all be specified together.
3. The **trelation** parameter is a modifier of the **time** parameter and should only be specified if the **time** parameter is specified.
4. The **filter** and **filter_language** parameters are mutually exclusive with the text search, identify search, spatial search, proximity search and temporal search parameters.
5. The implied logical operation for the search parameter is **AND**.
6. The default trelation value for a single time instant is **after**.
7. The default trelation value for a time period is **AnyInteracts**.

Basic text searching using the **q** parameter is expected to be processed as follows:

1. Match against the full text of all the character-valued properties of an entry
2. Matching is case insensitive
3. In the case of multiple search terms, if any of the property values being tested, as per (a), contains at least one of the specified search terms then that entry shall appear in the result set

A.10. Advanced adhoc query extension

The WFS 3.0 Core specification supports what can be classified as simple adhoc queries. That is, the ability to dynamically define a subset of features from a single collection using a small and simple subset of query predicates (i.e. bbox, time and feature property filters). The advanced adhoc query extension adds the ability to dynamically define a subset of features from one or more collections using a combination of simple and/or complex predicates that can include logically connected scalar, spatial and temporal predicates.

The root of all adhoc queries is the `/query` resource.

Individual adhoc queries have the path `/query/{queryId}` where the query identifier is assigned by the server when an adhoc query is created.

The following table summarizes the actions that may be performed on the `/query` and `/query/{queryId}` resources.

Table 3. Actions for query resources

Resource	HTTP Method	Description
all	OPTIONS	Get the list of supported representations and methods for the resource
/query	GET	Not specified
	PUT	Not specified
	POST	Creates a new adhoc query. The body of the request contains the query expressed using one of the representations supported by the server.
	DELETE	Not specified.
/query/{queryId}	GET	Executed the adhoc query and return the query response; or a 410 if the query has expired.
	POST	Not specified.
	PUT	Not specified.
	DELETE	Removes the adhoc query; not strictly necessary since the server automatically manages the adhoc query.

A.10.1. Examples

EXAMPLE: Does the server support complex queries?

```
CLIENT                                                     SERVER
  |                                                           |
  |   OPTIONS /cubewerx/cubeserv/default/wfs/3.0.0/query HTTP/1.1
  |   Host: www.pvretano.com                                  |
  |---------------------------------------------------------->|
  |                                                           |
  |   HTTP/1.1 200 OK                                         |
  |   Allow: OPTIONS, POST                                    |
  |   Accept: text/xml                                        |
  |<----------------------------------------------------------|
```

Since the server did not return an exception indicating that the `/query` path is invalid and because it also lists the POST method as a valid method, this server supports adhoc complex queries.

EXAMPLE: Create a query. In this case, the query uses a spatial join to find all lakes within Algonquin park.

NOTE The participants tried to come up with a complex query example using the Daraa data but most of the property values in that data set are empty, resulting in empty responses for most queries that were tried. So, the participants returned to a WFS 2.0 example since the idea with this extension is to illustrate the ability to express a complex query.

```
CLIENT                                                     SERVER
  |                                                           |
  |   POST /.../query HTTP/1.1                                |
  |   Host: www.pvretano.com                                  |
  |   Content-Type: text/xml                                  |
  |                                                           |
  |   <?xml version="1.0"?>                                   |
  |   <GetFeature                                             |
  |      service="WFS"                                        |
  |      version="2.0"                                        |
  |      xmlns="http://www.opengis.net/wfs/2.0"               |
  |      xmlns:cw="http://www.cubewerx.com/cw"                |
  |      xmlns:fes="http://www.opengis.net/fes/2.0"           |
  |      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"|
  |      xsi:schemaLocation="http://www.opengis.net/wfs/2.0 ../../wfs.xsd">
  |      <Query typeNames="cw:Parks cw:Lakes">            |
  |         <fes:Filter>                                      |
  |            <fes:And>                                      |
  |               <fes:PropertyIsEqualTo>                     |
  |                  <fes:ValueReference>/cw:Parks</fes:ValueReference>
  |                  <fes:Literal>Algonquin Park</fes:Literal>|
  |               </fes:PropertyIsEqualTo>                    |
  |               <fes:Contains>                              |
  |                  <fes:ValueReference>/cw:Parks/geometry</fes:ValueReference>
  |                  <fes:ValueReference>/cw:Lakes/geometry</fes:ValueReference>
  |               </fes:Contains>                             |
  |            </fes:And>                                     |
  |         </fes:Filter>                                     |
  |      </Query>                                             |
  |   </GetFeature>                                           |
  |---------------------------------------------------------->|
  |                                                           |
  |   HTTP/1.1 201 Created                                    |
  |   Location: /.../query/7f26634c                           |
  |<----------------------------------------------------------|
```

The response indicates that the server has created the query and assigned it the identifier `7f26634c`.

NOTE Creating a query does not mean that the result set of the query is also created. The result set of the query is generated at run time or when the query is actually executed using the GET method.

