OGC Testbed-14: Modernizing web service standards The next version of the WFS (and other OGC services)

Final Demonstration meeting
ESA/ESRIN, January 2019

CubeWerx

deimos elecnor group

GiS.FCU
GIS RESEARCH CENTER, FCU

GeoSolutions
Your one-stop-shop for geospatial open source software

interactive instruments

OGC®
Making location count.

WFS 2.X (aka OGC 09-025r2)

Key characteristics, strong points:

- Access to spatial databases over HTTP – read and write
- Fine-grained access to spatial data – by feature or even by property
- Full support for GML application schemas
- Advanced queries via Filter Encoding (ISO 19143:2010)
- Very powerful – supports many advanced use cases
- Captures community requirements collected over the last 20 years

Issues, weak points:

- Architecture was bleeding edge 20 years ago, but outdated today and not consistent with the Web architecture
- Significant initial investment required to understand and implement WFS as a server or client
  - Feature-rich, modular structure hard to understand
  - Too many OGC/WFS-specific concepts
- Strong reliance on XML (Capabilities, Filter Encoding, GML, XML Schema)
- A database interface without considerations for access control

WFS 2.X (aka OGC 09-025r2)

Key characteristics, strong points:

- Access to spatial databases over HTTP – read and write
- Fine-grained access to spatial data – by feature or even by property
- Full support for GML application schemas
- Advanced queries via Filter Encoding (ISO 19143:2010)
- Very powerful – supports many advanced use cases
- Captures community requirements collected over the last 20 years

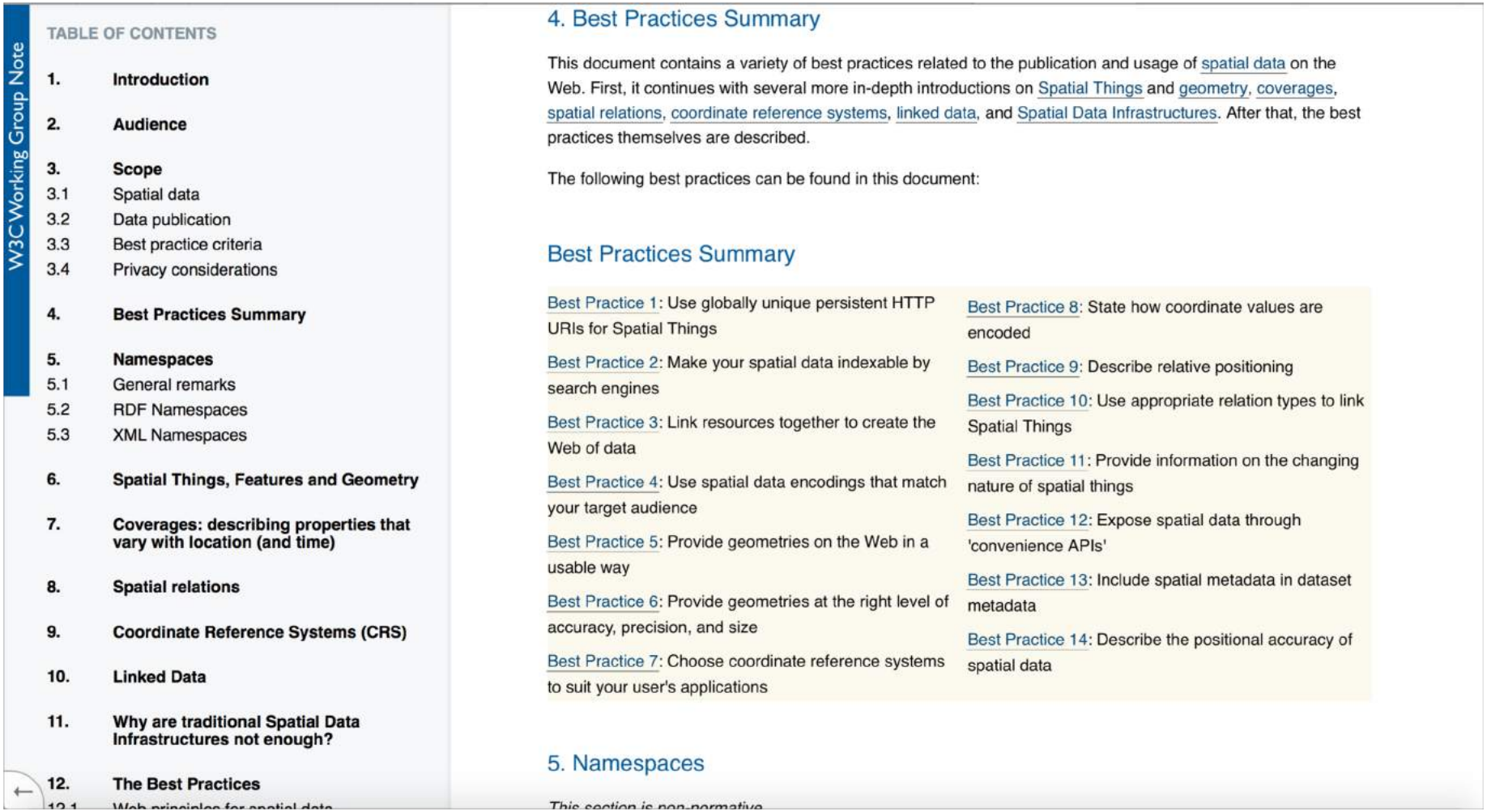
Issues, weak points:

- Architecture was bleeding edge 20 years ago, but outdated today and not consistent with the architecture of the Web
- Significant initial investment required to understand and implement WFS as a server or client
  - Feature-rich, modular structure hard to understand
  - Encapsulates too many OGC/WFS-specific concepts
- Strong reliance on XML (Capabilities, Filter Encoding, GML, XML Schema)
- A database interface without considerations for access control

Conclusions for the next revision of WFS/FES

- **BREAK FREE** of technological and documentation **legacy**
- Use a developer-driven process, do not standardize anything that has not been proven to be useful and **developer-friendly** – in client and server implementations
- Build on the knowledge about the community requirements, but **modernize** the architecture, align it with the current practices on the Web
- **Modularize** the standards into multiple parts – part 1, the “core”, should specify a simple interface to access spatial data that is already sufficient for many use cases
- Remove dependency to XML and XML Schema – in fact, **remove dependency** to any particular encoding and relax requirements for validation against a schema, at least in the core
- Where possible, replace WFS/FES-specific resources and **re-use** existing resources that Web developers are familiar with and which are supported by libraries that are freely available
- Support **secured** services
- **WFS 3.0 is intended to be simpler to use and more modern, but still an evolution from the previous versions and their implementations**

Background: W3C/OGC Spatial Data on the Web Best Practices



W3C Working Group Note

TABLE OF CONTENTS

4. Best Practices Summary

This document contains a variety of best practices related to the publication and usage of spatial data on the Web. First, it continues with several more in-depth introductions on Spatial Things and geometry, coverages, spatial relations, coordinate reference systems, linked data, and Spatial Data Infrastructures. After that, the best practices themselves are described.

The following best practices can be found in this document:

Best Practices Summary

Best Practice 1: Use globally unique persistent HTTP URIs for Spatial Things

Best Practice 2: Make your spatial data indexable by search engines

Best Practice 3: Link resources together to create the Web of data

Best Practice 4: Use spatial data encodings that match your target audience

Best Practice 5: Provide geometries on the Web in a usable way

Best Practice 6: Provide geometries at the right level of accuracy, precision, and size

Best Practice 7: Choose coordinate reference systems to suit your user's applications

Best Practice 8: State how coordinate values are encoded

Best Practice 9: Describe relative positioning

Best Practice 10: Use appropriate relation types to link Spatial Things

Best Practice 11: Provide information on the changing nature of spatial things

Best Practice 12: Expose spatial data through 'convenience APIs'

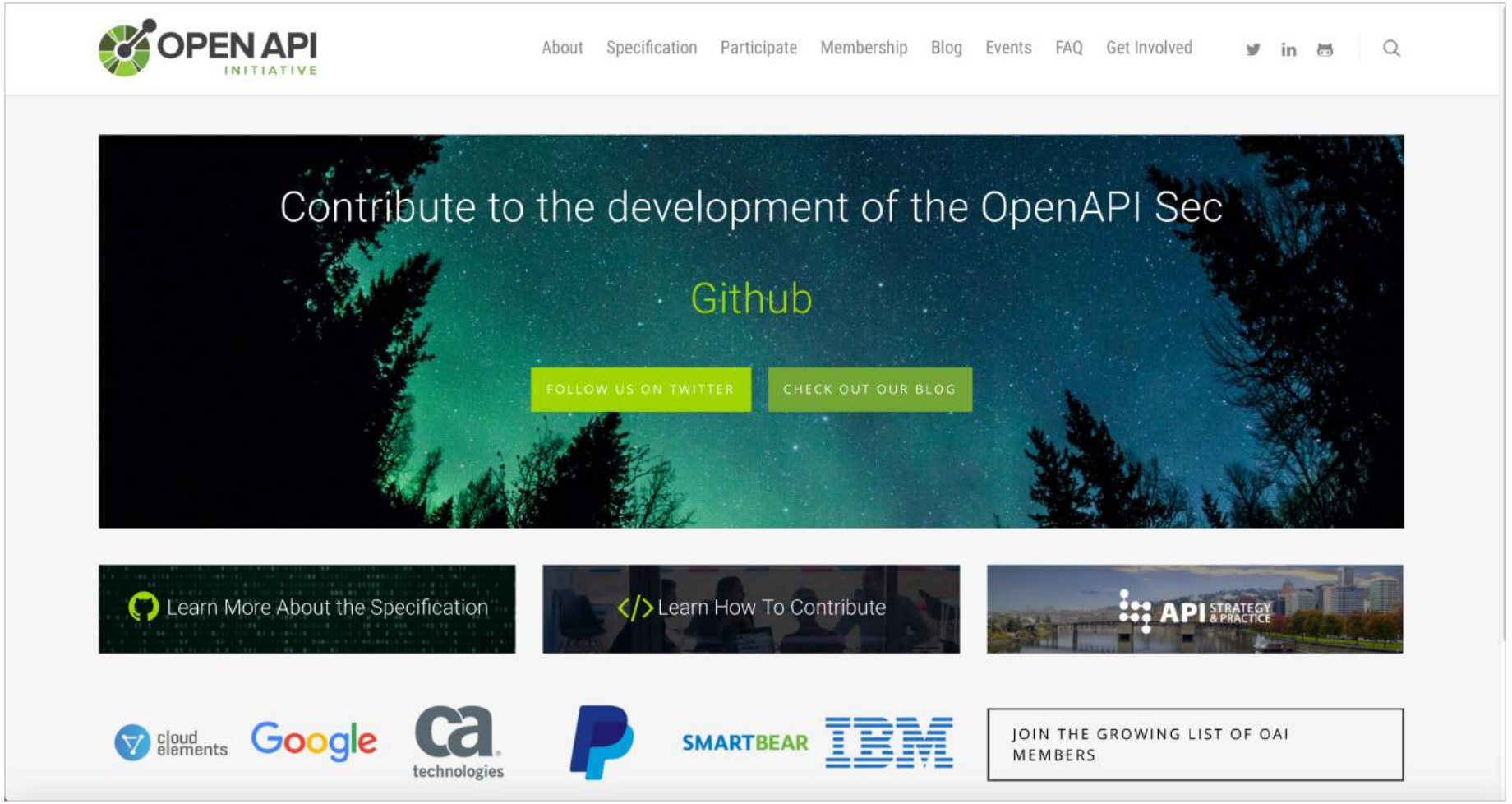
Best Practice 13: Include spatial metadata in dataset metadata

