Towards Flexible Data Schemas

Cloud-Native SDI: Schemas & ID's part 2

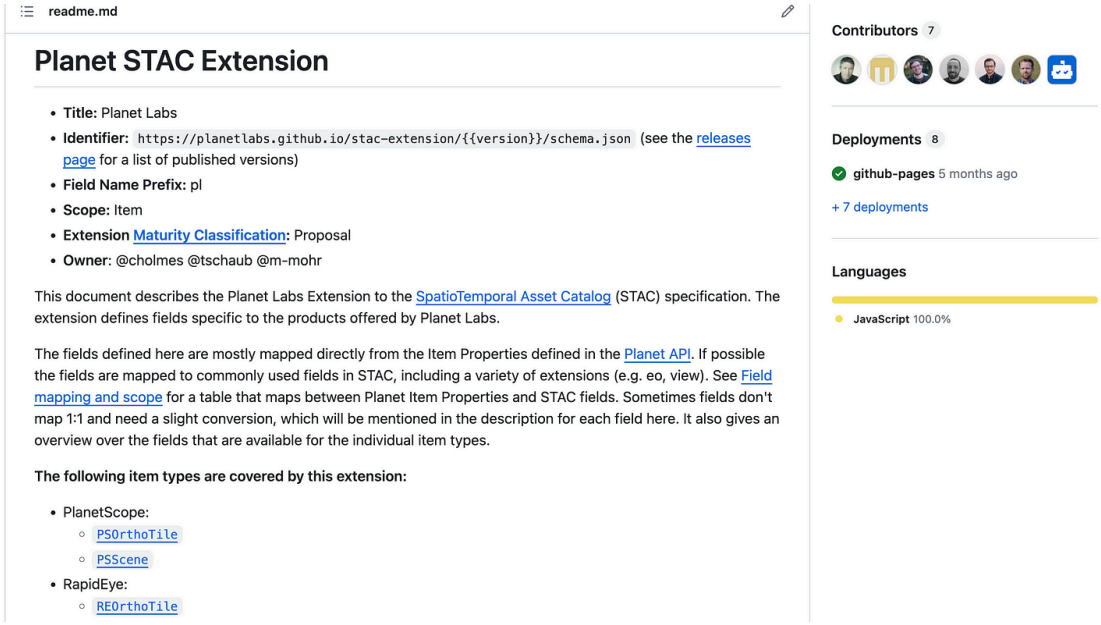
So as I mentioned in Part 1 of this series, the next big interest for me is common data schemas. I believe the foundation we've been laying with Cloud Native Geospatial formats has the potential to lead to much greater interoperability between data, if we can get a few things right.

Flexible Schemas

The first of these is being able to flexibly combine different data schemas. The potential that feels 'different' than past attempts in the geospatial world is getting away from XML, particularly the way validation works. In the XML world you'd use an XML Schema to define how each field should work and then the same XML Schema could validate if the fields in your document had the proper values. The big problem was that if an extra field was added then the validation would fail. You could of course extend the XML Schema definition, but it wasn't easy to 'mix and match'—to just have a few different XML Schemas validate different parts of your data.

The cool thing is that with JSON Schema you can easily do this. 6 different JSON Schemas can all validate the same JSON file, each checking their particular part. I've not yet dug deep into the options for validating Parquet & GeoParquet, but I've seen no indication there's anything that will break there either if the data has an extra column. We've used this to great effect in STAC—the core STAC spec only defines a few fields, and includes the JSON Schema to validate the core. But each 'extension' in STAC also has its own JSON Schema. Any STAC Validator will use each extension's schema to validate the whole file.