EXAMPLE: What query response representations does the server provide for this query.

```
CLIENT                                                     SERVER
  |                                                           |
  |   OPTIONS /.../query/7f26634c HTTP/1.1                    |
  |   Host: www.pvretano.com                                  |
  |---------------------------------------------------------->|
  |                                                           |
  |   HTTP/1.1 200 OK                                         |
  |   Allow: OPTIONS, GET, DELETE                             |
  |   Accept: application/gml+xml; version=3.2,               |
  |           application/geo+json, text/html                 |
  |<----------------------------------------------------------|
```

This server support HTML, GeoJSON and GML output formats.

EXAMPLE: Execute the query requesting GML.

NOTE The engineering report used an XML/GML response example here because the WFS 2.0 specification defines the necessary response containers to express join tuples which GeoJSON does not support.

```
CLIENT                                                     SERVER
  |                                                           |
  |   GET /.../query/7f26634c HTTP/1.1                        |
  |   Host: www.pvretano.com                                  |
  |   Accept: application/gml+xml; version=3.2                |
  |---------------------------------------------------------->|
  |                                                           |
  |   HTTP/1.1 200 OK                                         |
  |   Content-Type: application/gml+xml; version=3.2          |
  |                                                           |
  |   <?xml version="1.0" encoding="UTF-8"?>
  |   <wfs:FeatureCollection
  |      timeStamp="2008-08-15T11:36:00" numberMatched="12"
  |      numberReturned="12" xmlns="http://www.someserver.com/cw"
  |      xmlns:wfs="http://www.opengis.net/wfs/2.0"
  |      xmlns:gml="http://www.opengis.net/gml/3.2"
  |      xmlns:xlink="http://www.w3.org/1999/xlink"
  |      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  |      xsi:schemaLocation="http://www.someserver.com/cw
  |                          ./SampleSchema.xsd
  |                          http://www.opengis.net/wfs/2.0
  |                          ../../wfs.xsd
  |                          http://www.opengis.net/gml/3.2
  |                          http://schemas.opengis.net/gml/3.2.1/gml.xsd">
  |      <wfs:member>
  |         <wfs:Tuple>
  |            <wfs:member>
  |               <Parks gml:id="Parks.287796">
  |                  <Name>Algonquin Park</Name>
```

```
                  <Boundary>
                     <gml:Polygon gml:id="GID_13"
                        srsName="http://www.opengis.net/def/crs/epsg/0/4326">
                        <gml:exterior>
                           <gml:LinearRing>
                              <gml:posList>...</gml:posList>
                           </gml:LinearRing>
                        </gml:exterior>
                     </gml:Polygon>
                  </Boundary>
               </Parks>
            </wfs:member>
            <wfs:member>
               <Lakes gml:id="Lakes.287797">
                  <Name>Canisbay Lake</Name>
                  <Boundary>
                     <gml:Polygon gml:id="GID_14"
                        srsName="http://www.opengis.net/def/crs/epsg/0/4326">
                        <gml:exterior>
                           <gml:LinearRing>
                              <gml:posList>...</gml:posList>
                           </gml:LinearRing>
                        </gml:exterior>
                     </gml:Polygon>
                  </Boundary>
               </Lakes>
            </wfs:member>
         </wfs:Tuple>
      </wfs:member>
      <wfs:member>
         <wfs:Tuple>
            <wfs:member xlink:href="#Parks.287796"/>
            <wfs:member>
               <Lakes gml:id="Lakes.287798">
                  <Name>Kearney Lake</Name>
                  <Boundary>
                     <gml:Polygon gml:id="GID_15"
                        srsName="http://www.opengis.net/def/crs/epsg/0/4326">
                        <gml:exterior>
                           <gml:LinearRing>
                              <gml:posList>...</gml:posList>
                           </gml:LinearRing>
                        </gml:exterior>
                     </gml:Polygon>
                  </Boundary>
               </Lakes>
            </wfs:member>
         </wfs:Tuple>
      </wfs:member>
      <wfs:member>
         <wfs:Tuple>
```

```
    |            <wfs:member xlink:href="#Parks.287796"/>
    |            <wfs:member>
    |               <Lakes gml:id="Lakes.287799">
    |                  <Name>Lake Of Two Rivers</Name>
    |                  <Boundary>
    |                     <gml:Polygon gml:id="GML_16"
    |                        srsName="http://www.opengis.net/def/crs/epsg/0/4326">
    |                        <gml:exterior>
    |                           <gml:LinearRing>
    |                              <gml:posList>...</gml:posList>
    |                           </gml:LinearRing>
    |                        </gml:exterior>
    |                     </gml:Polygon>
    |                  </Boundary>
    |               </Lakes>
    |            </wfs:member>
    |         </wfs:Tuple>
    |      </wfs:member>
    |      .
    |      .
    |      .
    |   </wfs:FeatureCollection>
    |<-----------------------------------------------------------|
```

NOTE The result has been truncated in the interests of space!

A.11. Transaction extension

The following extension defines the ability to insert, update and delete features from collections.

The REST architectural style is resource oriented and most naturally deals with individual resources. For this reason this extension categorizes transactions as `simple` and `complex`.

A simple transaction is a transaction that acts on a single feature in a collection. That is adding a new single feature to a collection, modifying an existing feature in a collection or delete a feature from a collection.