Best Practice 14: Describe the positional accuracy of spatial data

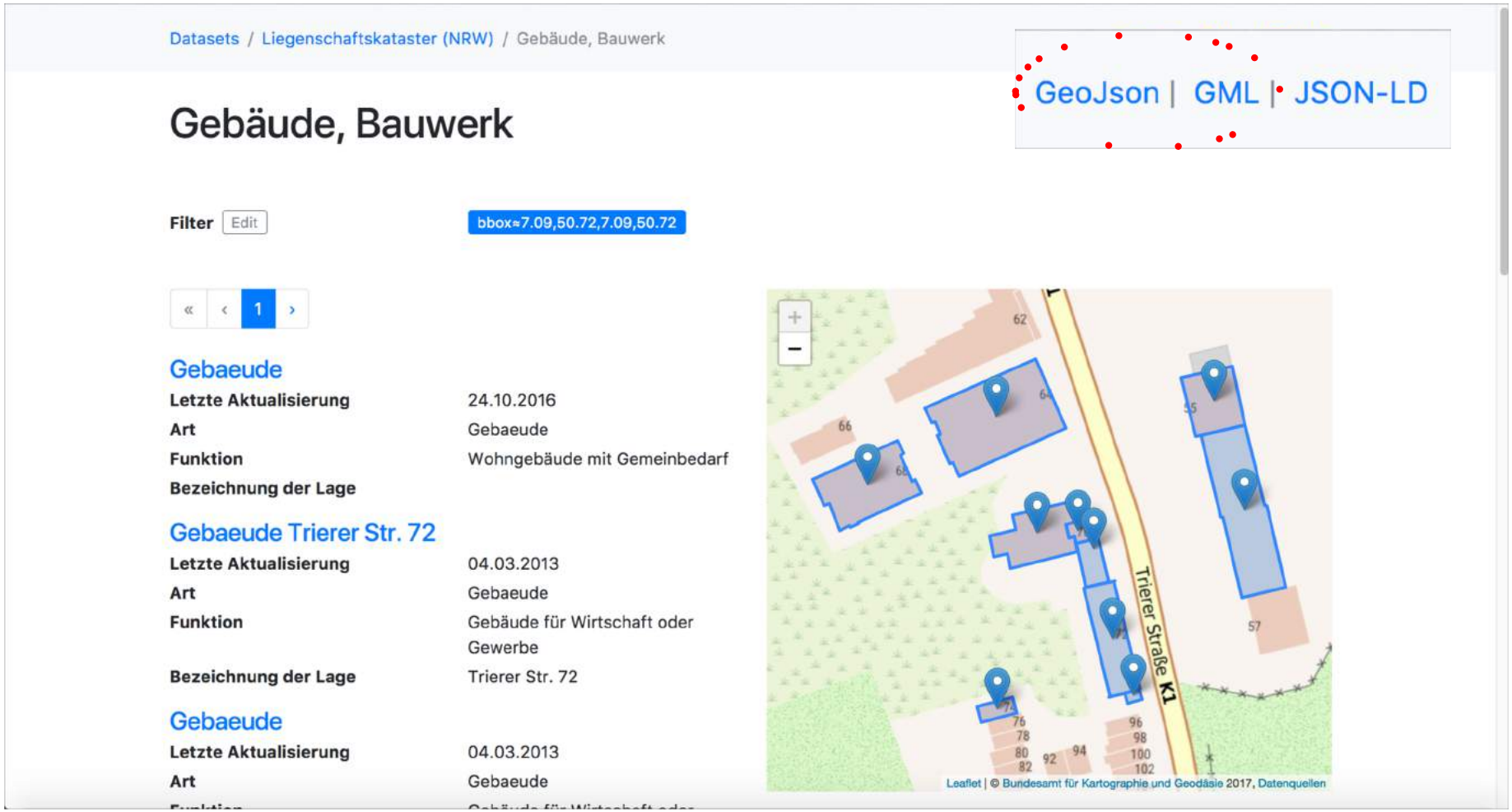
5. Namespaces

This section is non-normative

OpenAPI – replaces Capabilities in WFS 3.0
Supports code-generation, security and more



Encodings:
Rules for HTML, GeoJSON, GML – *all optional*



Web architecture:
Hypermedia driven, conform to HTTP, support for HTML, …

WFS Feature Collection: Built-Up Areas

name/id: builtupa_1m

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

served by: CubeSERV WFS - Foundation (oradb)

Features

Encoded as: text/html

Encoded as: application/gml+xml

Encoded as: application/geo+json

Schema

This feature type has the following schema:

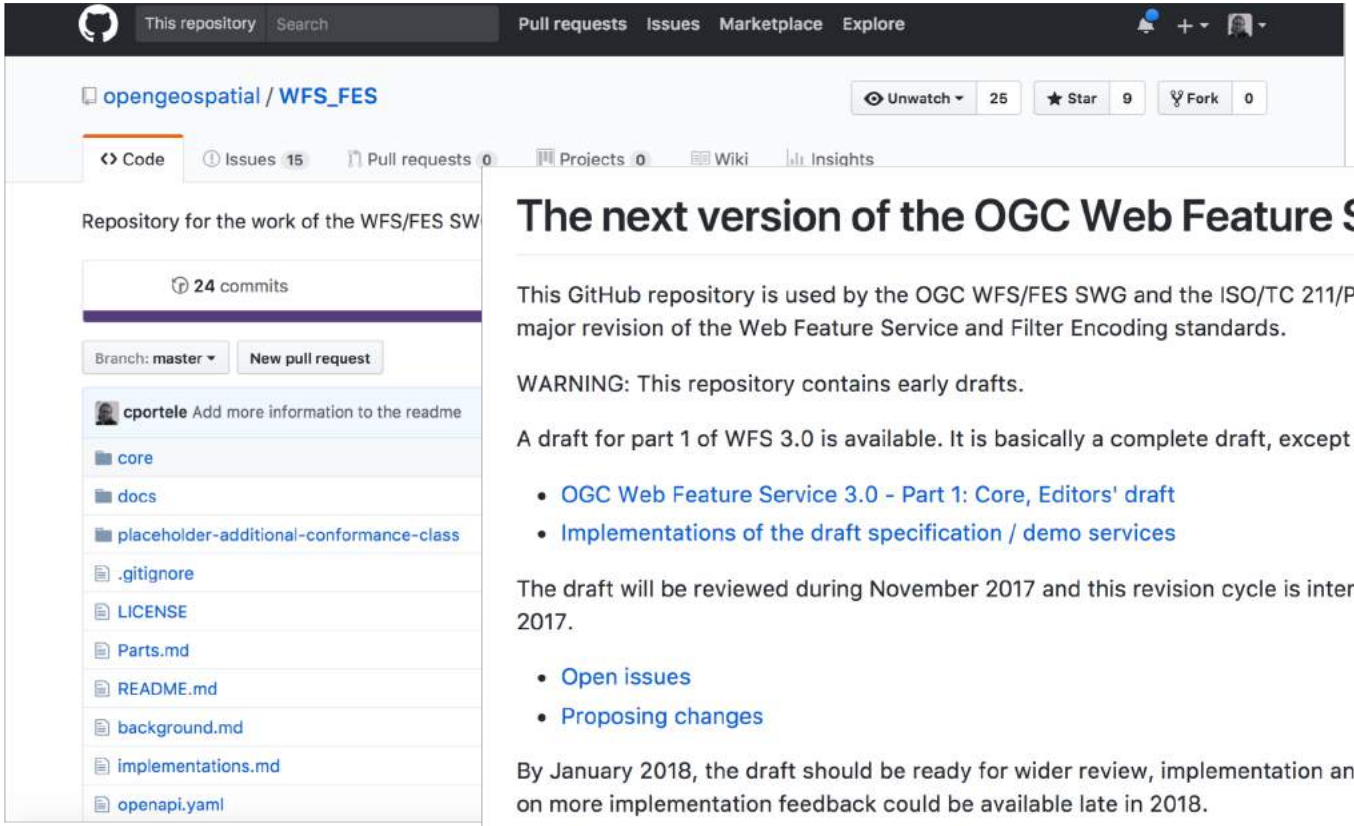
Column Name	Column Type
GEOMETRY	polygon geometry
Row Identifier (id)	integer
FACC Feature Code (f_code)	string (max length: 5)
Name (nam)	string (max length: 80)
Tile Reference ID (tile_id)	integer
Face Primitive ID (fac_id)	integer

Coordinate Reference Systems

The native coordinate reference system of this feature type is:
WGS84 (urn:ogc:def:crs:EPSG::4326)

The following other coordinate reference systems are also available:
NZGD2000_Mount_Pleasant_2000 (urn:ogc:def:crs:EPSG::42110)
WGS84 / Spherical Mercator (urn:ogc:def:crs:EPSG::3857)
NAD27 (urn:ogc:def:crs:EPSG::4267)
NAD83 (urn:ogc:def:crs:EPSG::4269)
NAD83 / UTM zone 3N (urn:ogc:def:crs:EPSG::26903)
NAD83 / UTM zone 4N (urn:ogc:def:crs:EPSG::26904)
NAD83 / UTM zone 5N (urn:ogc:def:crs:EPSG::26905)
NAD83 / UTM zone 6N (urn:ogc:def:crs:EPSG::26906)

Development of the new version in public GitHub repository



The next version of the OGC Web Feature Service standard

This GitHub repository is used by the OGC WFS/FES SWG and the ISO/TC 211/PT 19142+43 to develop the next major revision of the Web Feature Service and Filter Encoding standards.

WARNING: This repository contains early drafts.

A draft for part 1 of WFS 3.0 is available. It is basically a complete draft, except for editorial sections:

- OGC Web Feature Service 3.0 - Part 1: Core, Editors' draft
- Implementations of the draft specification / demo services

The draft will be reviewed during November 2017 and this revision cycle is intended to be completed in December 2017.

- Open issues
- Proposing changes

By January 2018, the draft should be ready for wider review, implementation and feedback. A revised version based on more implementation feedback could be available late in 2018.

Depending on the progress and community interest, work on additional parts supporting transactions, more complex queries, etc. could start in early 2018.

More information:

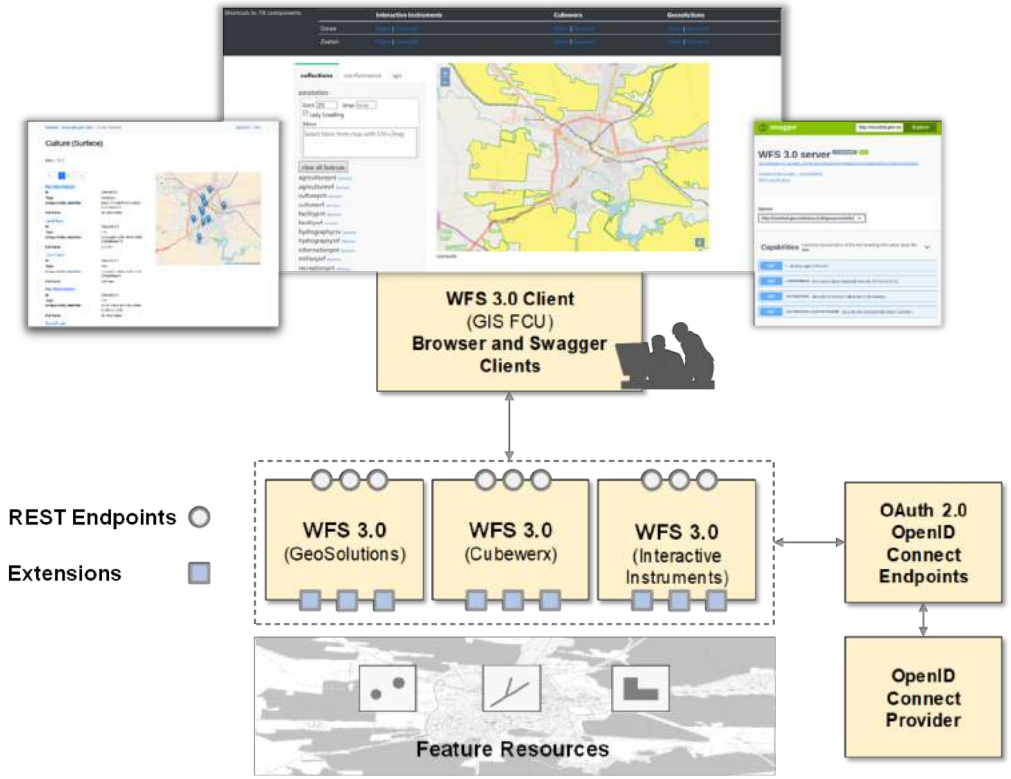
- Background of this activity
- The next version of WFS - an overview