This has profound implications, since it enables a much more 'bottom up' approach to the evolution of the 'data schema'. In the XML world you needed everyone to agree, and if someone disagreed they'd need to fork the XML Schema and redefine what they wanted. Users would then need to pick between which validation they wanted to use. So it was really important that a top down entity set the standard. With the bottom up approach anyone can define just a few fields for their own use, and others can define similar things in their own way. Of course the core 'thing', like SpatioTemporal Assets, needs to be done well, but I believe the key here is to make that core as simple as possible, so many extensions can thrive. Then it's just real world usage that decides which fields are important. And it also allows 'incubation'—a single organization can just decide to make their own validator for their fields. An example of this is Planet, with the Planet STAC Extension:



github.com/planetlabs/stac-extension

It uses a bunch of the common extensions like `view`, `eo`, `proj` and `raster`. And then it has some fields that are very specific to Planet's system (`item_type`, `strip_id`, `quality category`), but there are a number of fields for which there is not yet a STAC extension. I say 'yet' because these are fields that are likely things that other satellite may have for metadata, like `clear_percent`, `ground_control` and `black_fill`. They can be defined for Planet's validation, since their users expect them. But others in the community can also look at Planet's and decide to adopt what they did. And if a few different providers all do something similar we can come together and agree on a common definition that we'll all use. If two people decide to define the same 'thing' and both feel theirs is right then both can exist, but likely one will gain more adoption and become the standard.

Global Datasets

One of my main observations from the last twenty years of standards is that having more data following the standard is key. Whatever dataset is the largest and most important in an ecosystem usually becomes the standard way of doing things, even if it didn't set out to be a standard. Key to STAC's success was that early on both Landsat and Sentinel 2 were available in the standard, and indeed it enabled those two major datasets to be more interoperable with each other.

With the explosion of satellite imagery and the continued advances in AI and computer vision we're seeing more datasets that are truly global. The best of these will play a major role in setting the standard data schemas for whatever type of data they represent. Indeed I think we'll see an interplay between aligning schemas for validated training data about foundational geospatial data types and the schemas of the resulting models. And hopefully we'll see major governmental data providers look to embrace their role in setting standards—a federal

government defining a reusable schema for X, putting foundational data out in it, and also encouraging each state to use the same schema.

Overture Maps is also doing really great work in building open global datasets for some of the most foundational geospatial layers. And they are taking their role in setting a data schema standard seriously, working to make it a flexible core that other attributes can be added to.

Cloud-Native Geo Formats Require Deeper Alignment

I believe one of the main mistakes of past SDI efforts was to try to punt the hard problem of getting people to align their data. Their message was that everyone could keep their database in its same schema, and the application servers delivering the API's could just transform everything into standard schemas on the fly. This proved to be incredibly annoying to get right, as being able to map from anything to a complex data schema isn't easy, and the tools to help do this well never really took off.

I think the fact that a Cloud Native Spatial Data Infrastructure is fundamentally based on formats instead of API's means that it will force people to confront the hard problem of actually aligning their data. We *should* be trying to get everyone actually using the same data schema in their day to day work, not just doing their internal work in one schema and transforming it into another schema to share it with others. It'll be much easier if you share the buildings file from the city of Belém and its core fields follow the same definitions as Overture. We shouldn't have to go through WFS servers just to share interoperable data, our goal needs to be making the actual data interoperable.

Obviously its unrealistic to expect everyone to just change their core database to a new schema. But it's easier to use an ETL tool or a bit of code to actually transform the data and publish it than it is to set up a server and define an on the fly schema mapping against its database. And if we can get small core schema definitions with easy to use extensions then we won't need to convince Belém to drop their schema and fully adopt 'the standard'—they should be able to update a couple core fields and define their own extensions that match their existing data schema, and slowly migrate to implementing more of the standard extensions.

Riding the Wave of Mainstream Data Innovation

The other thing we are starting to do is to align with all the investment in mainstream data science and data engineering. One of the ways the founders would explain Planet in the early days was that they were leveraging the trillions of dollars of investment that has gone in to the cell phone. Other satellites would buy parts that were 'made for space', and were egregiously expensive because they were specially designed, with no economies of scale. Planet bought mostly off the shelf components, and was able to tap into the speed of innovation of the much bigger non-space world.

By embracing Parquet we're starting to do the same thing with geospatial. I think the comparison is apt—we've tended to build our own special stacks, reinventing how others do

things. Open source geospatial software has been much better, like with PostGIS drafting off of PostgreSQL. But there is now huge investment going into lots of innovation around data.