A complex transaction is a transaction that acts upon multiple features, perhaps spread across multiple collections and those actions have to all all succeed or the entire set of actions is rolled back leaving the affected collection(s) in a consistent state.

A.11.1. Simple transactions

The following table shows resources for simple transactions.

Table 4. Resources for simple transactions

Access path	Description

/collections/{collectionId}/items	The URL of a feature collection.
/collections/{collectionId}/items/{fid}	The URL of a feature.
/collections/{collectionId}/items/properties/{propertyName}	The URL of a property that is common to all features in the collection.
/collections/{collectionId}/items/{fid}/properties/{propertyName}	The URL of a property that belongs to a specific feature.

Representations

The canonical representation of features in a query response shall be features encoded using GML 3.2 (see OGC 07-036/ISO19136:2007). The MIME type for this representation is application/gml+xml; version=3.2.

The canonical representation of values in a property value query shall be:

- Plain text for scalar values (i.e. MIME type plain/text)
- XML for composite or complex values such as an address (i.e. MIME type text/xml)
- GML 3.2 for geometric values (i.e. MIME type application/gml+xml; version=3.2)

Other representations (e.g. GeoJSON) for both features and property values are allowed but are not described in this International Standard.

Methods

The following table describes the actions a server should take with the specified HTTP method is used to process the specified resource.

Table 5. Actions for simple transaction resources

Access path	Method	Description
All	OPTIONS	Get the list of supported representations and methods for the resource via the Accept and Allow HTTP headers
/collections/{collectionId}/items	POST	Add a new item or feature to the collection; a recognized representation of the new feature to be added is supplied in the body of the request.
	PUT	Replaces all features or a subset of features (if the query parameters in Table 38 are used) in a collection; a recognized representation of the replacement feature is supplied within the body of the request.
	DELETE	Deletes all features from the specified collection; probably not a good idea to not access control this method!

/collections/{collectionId}/items/{fid}	POST	Not specified.
	PUT	Replaces an existing feature; a recognized representation of the replacement feature is supplied within the body of the request
	DELETE	Deletes or removes the feature identified by the specified URL
/collections/{collectionId}/items/properties/{propertyName}	POST	Not specified
	PUT	Replaces the value of the named property for all features in a collection (or a subset of features in the collection if the query parameters in Table 38 are used); a recognized representation of the replacement value is supplied in the body of the request
	DELETE	Sets the value of the named property to NULL for all features in the collection (or a subset of features if the query parameters in Table 38 are used)
{feature URL}/{prop}	GET	Gets the value of the named property for the feature identified by the specified URL.
	/collections/{collectionId}/items/{fid}/properties/{propertyName}	POST
		Not specified.
		PUT
Replaces the value of the named property for the feature identified by the specified URL; a recognized representation of the value is supplied in the body of the request.	DELETE	Sets the value of the named property to NULL for the feature identified by the specified URL.

NOTE Although not discussed in this extension, using the GET method on the /collections/{collectionId}/items/{fid}/properties/{propertyName} access path can be used to retrieve the value of a feature property rather than fetching the entire feature. Simple scalar properties would probably be returned as text/plain. Complex properties (i.e. properties with many, perhaps nested sub-fields) could be returned using JSON or XML. Furthermore, the access path /collections/{collectionId}/items/properties/{propertyName} could be used to retrieve the value of the specified property for all features in a collection or a subset thereof if query predicates are used to restrict the scope of the operation.

Examples

EXAMPLE: Get the available representations for the collection StructureSrf.

```
CLIENT                                                     SERVER
   |                                                         |
   |   OPTIONS /default/wfs/3.0.0/daraa/collections/StructureSrf   HTTP/1.1
   |   Host: www.pvretano.com                                |
   |-------------------------------------------------------->|
   |                                                         |
   |   HTTP/1.1 200 OK                                       |
   |   Allow: OPTIONS, GET, POST, PUT, DELETE                |
   |   Accept: application/gml+xml; version=3.2,             |
   |           application/geo+json, text/html                |
   |<--------------------------------------------------------|
```

EXAMPLE: Get the available representations and allowed methods for the property F_CODE.

```
CLIENT                                                     SERVER
   |                                                         |
   |   OPTIONS
/default/wfs/3.0.0/daraa/collections/StructureSrf/items/properties/F_CODE   HTTP/1.1
   |   Host: www.pvretano.com                                |
   |-------------------------------------------------------->|
   |                                                         |
   |   HTTP/1.1 200 OK                                       |
   |   Allow: OPTIONS, GET, PUT                              |
   |   Accept: text/plain                                    |
   |<--------------------------------------------------------|
```

NOTE Since the PUT method is included in the list of allowed methods, that means that the value of the property can be updated directly.