Status and plans

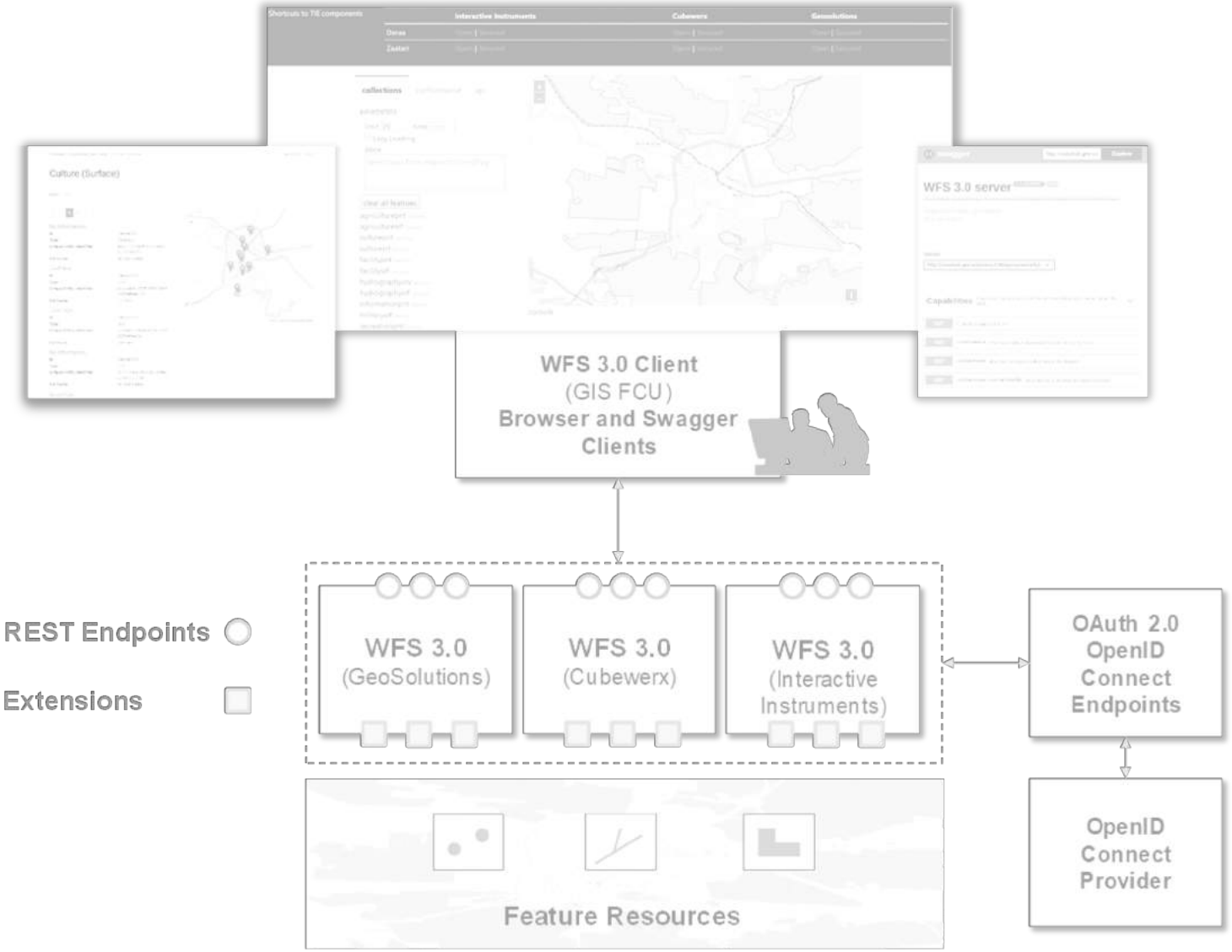
- WFS 3.0, Part 1:
  - Draft has been available for some time
  - Multiple implementations are available
  - Multiple avenues of validation pursued including a hack-a-thon in Fort Collins early in 2018
  - Under review by the joint ISO/OGC working group
  - Release candidate based on implementation feedback in mid/late 2019
  - We are not is a hurry; want to let the draft “bake” well
- Additional parts, Filter Encoding:
  - Work ongoing on additional extensions
    - crs, transactions, advanced queries, etc.
  - Includes existing work from revision of WFS/FES 2.0 since 2012
  - Depending on progress and community interest
  - Verify the modularization approach during 2018
    - Which brings us to TB14 ...

Next Generation APIs - WFS 3.0

- Objective was to develop and test Web Feature Services (WFS) 3.0
- Experiment with new WFS 3.0 specification, OpenAPI, Swagger
- Test security mechanisms based on OpenID Connect and OAuth 2.0
- Assess WFS 3.0 extensions and methods to ease geospatial enterprise transition to next generation APIs

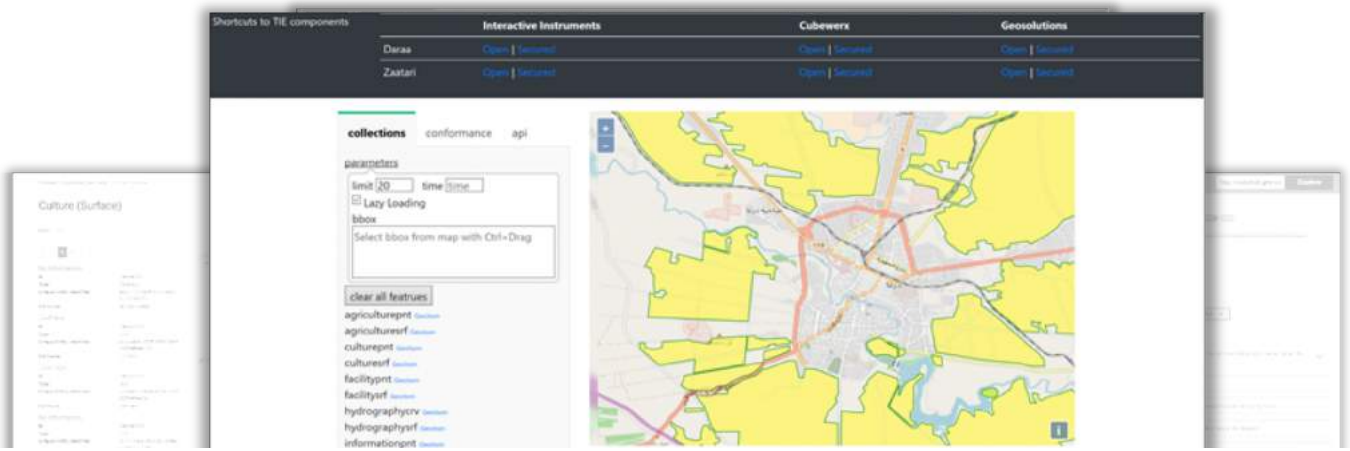


Background

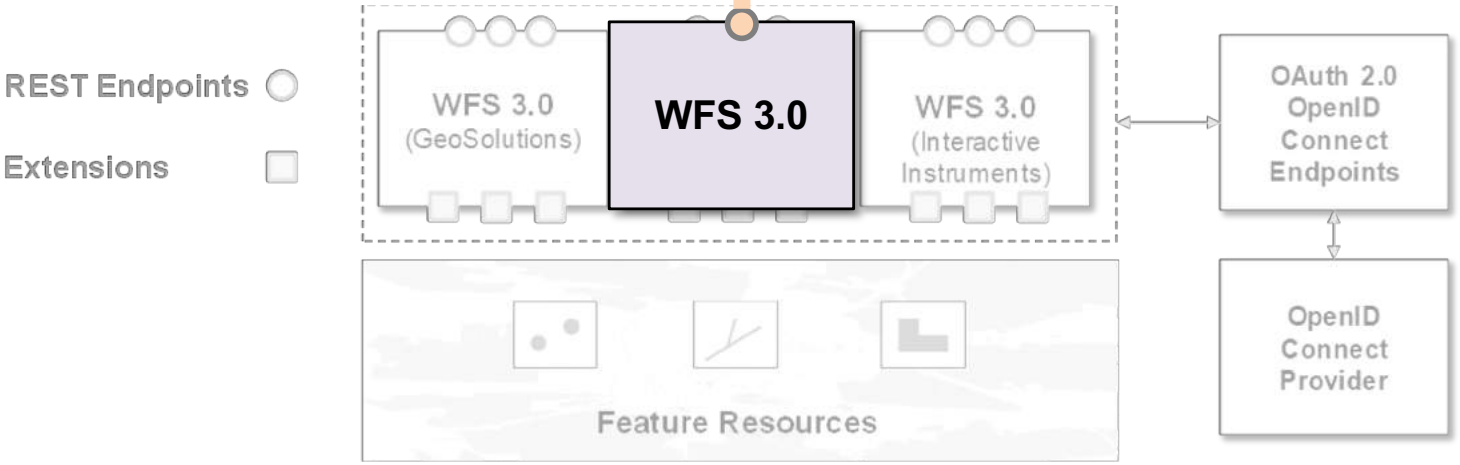


Before we get into details of the Next Generation APIs ER… let's discuss the basics of WFS 3.0, OpenAPI, Swagger

Collections

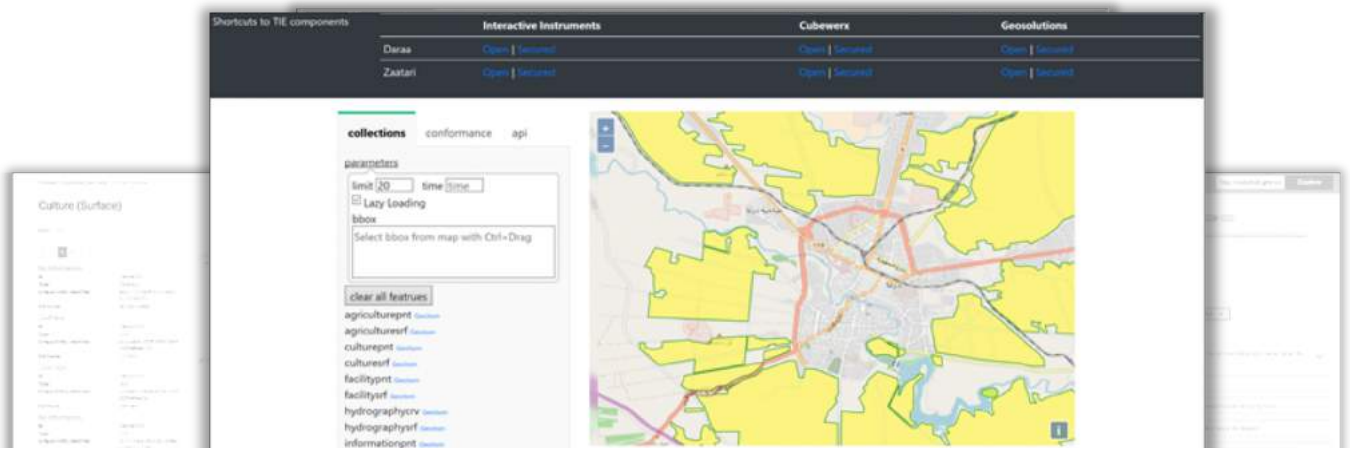



```
GET /collections
```

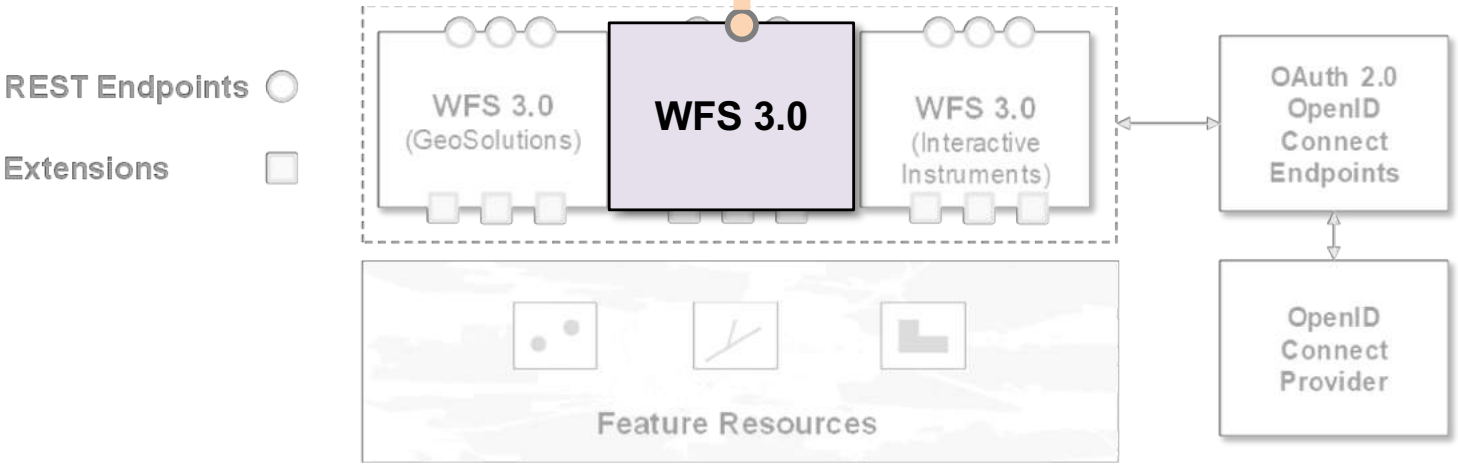


Foundation of WFS 3.0 is set of resources which define ‘core' of the specification. The core provides simple API to access geospatial feature resources as 'collections’. For example, path above lists collections offered by the server.

