Sponsors

USGS science for a changing world

Radiant.Earth Earth Imagery for Impact

Extensions and Spec Evolution

Some thoughts by Chris Holmes

WFS 3.0 Hackathon

6 – 7 March 2018

OGC® Making location count.

Background

- WFS has a simple core that is implementable but leaves out many things that people have come to expect in WFS
- STAC is not quite as mature, and currently defines an even narrower interface, leaving lots of things open to the implementer.
- How do we enable interoperability and eventual standardization of the types of functionality that people naturally want to build, without ending up with a huge, unwieldly specification? And can we do it in a way that encourages open collaboration and community?

Towards Extensions

- I believe that making it easy to define, publish and share extensions is key.
- Each should be its own mini-specification that is easily understandable (and assumes a working knowledge of the core, or even other extensions)
- OpenAPI has a number of advantages here, can be used to define additional end points
- Should likely be accompanied by a markdown / asciidoc file in GitHub that explains things
- Ideally includes ways to test compliance – JSON Schema, test engine, etc. – though that should not be required to publish, only to make it more of a standard
- Encourage everyone to publish OpenAPI definitions and github repos of their extensions. Keep the barrier low – make templates, etc.
- Can just be formalization of best practices, for example an extension could mostly point at other web standards, but explain how it works with geospatial

Potential Extension ‘levels’

- Implementation Specific Extensions
  - Lowest level of requirements, likely just an OpenAPI definition, a simple narrative in a Github repo, and a link in the WFS repo
- Shared Implementation Extensions
  - If another implementation makes use of the same OpenAPI definition then it can evolve to a ‘shared extension’
  - Should include links to an online implementation for others to try out
  - Rough commitment to reviewing pull requests in GitHub
- Community Extensions
  - Require 3+ server implementation and 2+ client implementations, at least one open source of each
  - Defined process to take changes on the document, with at least one designated ‘extension lead’ who commits to being responsive on github
  - Should explain how it fits in with complementary and competing extensions. But can compete with each other, and with ‘official’ extensions (showing a new way to do things, like using GRPC or a new pubsub method)
- Official Extensions
  - Likely group in to ‘domain specific’ and ‘general’
  - Require test engine (part of main test engine, as additional options)
  - Should only accept one official extension per ‘functionality’ – should not be two different official ’pub sub’ or ‘transactions’

Extensions to Explore

- Aggregation / Statistics – total and return buckets of information to draw graphs, etc.
- Spatial Aggregation / coverage maps – return heatmap / gridded type results
- Updates / pubsub – keep two WFS's in sync, formalize an 'update' field
- Transactions
- Links to tile servers / other OGC services that portray the data
- Alternate output formats – geopackage, shapefile, postgis, protobuf, etc
- Generalization – return simplified geometries for display, etc
- Bulk download – async operations to enable download of millions of rows in GIS formats
- gRPC – alternate endpoint structure for streaming
- Many more, additions welcome