EXAMPLE: Get the value of the F_CODE property for a specific feature.

```
CLIENT                                                     SERVER
   |                                                         |
   |   GET
/default/wfs/3.0.0/daraa/collections/StructureSrf/items/CWFID.STRUCTURESRF.0.9/propert
ies/F_CODE
   |   Host: www.pvretano.com                                |
   |-------------------------------------------------------->|
   |                                                         |
   |   HTTP/1.1 200 OK                                       |
   |   Content-Type: text/plain                              |
   |                                                         |
   |   GH301                                                 |
   |<--------------------------------------------------------|
```

EXAMPLE: Add a new StructureSrf feature to the collection. (INSERT)

```
CLIENT                                                      SERVER
   |                                                           |
   |   POST /default/wfs/3.0.0/daraa/collections/StructureSrf/items   HTTP/1.1
   |   Host: www.pvretano.com                                  |
   |   Content-Type: application/geo+json                      |
   |                                                           |
   |   {                                                       |
   |      "type": "Feature",                                   |
   |      "geometry": {                                        |
   |      "type": "Polygon",                                   |
   |      "coordinates":
[[[36.1092489,32.6071056],[36.109329,32.6072337],[36.1091835,32.6072983],[36.1091033,3
2.6071702],[36.1092489,32.6071056]]]
   |      },                                                   |
   |      "properties": {                                      |
   |         "F_CODE": "AL013",                                |
   |         "UFI": "9d0e0ea8-dae1-41c6-b053-44561db4f780",    |
   |         "ZI001_SDV": "2011-01-09T00:30:09Z",              |
   |         "ZI026_CTUC": 5,                                  |
   |         "FCSUBTYPE": 100083                               |
   |      }                                                    |
   |   }                                                       |
   |                                                           |
   |---------------------------------------------------------->|
   |                                                           |
   |   HTTP/1.1 201 Created                                    |
   |   Location:
/default/wfs/3.0.0/daraa/collections/StructureSrf/items/CWFID.STRUCTURESRF.0.10
|
   |<----------------------------------------------------------|
```

EXAMPLE: Update an existing feature. (UPDATE)

```
CLIENT                                                     SERVER
   |                                                          |
   |   PUT
/default/wfs/3.0.0/daraa/collections/StructureSrf/items/CWFID.STRUCTURESRF.0.10
HTTP/1.1
   |   Host: www.pvretano.com                                 |
   |   Content-Type: application/geo+json                     |
   |                                                          |
   |   {                                                      |
   |      "type": "Feature",                                  |
   |      "geometry": {                                       |
   |      "type": "Polygon",                                  |
   |      "coordinates":
[[36.0402269,32.578595],[36.0403499,32.578645],[36.040208,32.5788926],[36.0400851,32.5
788426],[36.0402269,32.578595]]]
   |      },                                                  |
   |      "properties": {                                     |
   |         "F_CODE": "AL013",                               |
   |         "UFI": "9d0e0ea8-dae1-41c6-b053-44561db4f780",   |
   |         "ZI001_SDV": "2011-01-09T00:30:09Z",             |
   |         "ZI026_CTUC": 5,                                 |
   |         "FCSUBTYPE": 100083                              |
   |      }                                                   |
   |   }                                                      |
   |--------------------------------------------------------->|
   |                                                          |
   |   HTTP/1.1 200 OK                                        |
   |   Location:
/default/wfs/3.0.0/daraa/collections/StructureSrf/items/CWFID.STRUCTURESRF.0.10
|
   |<---------------------------------------------------------|
```

EXAMPLE: Change the value of the F_CODE property.

```
CLIENT                                                     SERVER
   |                                                          |
   |   PUT
/default/wfs/3.0.0/daraa/collections/StructureSrf/items/CWFID.STRUCTURESRF.0.10/proper
ties/F_CODE   HTTP/1.1
   |   Host: www.pvretano.com                                 |
   |   Content-Type: text/plain                               |
   |                                                          |
   |   AL999                                                  |
   |--------------------------------------------------------->|
   |                                                          |
   |   HTTP/1.1 200 OK                                        |
   |<---------------------------------------------------------|
```

EXAMPLE: Change the geometric property of a feature.

```
CLIENT                                                           SERVER
   |                                                                 |
   |   PUT
/default/wfs/3.0.0/daraa/collections/StructureSrf/items/CWFID.STRUCTURESRF.0.10/proper
ties/geometry   HTTP/1.1
   |   Host: www.pvretano.com                                        |
   |   Content-Type: application/geo+json                            |
   |                                                                 |
   |   {                                                             |
   |      "type": "Polygon",                                         |
   |
"coordinates":[[[36.1084202,32.605522],[36.1085062,32.6056253],[36.1083383,32.6057245]
,[36.1082523,32.6056213],[36.1084202,32.605522]]]}
   |---------------------------------------------------------------->|
   |                                                                 |
   |   HTTP/1.1 200 OK                                               |
   |<----------------------------------------------------------------|
```

EXAMPLE: Get the modified feature back.

```
CLIENT                                                        SERVER
   |                                                            |
   |   GET
/default/wfs/3.0.0/daraa/collections/StructureSrf/items/CWFID.STRUCTURESRF.0.10
HTTP/1.1
   |   Host: www.pvretano.com                                   |
   |   Accept: application/geo+json                             |
   |----------------------------------------------------------->|
   |                                                            |
   |   HTTP/1.1 200 OK                                          |
   |   Content-Type: application/geo+json                       |
   |                                                            |
   |   {                                                        |
   |      "type": "Feature",                                    |
   |      "geometry": {                                         |
   |      "type": "Polygon",                                    |
   |
"coordinates":[[[36.1084202,32.605522],[36.1085062,32.6056253],[36.1083383,32.6057245]
,[36.1082523,32.6056213],[36.1084202,32.605522]]]}
   |      },                                                    |
   |      "properties": {                                       |
   |         "F_CODE": "AL999",                                 |
   |         "UFI": "9d0e0ea8-dae1-41c6-b053-44561db4f780",     |
   |         "ZI001_SDV": "2011-01-09T00:30:09Z",               |
   |         "ZI026_CTUC": 5,                                   |
   |         "FCSUBTYPE": 100083                                |
   |      }                                                     |
   |   }                                                        |
   |<-----------------------------------------------------------|
```