Feature Resources

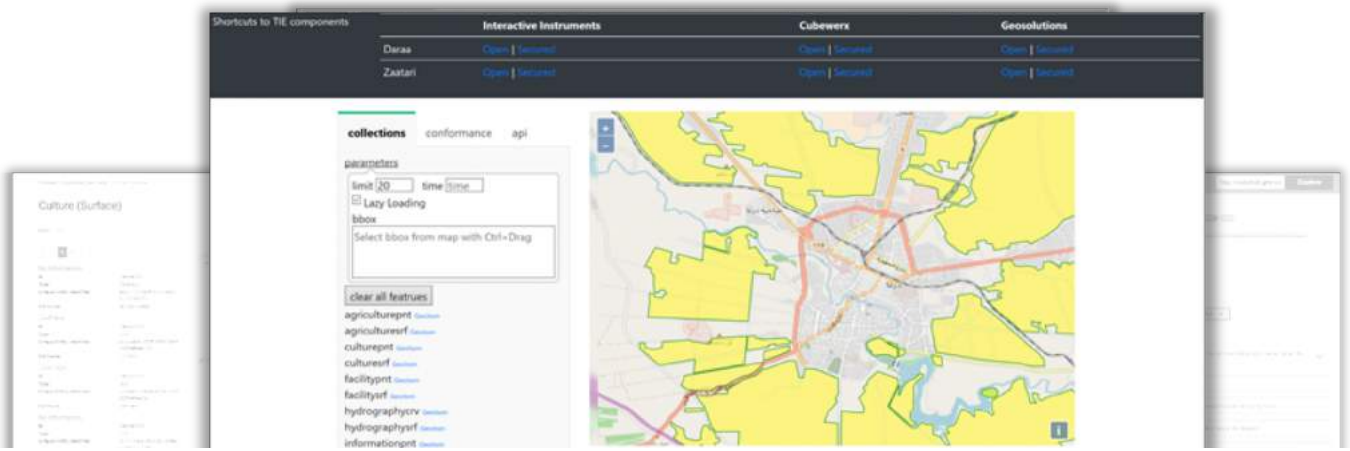



```
GET / collections/agriculturesrf
```

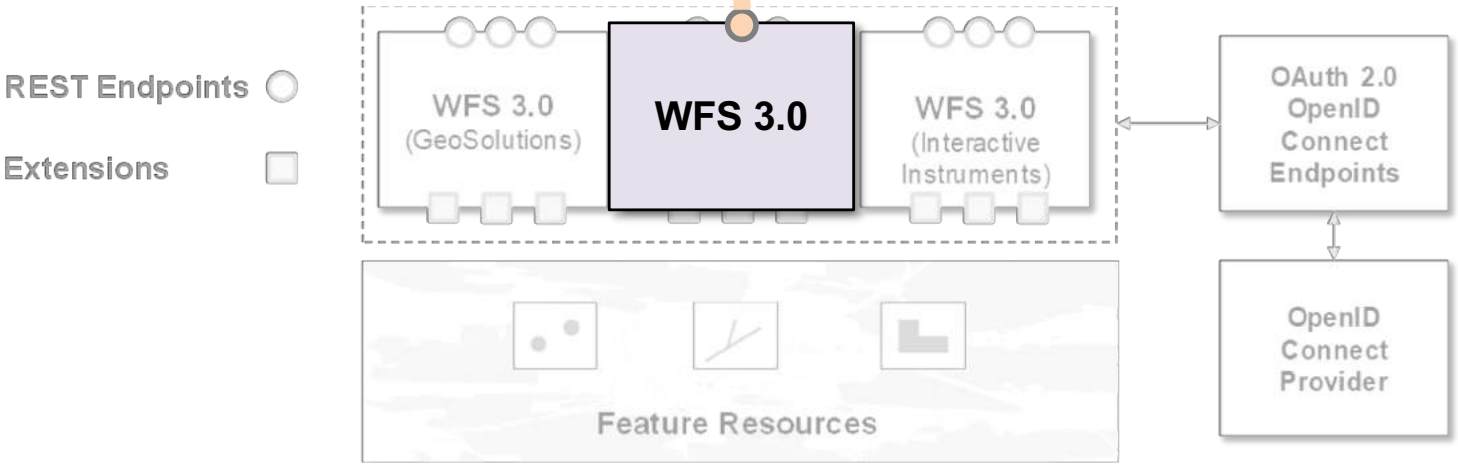


GeoJSON is a recommended encoding for collections provided by WFS 3.0, along with HTML. For example, path above returns metadata about a geospatial feature collection

HTTP Methods & Uniform Interface

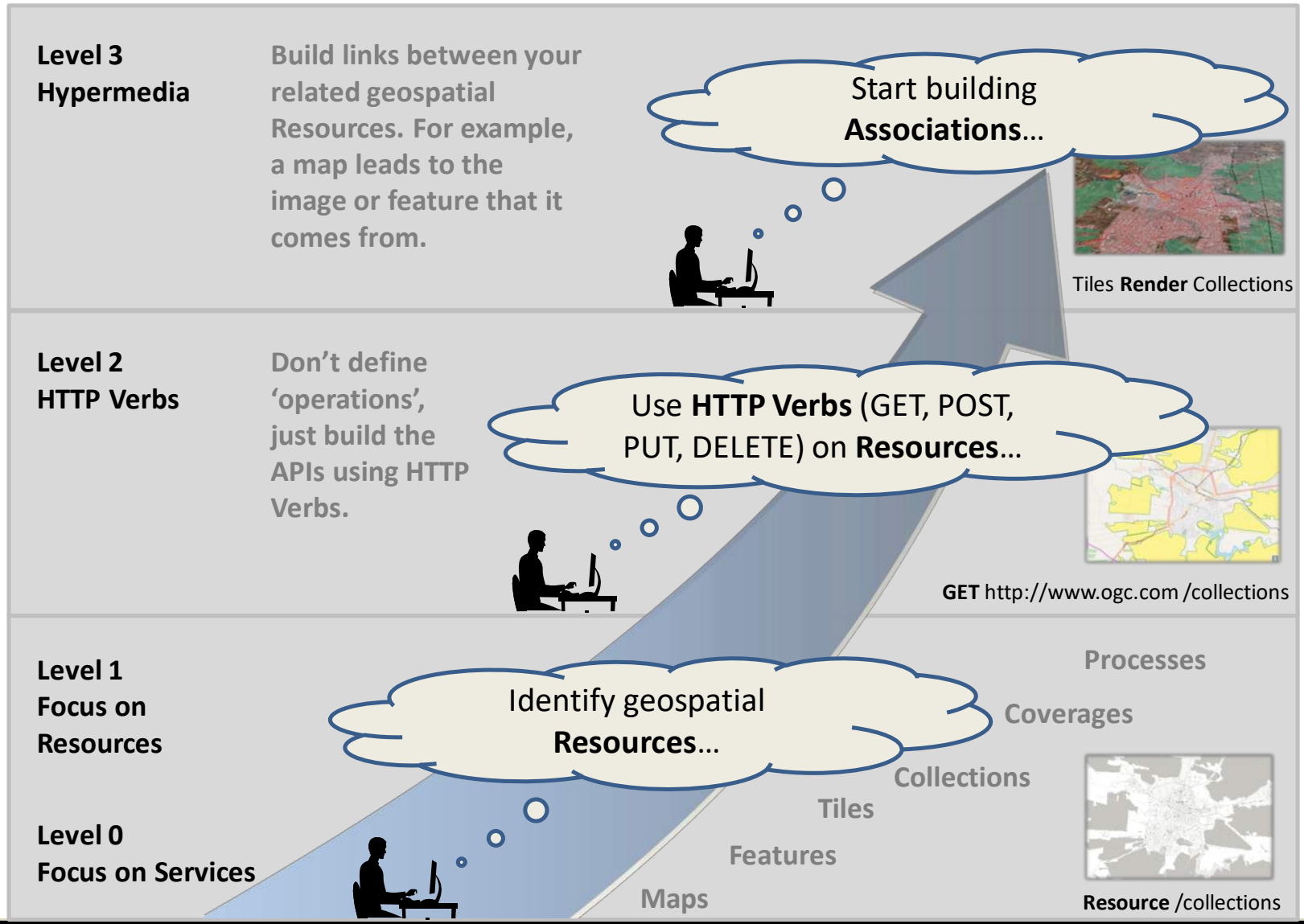



```
GET / collections/agriculturesrf
```



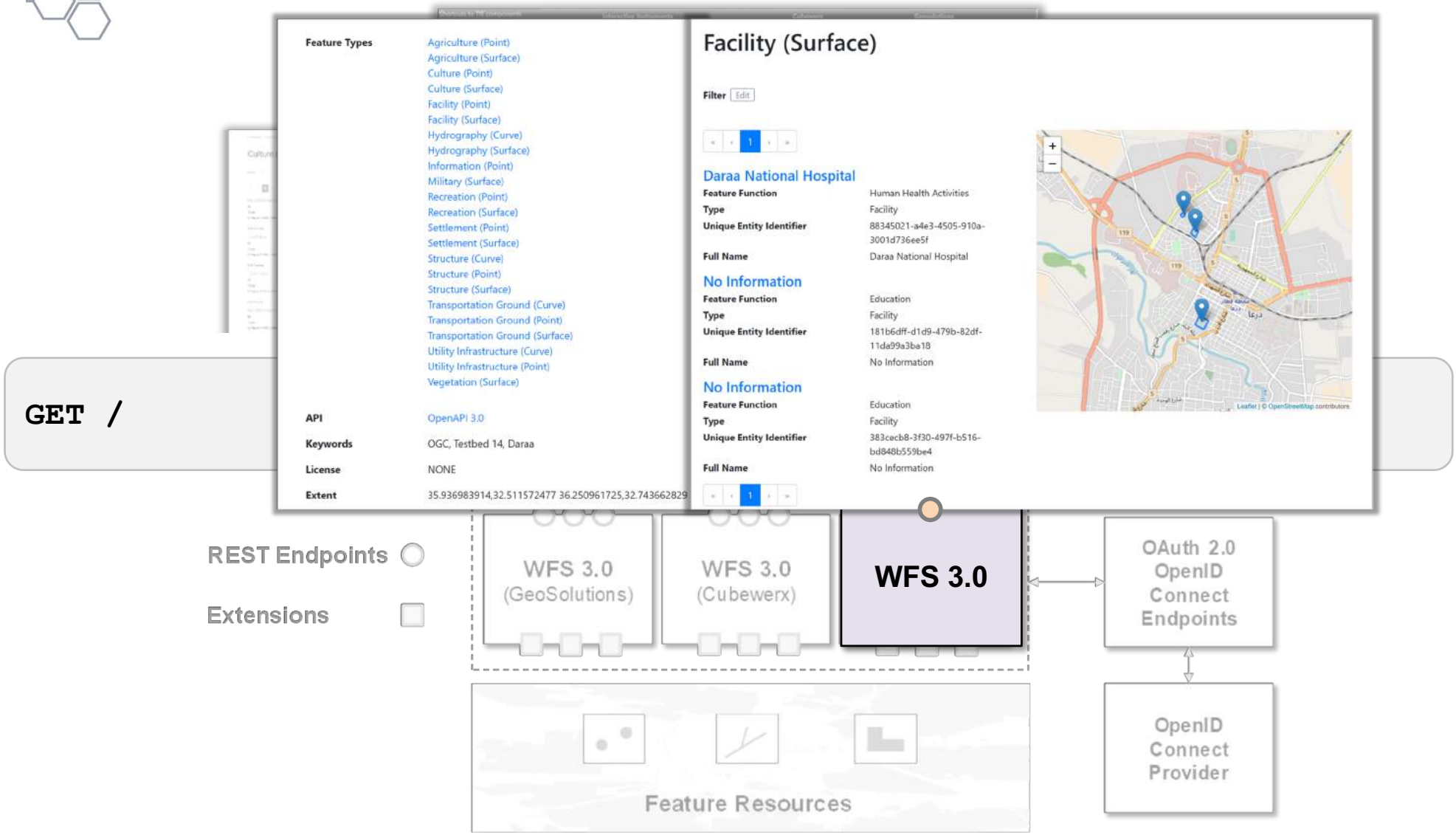
In this approach, the *agriculturesrf* feature resource is accessed from WFS 3.0 API using the HTTP verb GET.
Using HTTP methods GET, POST, PUT, DELETE can make things much easier for developers because the interface is uniform.

Resource Oriented Approach



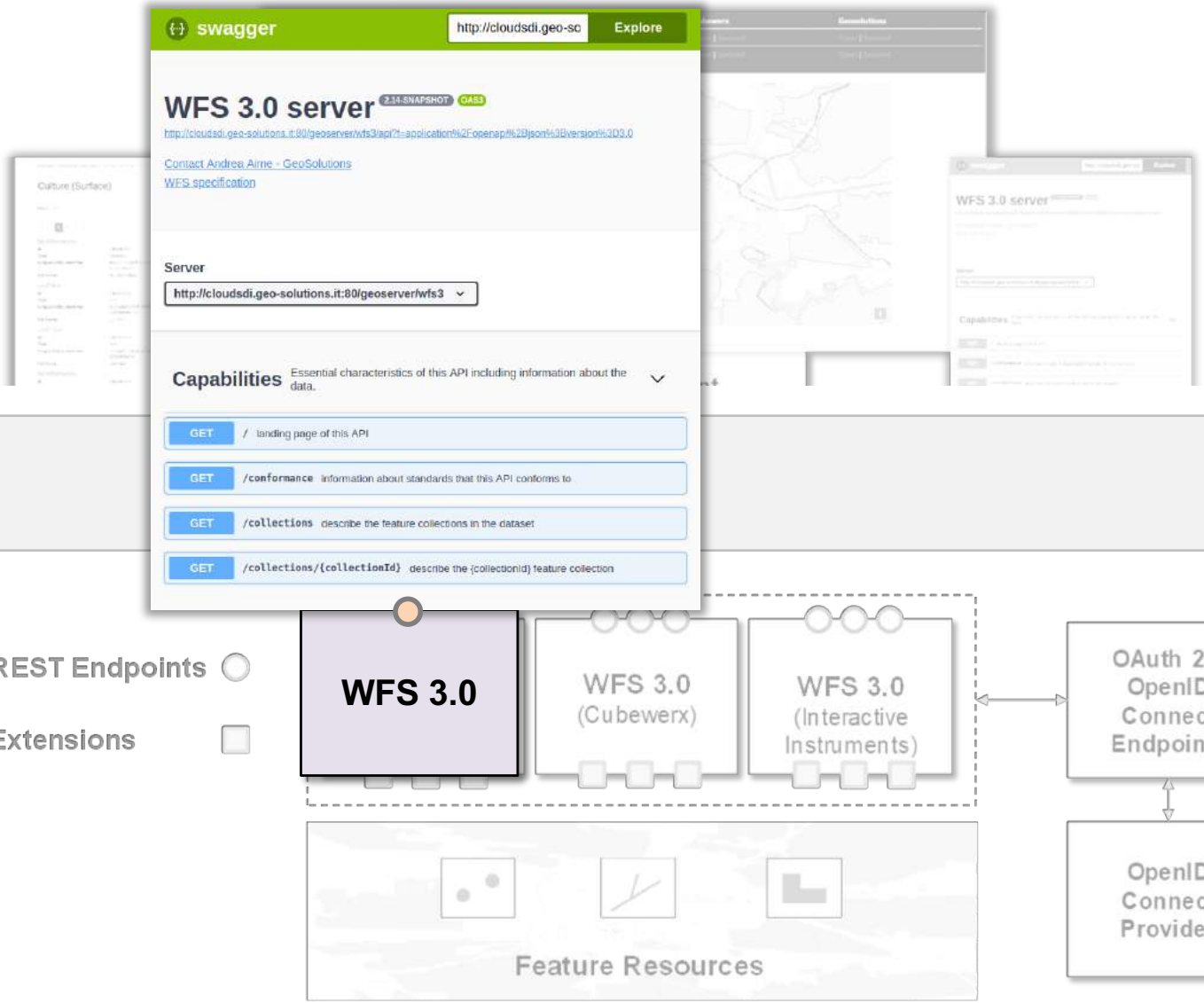
WFS 3.0 consistent with emerging OGC Web API Guidelines and resource oriented approach described in Testbed 12. Advanced functionality is separated into WFS 3.0 extensions – transactions for updates, feature generalization etc.

Landing Page



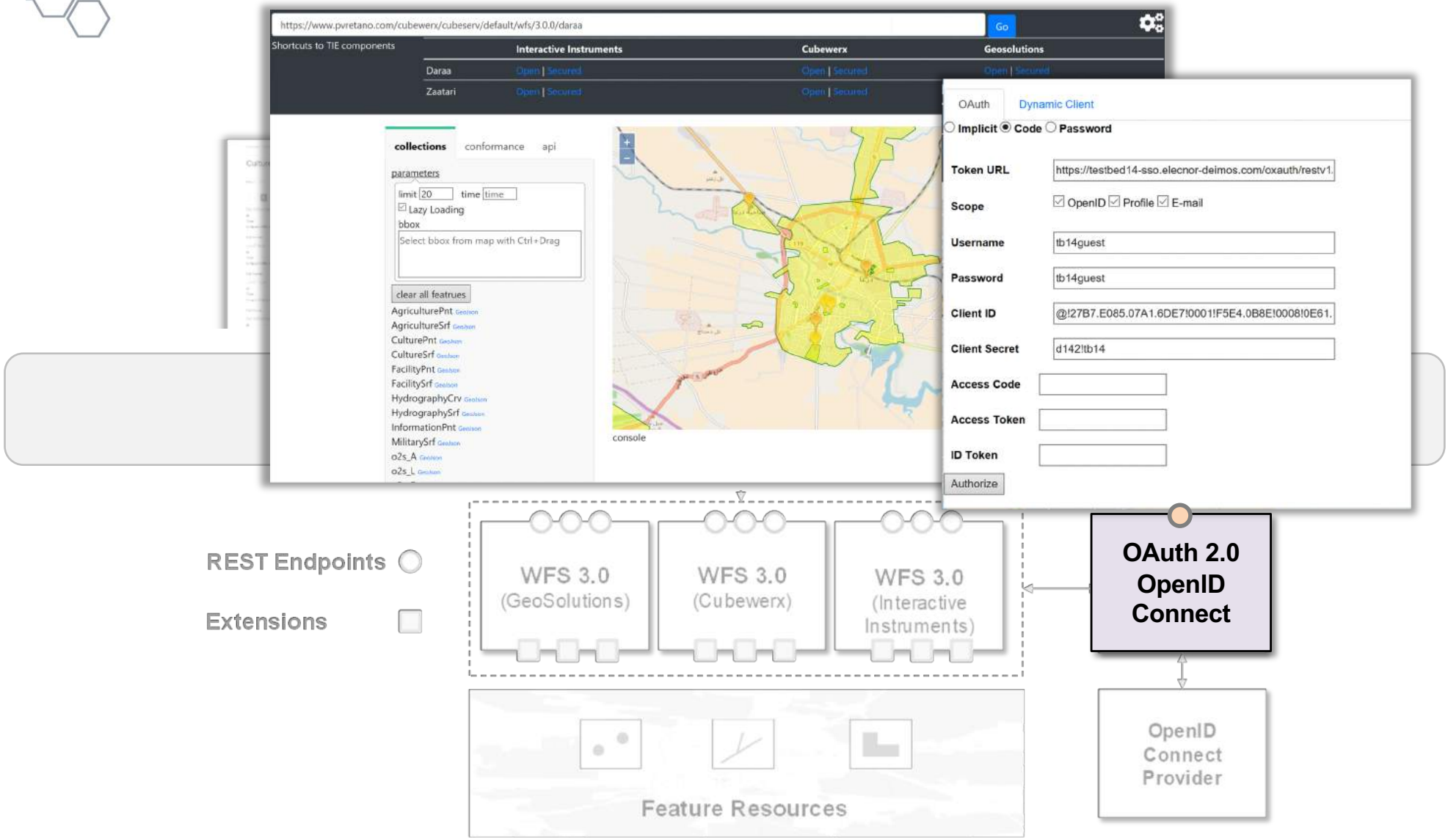
Each WFS 3.0 deploys a landing page available at the 'root' path of the API. Landing page provides links to the resources offered by the service including links to the API description (OpenAPI & others) , supported conformance classes, feature collections description and the feature resources themselves.