EXAMPLE: Delete the feature.

```
CLIENT                                                        SERVER
   |                                                            |
   |   DELETE
/default/wfs/3.0.0/daraa/collections/StructureSrf/items/CWFID.STRUCTURESRF.0.10
HTTP/1.1
   |   Host: www.pvretano.com                                   |
   |----------------------------------------------------------->|
   |                                                            |
   |   HTTP/1.1 200 OK                                          |
   |<-----------------------------------------------------------|
```

EXAMPLE: Try to fetch the deleted feature.

```
CLIENT                                                      SERVER
   |                                                           |
   |   GET
/default/wfs/3.0.0/daraa/collections/StructureSrf/items/CWFID.STRUCTURESRF.0.10
HTTP/1.1
   |   Host: www.pvretano.com                                  |
   |---------------------------------------------------------->|
   |                                                           |
   |   HTTP/1.1 404 Not Found                                  |
   |<----------------------------------------------------------|
```

A.11.2. Complex transactions

Introduction

Unlike simple transactions, complex transactions act on multiple features perhaps across multiple collections. During Testbed-14, two approaches for transactions where explored; the document approach and the "shopping-cart" approach.

In the document approach, a single document is presented to the server that contains an encoding of the entire transaction to be preformed. The server reads this document and processes the transaction described therein. As was the case with WFS 2.X, the transaction document is posted to a transaction endpoint.

The shopping cart method makes use to a `transaction` resource. It is assumed that when a new `transaction` resource is created, the entire offerings of the server are accessible through that transactions resource's path. That is to say, a "virtual" and isolated view of the server's entire offerings are available using the transaction resource's path as the root element of the access paths to other feature resources. Using the shopping cart method, a transaction proceeds roughly as follows:

- A new transaction is created by POST-ing an empty body to a transaction resource.
  - The server responds with a Location header indicating the URL of the newly created transaction resource.
- Features are created, modified or deleted within the context of the transaction using the POST, PUT and DELETE methods as described for simple transactions.
- All actions performed within the context of the transaction (i.e. POST, PUT and DELETE) are not committed to the server's repository until a commit token is PUT to the transaction resource URL.
- A transaction can be rolled back by deleting the transaction resource without sending the commit token.

Resources

The following table lists the resource paths for complex transactions.

Table 6. Resources for complex transactions

Access path	Description
/transactions	The transaction factory used to create transaction resources or execute transactions.
/transactions/{txId}	A transaction resource.
/transactions/{txId}/collections/{collectionId}/items	The URL of a feature collection within the context of a transaction.
/transactions/{txId}/collections/{collectionId}/items/{fid}	The URL of a feature within the context of a transaction.
/transactions/{txId}/collections/{collectionId}/items/properties/{propertyName}	The URL of a feature property, within the context of a transaction, that is common to all features in the collection.
/transactions/{txId}/collections/{collectionsId}/items/{fid}/properties/{propertyName}	The URL of a property of a feature within the context of a transaction.

Methods

The following table defines what actions shall be performed for each HTTP method when applied to the resources that support complex feature management.

Table 7. Actions for complex transaction resources

Access path	Method	Description
All	OPTIONS	Gets list of supported representations and methods for the resource via the Accept and Allow HTTP headers.
/transaction	GET	Not specified.
	POST	With an empty body, creates a new transaction resource. With a body containing a transaction document executes and commits the transaction.
	PUT	Not specified.
	DELETE	Not specified.
/transaction/{txId}	GET	See notes after table.
	POST	Not specified.
	PUT	When the request body contains the token "commit", commits the transaction; upon commit the server cleans up the transaction resource. Otherwise not specified.
	DELETE	Deletes the transaction and rolls back any pending changes if the transaction was not previously committed.

/transactions/{txId}/collections/{collectionId}/items	GET	See notes after table.
	POST	Add a new feature to the collection; a representation of the new feature is provided in body of the request.
	PUT	Replaces all existing features in the collection (or a subset thereof based on any query parameters); a representation of the replacement feature is provided in the body of the request.
	DELETE	Deletes all existing features in the collection (or a subset thereof based on any query parameters).
/transactions/{txId}/collections/{collectionId}/items/{fid}	GET	See notes after table.
	POST	Not specified.
	PUT	Replaces an existing feature; a representation of the replacement feature is provided in the body of the request.
	DELETE	Deletes a feature.
/transactions/{txId}/collections/{collectionId}/items/properties/{propertyName}	GET	See notes after table.
	POST	Not specified.
	PUT	Replaces the existing value of the specified property for all features in the collection (or a subset thereof based on any query parameters); a representation of the replacement value is provided in the body of the request.
	DELETE	Sets the value of the specified property to NULL or all features in the collection (or a subset thereof based on any query parameters).
/transactions/{txId}/collections/{collectionId}/items/{fid}/properties/{propertyName}	GET	See notes after table.
	POST	Not specified.
	PUT	Replaces the existing value of the specified property for the specified feature; a representation of the replacement value is provided in the body of the request.
	DELETE	Sets the value of the specified property for the specified feature to NULL.

NOTE The GET method for the `/transactions/{txId}` path is not specified in this ER because it was not tested. The "going-in" idea was that doing a GET on this resource would return a list of editable collections from the server. For each collection, the resource would include a 'rel="edit"' link to indicate where POST, PUT and DELETE method could be used to add, modify or remove feature from that collection within the context of the transactions. This idea was not implemented because of time and resource limitations.

NOTE The GET method for feature resources within the context of a transaction, was not implemented or tested during Testbed-14. However, it can be assumed that GET could be used to retrieve the current value of feature or feature property within the context of a transaction. That is to say (for example), the participants can change the value of a property within the context of a transaction and then get that value back; other users of the system, however, will not see that change until the transaction is committed. This is just the standard database transaction isolation semantic.

NOTE No investigation was made into locking which is something that WFS 2.0 supported. However, defining `lockid` and `lockAction` parameters that could be appended to resources might be an easy way to add to this capability (perhaps for Testbed-15?).

Examples

NOTE Because of time and resource limitation most work on the complex transaction extension was performed using XML in order to leverage the work done to support previous version of the WFS specification. In other words, a lot of existing code can be reused, specifically the code written to implement the abandoned WFS 2.5 specification.

EXAMPLE: Insert, update and delete features atomically using the "shopping cart" method.

STEP 1: Create a Transaction resource

```
CLIENT                                                   SERVER
   |                                                        |
   |   POST /default/wfs/3.0.0/daraa/transactions   HTTP/1.1   |
   |   Host: www.pvretano.com                               |
   |------------------------------------------------------->|
   |                                                        |
   |   HTTP/1.1 201 Created                                 |
   |   Location: /default/wfs/3.0.0/daraa/transactions/1    |
   |   Content-Type: text/xml                               |
   |                                                        |
   |   <?xml version="1.0" encoding="UTF-8"?>               |
   |   <FeatureTypeList xmlns="http://www.opengis.net/wfs/2.5" |
   |      xmlns:xlink="http://www.w3.org/1999/xlink"        |
   |      xmlns:cw="http://www.cubewerx.com/cw"             |
   |      xmlns:atom="http://www.w3.org/2005/Atom"          |
   |      xmlns:xsd="http://www.w3.org/2001/XMLSchema"      |
   |      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"|
   |      xsi:schemaLocation="http://www.opengis.net/wfs/2.5   |
   |          http://www.pvretano.com/schemas/wfs/2.5/wfs.xsd">|
   |      <FeatureType>                                     |
   |         <atom:link rel="edit"                          |
   |
href="http://www.pvretano.com/default/wfs/3.0.0/daraa/transactions/1/AeronauticCrv/ite
ms"/>
   |         <Name>cw:AeronauticCrv</Name>                  |
   |         <NoCRS/>                                       |
   |      </FeatureType>                                    |
   |      .                                                 |
   |      .                                                 |
   |      .                                                 |
   |      <FeatureType>                                     |
   |         <atom:link rel="edit"                          |
   |
href="http://www.pvretano.com/default/wfs/3.0.0/daraa/transactions/1/VegetationSrf/ite
ms"/>
   |         <Name>cw:VegetationSrf</Name>                  |
   |         <NoCRS/>                                       |
   |      </FeatureType>                                    |
   |   </FeatureTypeList>                                   |
   |<-------------------------------------------------------|
```

NOTE Normally an empty body is all that is required but since the code was available, the testbed included a FeatureTypeList body in the response with edit links that a client can use to now POST, PUT and DELETE features to the various editable collections.

STEP 2: Insert a new feature encoded as JSON

```
CLIENT                                                     SERVER
   |                                                         |
   |   POST /default/wfs/3.0.0/daraa/transactions/1/collections/StructureSrf/items
HTTP/1.1
   |   Host: www.pvretano.com                                |
   |   Content-Type: application/geo+json                    |
   |                                                         |
   |   {                                                     |
   |      "type": "Feature",                                 |
   |      "geometry": {                                      |
   |      "type": "Polygon",                                 |
   |      "coordinates":
[[[36.1092489,32.6071056],[36.109329,32.6072337],[36.1091835,32.6072983],[36.1091033,3
2.6071702],[36.1092489,32.6071056]]]
   |      },                                                 |
   |      "properties": {                                    |
   |         "F_CODE": "AL013",                              |
   |         "UFI": "9d0e0ea8-dae1-41c6-b053-44561db4f780",    |
   |         "ZI001_SDV": "2011-01-09T00:30:09Z",             |
   |         "ZI026_CTUC": 5,                                |
   |         "FCSUBTYPE": 100083                             |
   |      }                                                  |
   |   }                                                     |
   |                                                         |
   |-------------------------------------------------------->|
   |                                                         |
   |   HTTP/1.1 201 Created                                  |
   |   Location:
/default/wfs/3.0.0/daraa/transactions/1/collections/StructureSrf/items/CWFID.STRUCTURE
SRF.0.11
   |<--------------------------------------------------------|
```

NOTE Question: Should this really be a 201? A 202 Accepted might be better since the feature will not actually be created in the repository until the transaction is committed. For the moment, the feature is simply accepted.