OpenAPI and Swagger



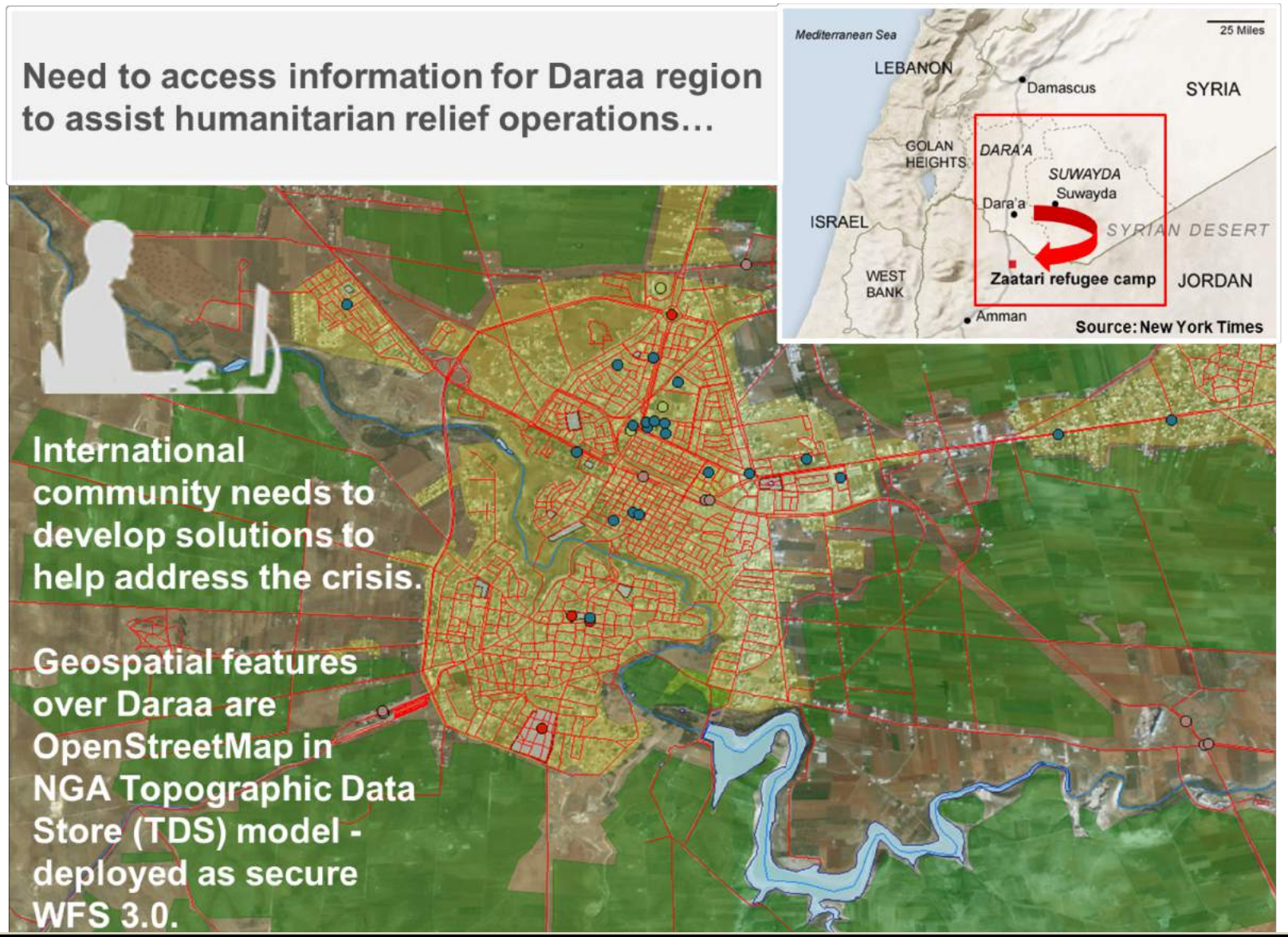
WFS 3.0 minimizes use of WFS-specific components. Uses OpenAPI to provide simple, developer and tool-friendly description of the API. The OpenAPI document can be used in tools such as Swagger.

OpenID Connect and OAuth 2.0



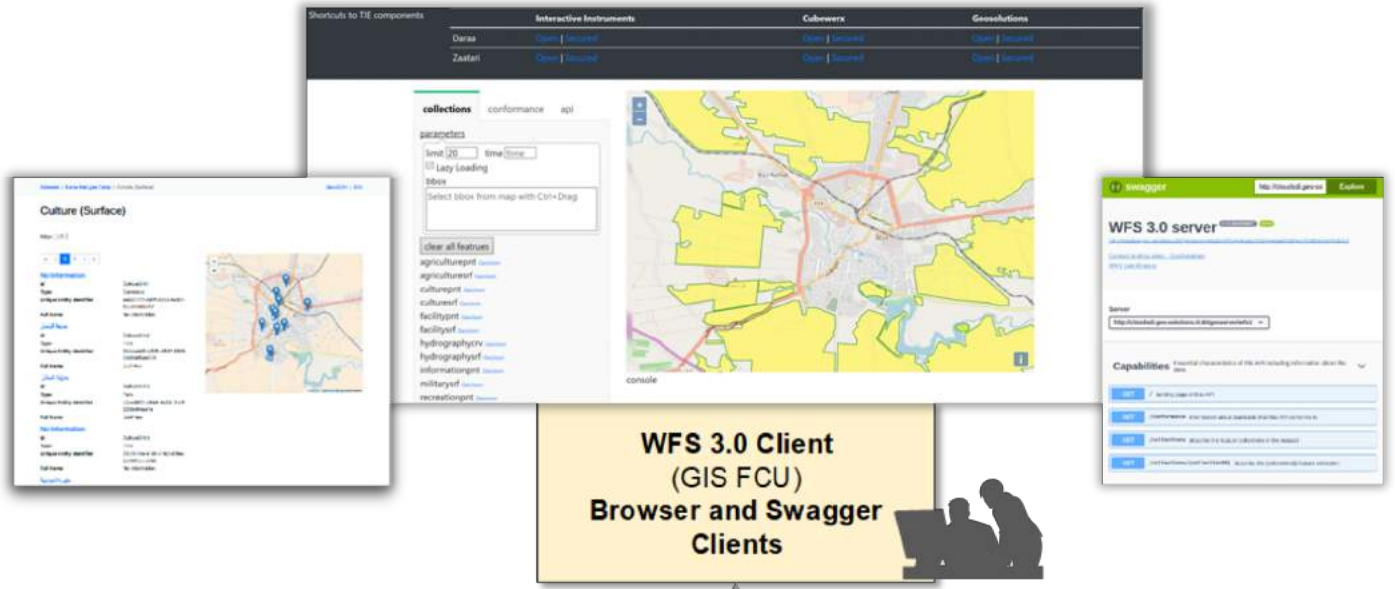
OpenAPI on WFS 3.0 supports multiple security frameworks. For Testbed 14, OpenID Connect and OAuth 2.0 were assessed. OpenID Connect is an authentication layer on top of OAuth 2.0, an authorization framework.

Experiments - Demonstration Scenario



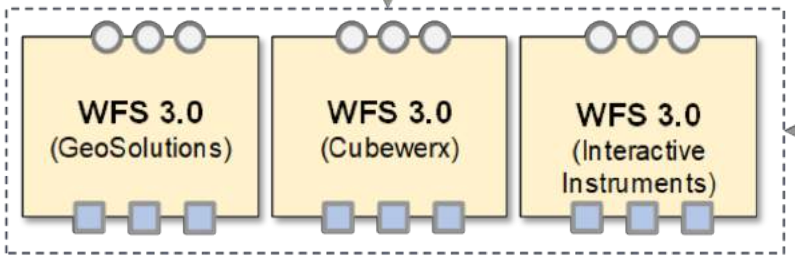
Participants assessed the ability of WFS 3.0 to support simulated users in a humanitarian relief scenario.

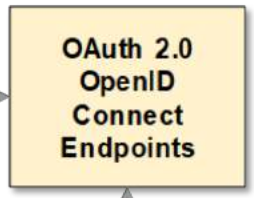
Component Implementation Design

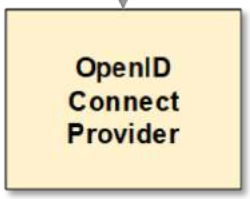


REST Endpoints

Extensions





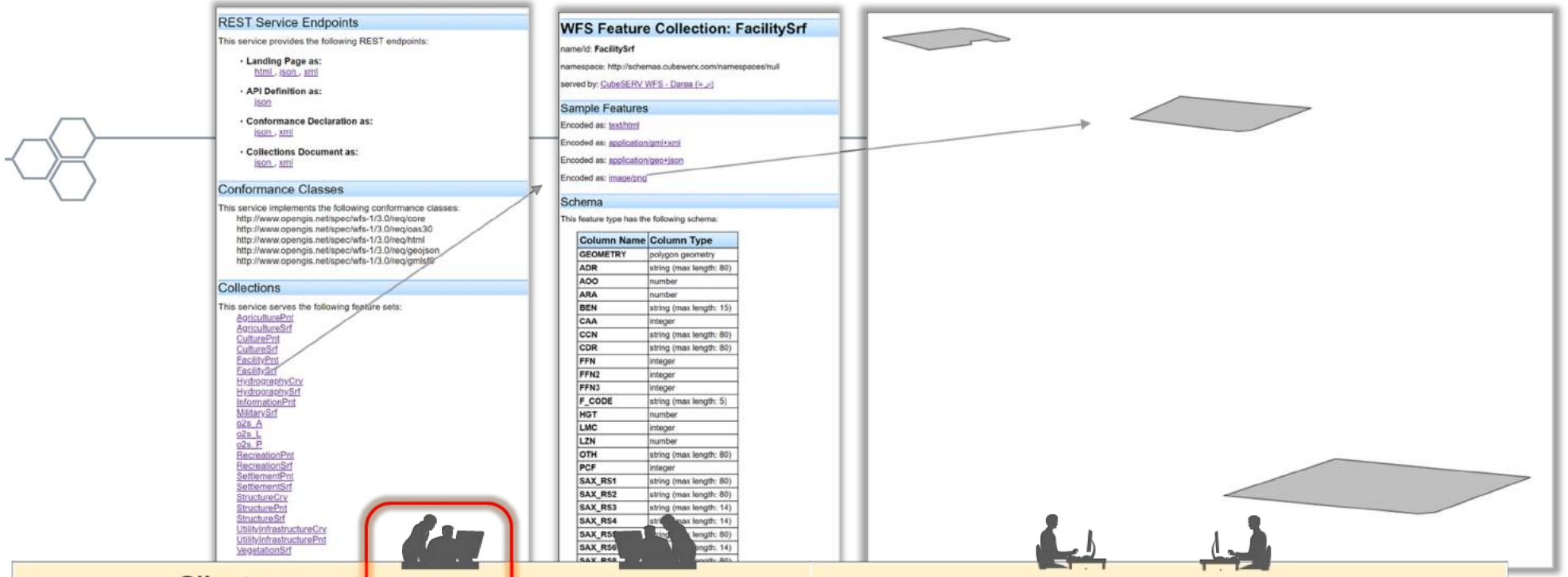


Test architecture included a sequence of interactions between APIs, client applications and security frameworks…

Technology Integration Experiments

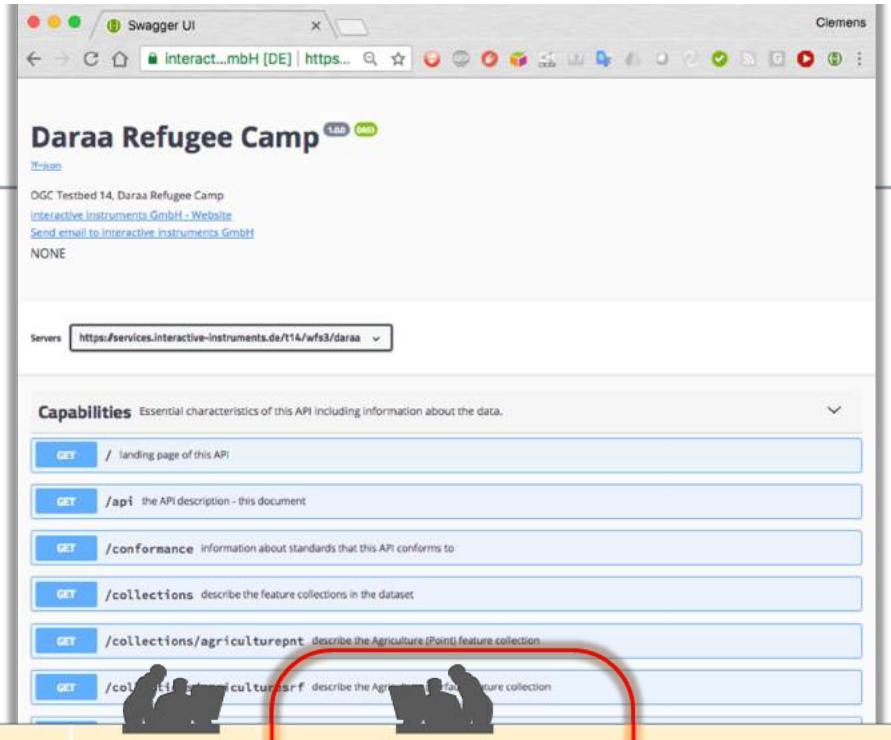
Clients	Browser	Swagger		GIS.FCU Client			
Experiments WFS 3.0	Landing Page	OpenAPI	Client Generator	/api	/conformance	/collections	/collections/ {name}
InteractiveInstruments (Open)	X	X	X	X	X	X	X
InteractiveInstruments (Secure)				X	X	X	X
Cubewerx (Open)	X	X	X	X	X	X	X
Cubewerx (Secure)				X	X	X	X
Geoserver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

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…



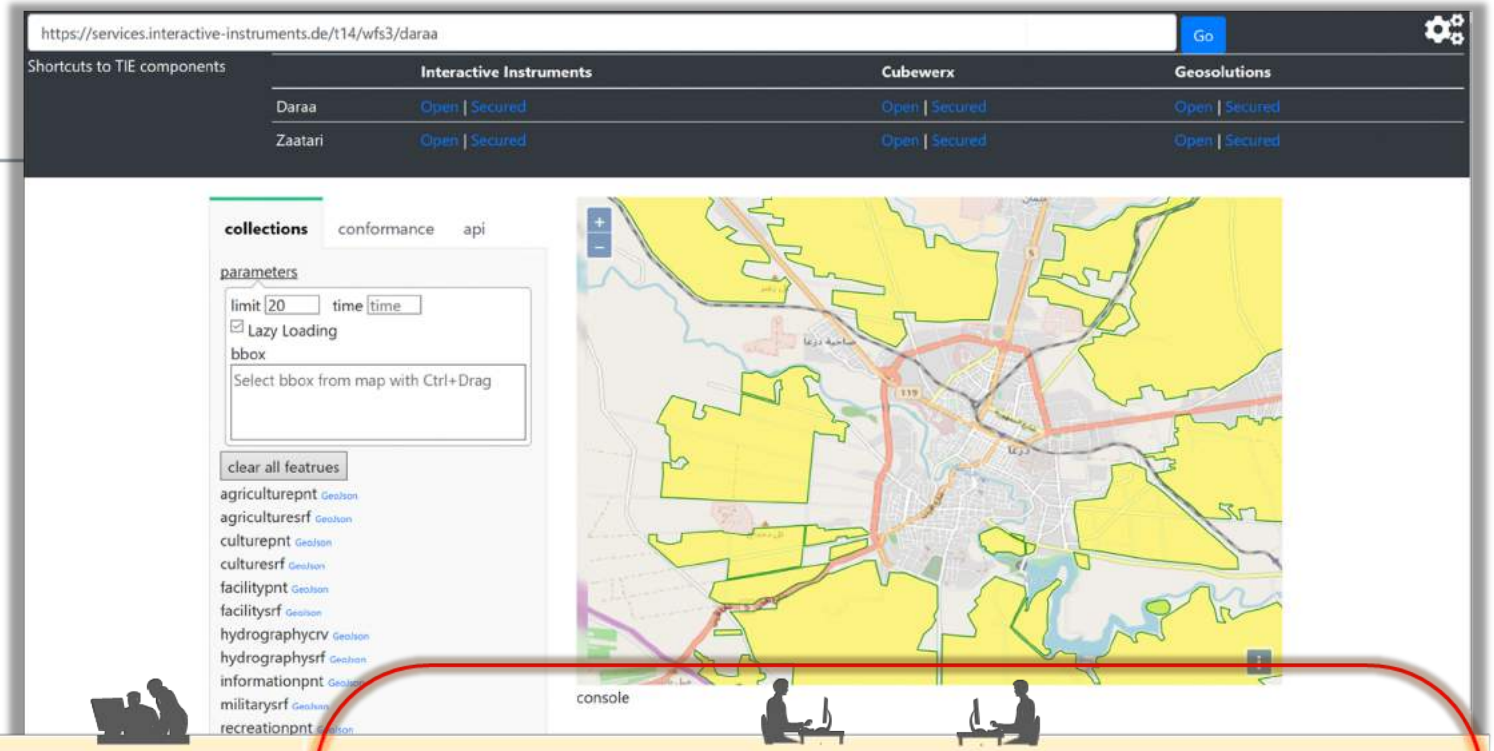
Clients	Browser	Swagger		GIS.FCU Client			
Experiments **WFS 3.0**	Landing Page	OpenAPI	Client Generator	/api	/conformance	/collections	/collections/{name}
Interactive Instruments (Open)	X	X	X	X	X	X	X
Cubewerx (Open)	X	X	X	X	X	X	X
Geoserver (Open)	X	X	X	X	X	X	X

TIEs for Landing Pages, OpenAPIs, Core Conformance classes…



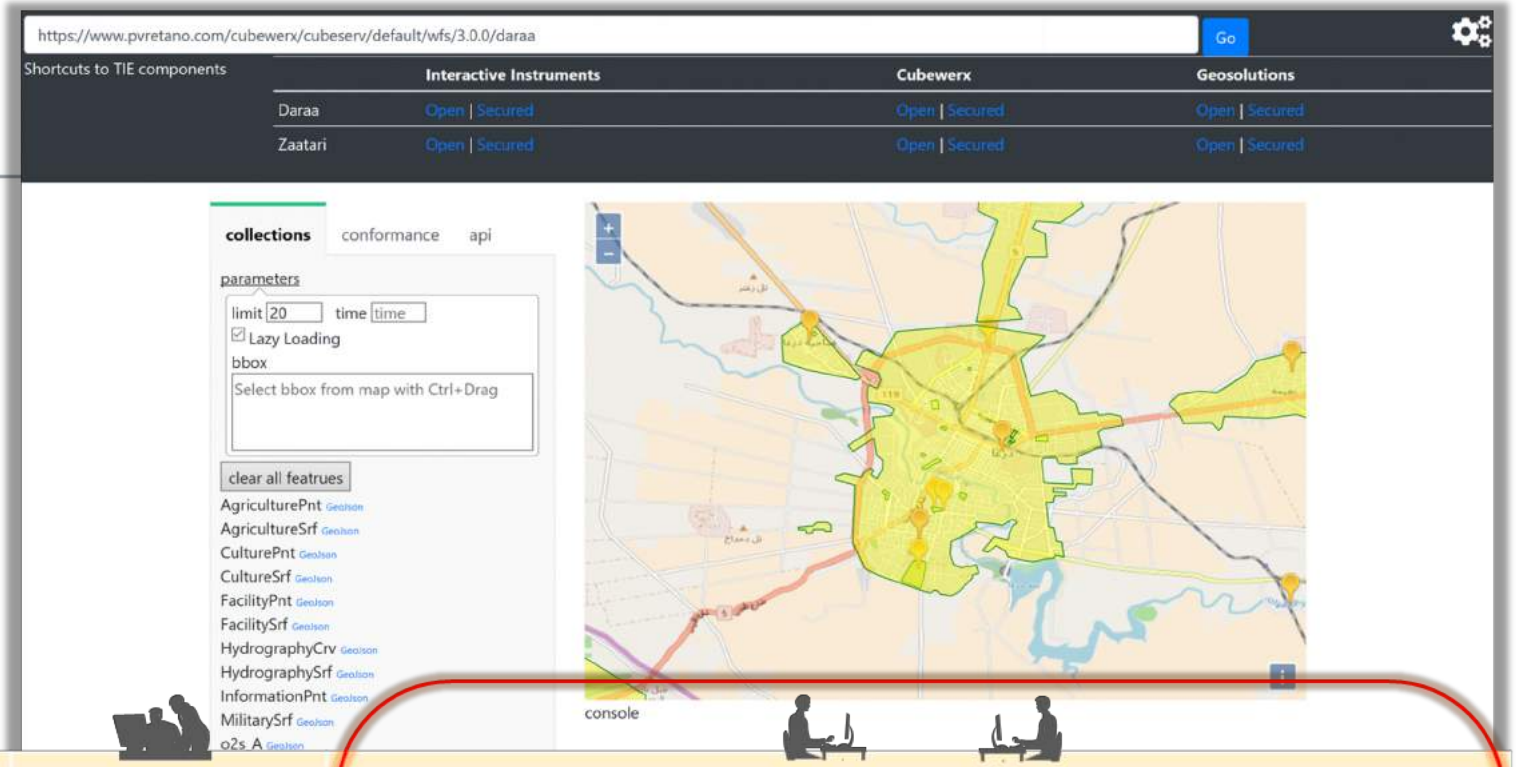
Clients	Browser	Swagger		GIS.FCU Client			
Experiments **WFS 3.0**	Landing Page	OpenAPI	Client Generator	/api	/conformance	/collections	/collections/ {name}
Interactive Instruments (Open)	X	X	X	X	X	X	X
Cubewerx (Open)	X	X	X	X	X	X	X
Geoserver (Open)	X	X	X	X	X	X	X

TIEs for Landing Pages, OpenAPIs, Core Conformance classes…



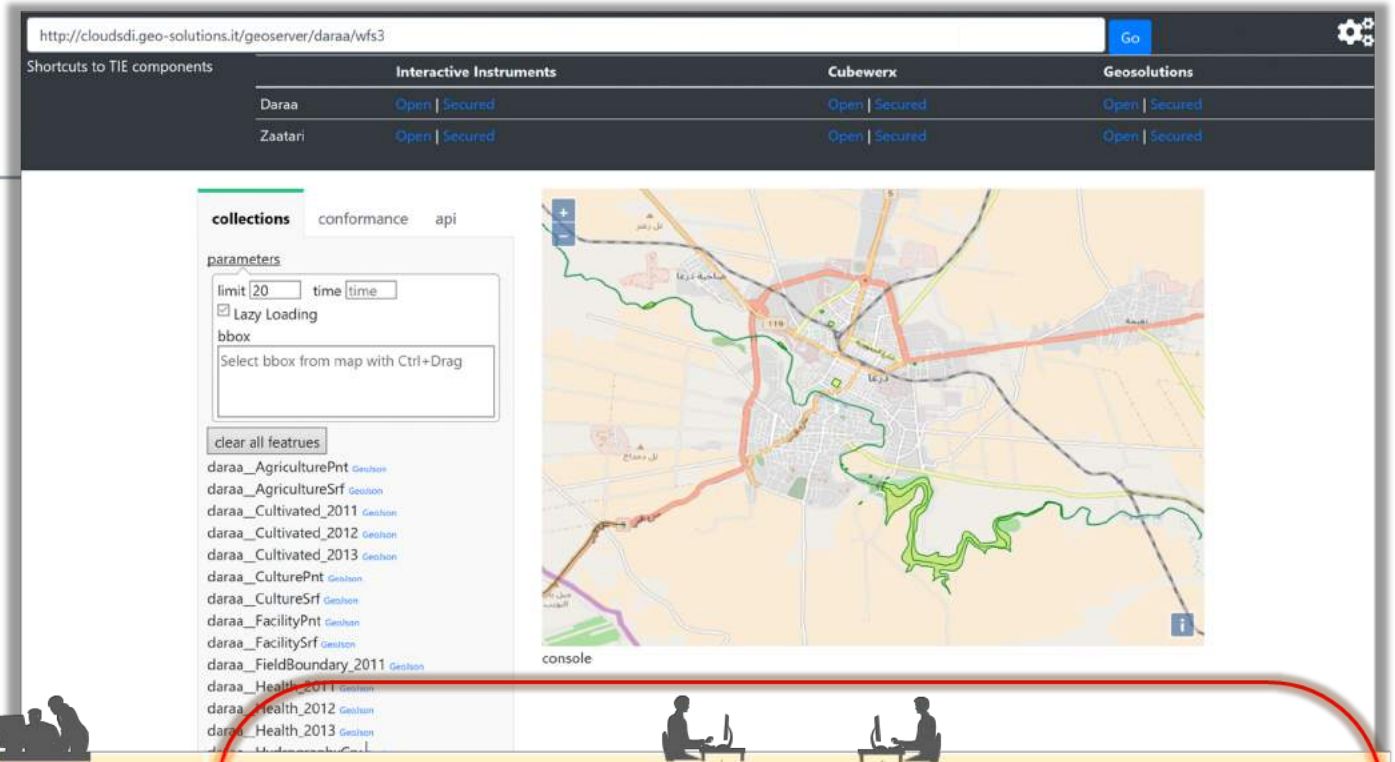
Clients	Browser	Swagger		GIS.FCU Client					
Experiments **WFS 3.0**	Landing Page	Open API	Client Generator	/api	/conformance	/collections	/collections /{name}	/collections /{name} /items	/collections /{name} /items /{id}
Interactive Instruments (Open)	X	X	X	X	X	X	X	X	X
Cubewerx (Open)	X	X	X	X	X	X	X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