STEP 3: Replace a feature

```
CLIENT                                                     SERVER
   |                                                         |
   |   PUT
/default/wfs/3.0.0/daraa/transactions/1/AgricultureStr/CWFID.AGRICULTURESRF.0.9
HTTP/1.1
   |   Host: www.pvretano.com                                |
   |   ContentType: application/geo+json                     |
   |                                                         |
   |   {                                                     |
   |      "type": "Feature",                                 |
   |      "geometry": {                                      |
   |      "type": "Polygon",                                 |

"coordinates":[[[36.0782644,32.608309],[36.0768589,32.6082005],[36.0768171,32.6085991]
,[36.0756471,32.608562],[36.0754653,32.6091413],[36.0736628,32.6090781],[36.0732868,32
.6053536],[36.0788867,32.6063478],[36.0790262,32.6069171],[36.0806902,32.6071621],[36.
0806999,32.6067093],[36.0845837,32.607369],[36.0845622,32.6077034],[36.083307,32.60867
95],[36.0782644,32.608309]]]
   |      },                                                 |
   |      "properties": {                                    |
   |         "F_CODE": "EA010",                              |
   |         "UFI": "4f1ffd2f-3918-469a-850f-43ed3ad35b9f",  |
   |         "ZI001_SDV": "2011-12-31T12:41:16Z",            |
   |         "ZI026_CTUC": 5,                                |
   |         "FCSUBTYPE": 100380                             |
   |      }                                                  |
   |   }                                                     |
   |-------------------------------------------------------->|
   |                                                         |
   |    HTTP/1.1 202 Accepted                                |
   |<--------------------------------------------------------|
```

STEP 4: Update the property of a feature

```
CLIENT                                                     SERVER
   |                                                         |
   |   PUT
/default/wfs/3.0.0/daraa/transactions/1/collections/StructureSrf/items/CWFID.STRUCTURE
SRF.0.10/properties/F_CODE   HTTP/1.1
   |   Host: www.pvretano.com                                |
   |   Content-Type: text/plain                              |
   |                                                         |
   |   AL999                                                 |
   |-------------------------------------------------------->|
   |                                                         |
   |   HTTP/1.1 200 OK                                       |
   |<--------------------------------------------------------|
```

STEP 5: Delete a feature

```
CLIENT                                                      SERVER
   |                                                           |
   |   DELETE
/default/wfs/3.0.0/daraa/transactions/1/collections/StructureSrf/items/CWFID.STRUCTURE
SRF.0.7   HTTP/1.1
   |   Host: www.pvretano.com                                  |
   |---------------------------------------------------------->|
   |                                                           |
   |   HTTP/1.1 200 OK                                         |
   |<----------------------------------------------------------|
```

STEP 6: Commit the transaction

```
CLIENT                                                      SERVER
   |                                                           |
   |    PUT /default/wfs/3.0.0/daraa/transactions/1
   |    Host: www.pvretano.com                                 |
   |    ContentType: text/plain                                |
   |                                                           |
   |    commit                                                 |
   |---------------------------------------------------------->|
   |                                                           |
   |    HTTP/1.1 200 OK                                        |
   |    ContentType: text/xml                                  |
   |                                                           |
   |    <?xml version="1.0" encoding="UTF-8"?>                 |
   |    <wfs:TransactionResponse version="2.5.0"               |
   |       xmlns:wfs="http://www.opengis.net/wfs/2.5"          |
   |       xmlns:fes="http://www.opengis.net/fes/2.5"          |
   |       xmlns:atom="http://www.w3.org/2005/Atom"            |
   |       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   |       xsi:schemaLocation="http://www.opengis.net/wfs/2.5  |
   |       http://schemas.opengis.net/wfs/2.5/wfs.xsd          |
   |       http://www.w3.org/2005/Atom                         |
   |       http://schemas.opengis.net/kml/2.2.0/atom-author-link.xsd">
   |       <wfs:TransactionSummary>                            |
   |          <wfs:totalInserted>1</wfs:totalInserted>         |
   |          <wfs:totalUpdated>1</wfs:totalUpdated>           |
   |          <wfs:totalReplaced>1</wfs:totalReplaced>         |
   |          <wfs:totalDeleted>1</wfs:totalDeleted>           |
   |       </wfs:TransactionSummary>                           |
   |       <wfs:InsertResults>                                 |
   |          <wfs:Feature>                                    |
   |             <fes:ResourceId rid="CWFID.STRUCTURESRF.0.11"/>
   |             <atom:link
href="/default/wfs/3.0.0/daraa/transactions/1/collections/StructureSrf/items/CWFID.STR
UCTURESRF.0.11"/>
```

```
   |          </wfs:Feature>                                      |
   |       <wfs:UpdateResults>                                    |
   |          <wfs:Feature>                                       |
   |             <fes:ResourceId rid="CWFID.STRUCTURESRF.0.10"/>
   |             <atom:link
href="/default/wfs/3.0.0/daraa/transactions/1/collections/StructureSrf/items/CWFID.STR
UCTURESRF.0.10"/>
   |          </wfs:Feature>                                      |
   |       </wfs:UpdateResults>                                   |
   |       <wfs:ReplaceResults>                                   |
   |          <wfs:Feature>                                       |
   |             <fes:ResourceId rid="CWFID.AGRICULTURESRF.0.9"/>
   |             <atom:link
href="/default/wfs/3.0.0/daraa/transactions/1/AgricultureStr/CWFID.AGRICULTURESRF.0.9"
/>
   |          </wfs:Feature>                                      |
   |       </wfs:ReplaceResults>                                  |
   |       <wfs:DeleteResults>                                    |
   |          <wfs:Feature>                                       |
   |             <fes:ResourceId rid="CWFID.STRUCTURESRF.0.7"/>|
   |          </wfs:Feature>                                      |
   |       </wfs:DeleteResults>                                   |
   |    </wfs:TransactionResponse>                                |
   |<---------------------------------------------------------|
```