TIEs for Landing Pages, OpenAPIs, Core Conformance classes…



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

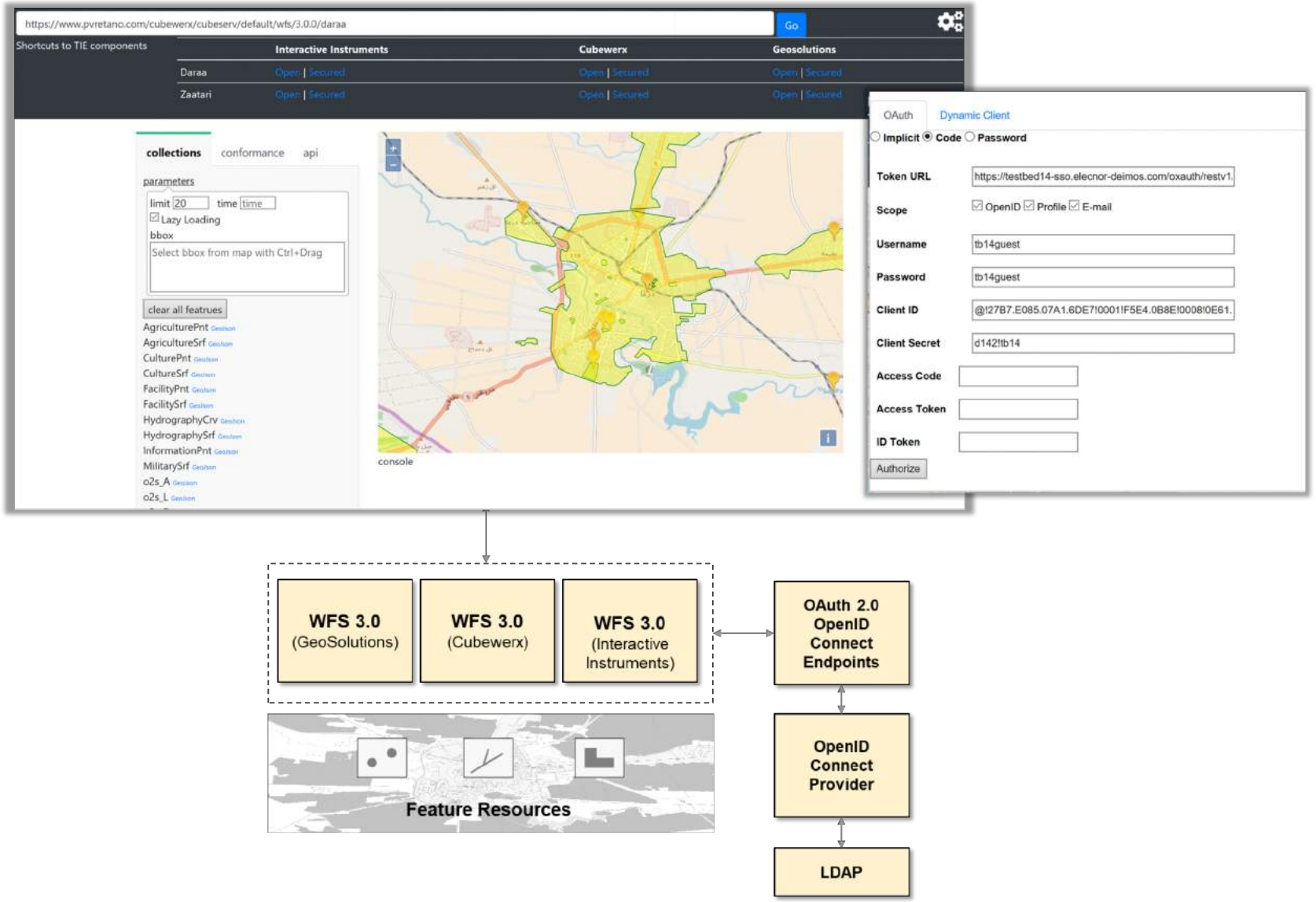
TIEs for Landing Pages, OpenAPIs, Core Conformance classes…



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

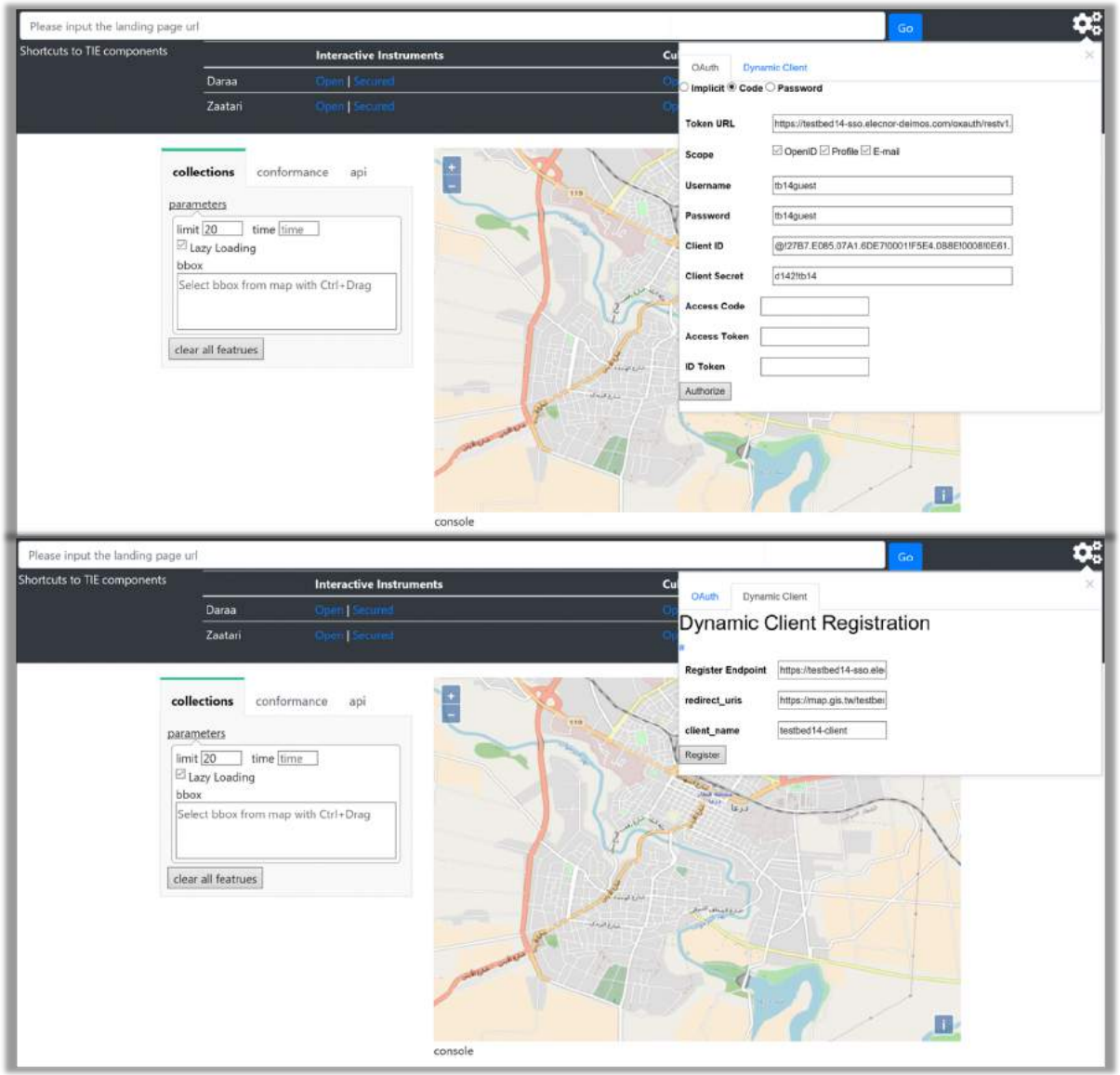
TIEs for Landing Pages, OpenAPIs, Core Conformance classes…

Secured WFS 3.0



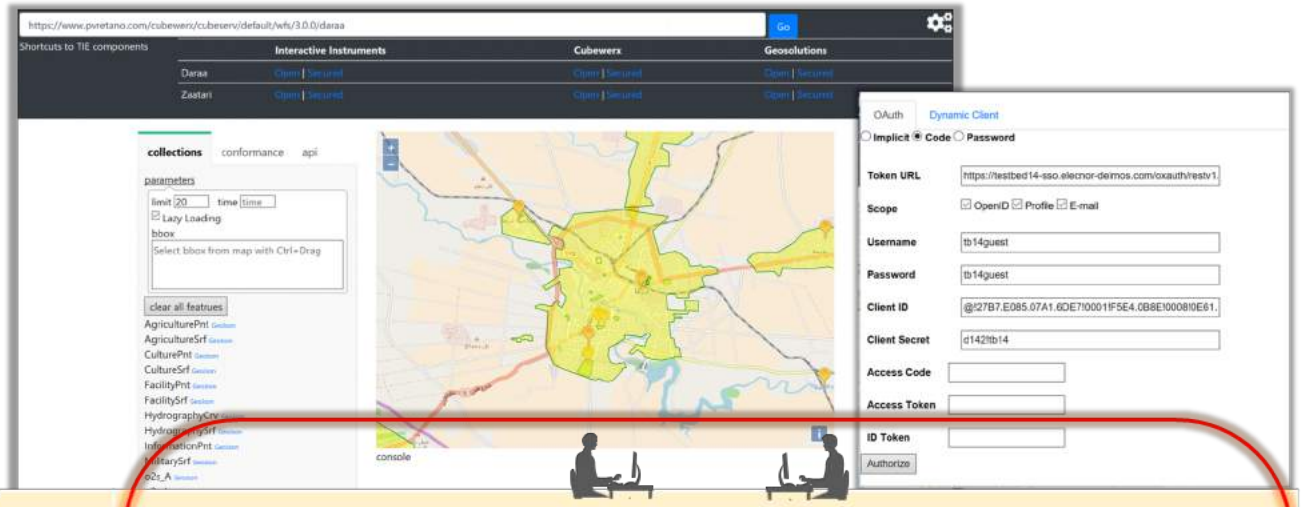
Configuration of OAuth2.0 and OpenID Connect in the Next Generation APIs - WFS 3.0 component implementation design. The client application with security handling is provided by GIS.FCU. Authorization Server is provided by Deimos.

Secured WFS 3.0



In the client application users can choose different OAuth 2.0 permission flows – Implicit Grant, Authorization Code Grant, Password Grant, Dynamic Client Registration flow, etc…

Secured WFS 3.0



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

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

Security (in OpenAPI)

```
"security": [
    …,
    { "oauth2": ["profile",
"openid", "email"] },
    ...
],
```

```
"securitySchemes": {
    …,
    "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"
          }
        }
      }
    },
    …
```

Security (lessons learned)

- Servers that support HTML are also clients and so the client workflow needs to be implemented as well.
- OpenID Connect security scheme in the OpenAPI definition is not visible/supported in the HTML generated by SwaggerUI
- OpenAPI security object does sufficiently describe Oauth/OpenID but does not cover all OGC requirements (Chuck H.)
- Chicken-egg-problem. In order to access secured resources need to read OpenAPI document but must be a “light” OpenAPI document providing just enough info to allow authentication.
- As with other aspects of WFS 3.0, security scheme negotiation must take place between clients and servers
- Cross-Origin Resource Sharing (CORS) scenarios

Extensions

- A number of WFS 3.0 extensions where implemented and tested in the NextGen thread during TB14
- Some of these extensions were for adding new parts to the specification
- Some of these extensions where to test the limits of the API and assess its suitability for resources other than features (e.g. maps, tiles)

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

So What?

- So why is this work important to an imagery organization?
- A fast growing pool of imagery data also generates a fast growing pool of feature data (i.e. derived information)
- In order to to be able to query this information, say to produce some non-imagery information like a report or chart, you need an easy to use, modern, capable and extensible API and that is what we a striving for with WFS 3.0
- Finally, the WFS 3.0 pattern is now being applied to other OGC web services
  - We saw in the previous presentations the application of the pattern to WPS
  - The pattern is also being applied of WCS, CSW, WMTS, etc...