NOTE Normally an empty body is all that is required but since the code was available, the testbed included a wfs:TransactionResponse body in the response.

Appendix B: Revision History

Table 8. Revision History

Date	Editor	Release	Primary clauses modified	Descriptions
June 21, 2018	Vretanos	0.1	all	initial version
June 23, 2018	Portele	0.1	all	comments integrate
June 25, 2018	Harrison	0.1	various	comments integrate
June 29, 2018	Lin	0.1	various	comments integrate
July 11, 2018	Harrison	0.1	various	comments integrate
July 18, 2018	Lin	0.1	various	comments integrate
August 15, 2018	Harrison	0.1	various	comments integrate
August 21, 2018	Lin	0.1	various	comments integrate
August 22, 2018	Vretanos	0.1	various	comments integrate
August 23, 2018	Harrison	0.1	various	comments integrate
August 28, 2018	Portele	0.1	various	comments integrate
August 29, 2018	Lin	0.1	various	comments integrate
August 30, 2018	Vretanos	0.1	various	comments integrate
Sept 10, 2018	Harrison	0.1	various	comments integrate
Sept 11, 2018	Portele	0.1	various	comments integrate
Sept 12, 2018	Lin	0.1	various	comments integrate
Sept 13, 2018	Vretanos	0.1	various	comments integrate
Sept 19, 2018	Harrison	0.1	various	comments integrate
Sept 20, 2018	Lin	0.1	various	comments integrate

Date	Editor	Release	Primary clauses modified	Descriptions
Sept 27, 2018	Vretanos	0.1	various	comments integrate
Sept 28, 2018	Lin	0.1	various	comments integrate
Sept 29, 2018	Portele	0.1	various	comments integrate
Sept 30, 2018	Harrison	0.1	various	comments integrate
Oct 2, 2018	Harrison	0.1	various	comments integrate
Oct 3, 2018	Vretanos	0.1	various	comments integrate
Oct 5, 2018	Harrison	0.1	various	comments integrate
Oct 10, 2018	Harrison	0.1	various	comments integrate
Oct 16, 2018	Harrison	0.1	various	comments integrate
Oct 19, 2018	Harrison	0.1	various	comments integrate
Oct 26, 2018	Vretanos	0.1	various	comments integrate
Oct 26, 2018	Harrison	0.1	various	comments integrate
Oct 29, 2018	Harrison	0.1	various	comments integrate
Nov 21, 2018	Vretanos	0.1	various	Finish update to Annex A. Add cross ref anchors. Handle animated GIF issue.

Appendix C: Bibliography

1. OGC: Web Feature Service (WFS) Conformance Checklist, https://github.com/opengeospatial/WFS_FES/blob/master/guide/conformance_checklist.md.
2. IETF OAuth Working Group: OAuth 2.0 Implicit Grant, https://oauth.net/2/grant-types/implicit/.
3. IETF OAuth Working Group: OAuth 2.0 Authorization Code Grant, https://oauth.net/2/grant-types/authorization-code/.
4. GeoServer: WFS 3 prototype, https://github.com/geoserver/geoserver/tree/master/src/community/wfs3.
5. Portele, C., Genuchten, P. van, Verhelst, L., Zahnen, A.: Spatial Data on the Web using the current SDI, http://geo4web-testbed.github.io/topic4/, (2016).
6. Tandy, J., Brink, L. van den, Barnaghi, P.: Spatial Data on the Web Best Practices. OGC 15-107,Open Geospatial Consortium and World Wide Web Consortium, https://www.w3.org/TR/2017/NOTE-sdw-bp-20170928/ (2017).
7. Portele, C.: Change Request 488: WFS/FES 2.x and work on a multi-part WFS/FES 3.0 standard supporting the Spatial Data on the Web Best Practices, http://ogc.standardstracker.org/show_request.cgi?id=488.