In the context of data schemas, there are many people looking at data governance, and tools to define and validate data schemas. I've not yet gone deep into these, but we should be able to tap into a number of existing tools to do what we want with Parquet, instead of having to build all the tools from scratch.

Making it work

So I think there's some relatively easy things we can do to usher in an era of bottom-up innovation in data schemas. This should lead to much greater collaboration, and hopefully start a fly wheel of Cloud-Native Spatial Data Infrastructure participation that will lead to a successful global SDI.

I think the key is to make it easy to create simple core schemas with easy to define extensions on top of the core. This mostly means creating a core toolset so anyone can create a schema, translate data into the schema, and validate any data against that both that schema and extensions.

The cool thing is that I think STAC has defined a really good way to do this, that just needs to be generalized and enhanced a bit.

Generalizing the STAC way

So STAC is ready to use if you want a data schema for data where the geometry is an indicator of the footprint of some other type of data, and there are links to the actual data. And then you can tap into all sorts of STAC extensions that help define additional parts of a flexible data schema. But if your data is like the vast majority of vector data, where the geometry and properties *are* the data, not metadata about some other data, then you can't tap into all the great extensions and validation tools.

To generalize what STAC does I believe we can build a construct that lets any type of vector data define a core JSON Schema and links to extensions. With STAC you just look for `stac_version` and then you know that it can validate against the core STAC extension, and then `stac_extensions` is a list of links to the JSON schemas of the extensions it implements. The links are naturally versioned, as part of the URL of the schemas.

A general version could just have a `definition` that links to a single core schema (validating the geometry and any other attributes that are considered 'core'). And then an `extensions` list that works the exact same way as STAC extensions. It perhaps could even directly use some STAC extension definitions like the MGRS extension, which could be used by any number of vector datasets that want to include MGRS:

Item Properties

Field Name	Type	Description
mgrs:latitude_band	string	**REQUIRED**. The latitude band of the Item's centroid
mgrs:grid_square	string	**REQUIRED**. MGRS grid square of the Item's centroid
mgrs:utm_zone	integer	The UTM Zone of the Item centroid

Between 80°S and 84°N UTM Zone is required. In the polar regions UTM zone is not provided.

Close readers will likely note that this approach would all unfortunately be incompatible with STAC, since STAC has hard coded versions. But I think if a wider ecosystem takes off in a big way we could consider a STAC 2.0 that fits properly into the hierarchy. And there's probably some less elegant hacks you could do to make it all work together if that was needed.

The other bit that would make a ton of sense to generalize is the STAC extension repository template. This to me is one of the most clever parts of the STAC ecosystem, and it's all thanks to Matthias Mohr.

README.md

Template Extension Specification

- **Title:** Template
- **Identifier:** https://stac-extensions.github.io/template/v1.0.0/schema.json
- **Field Name Prefix:** template
- **Scope:** Item, Collection
- **Extension Maturity Classification:** Proposal
- **Owner**: @your-gh-handles @person2

This document explains the Template Extension to the SpatioTemporal Asset Catalog (STAC) specification. This is the place to add a short introduction.

- Examples:
  - Item example: Shows the basic usage of the extension in a STAC Item
  - Collection example: Shows the basic usage of the extension in a STAC Collection
- JSON Schema
- Changelog

Packages

No packages published
Publish your first package

Contributors 6

Deployments 5

github-pages 2 years ago

+ 4 deployments

https://github.com/stac-extensions/template

The core is it gives you a clear set of guidelines to fill out your own extension. You don't need to check 3 other extensions to see how they do it, you just change the right places for yours and it then 'fits' with the ecosystem. But it goes far beyond that, as it clones a set of continuous integration tools. It will automatically check your markdown formatting, and once you finish your JSON Schema it will also check that all your examples conform to STAC and your defined extension.

And then when you publish a release it will automatically publish the JSON Schema in your repo on github pages, to be the official link. And then STAC validators can immediately make use of it. So to create any new version of your extension you just need to cut a release. I hadn't even known that anything like this was possible, but it made it such a breeze to create a new extension. You can focus on your data model, and not on how to release it and integrate into tooling, since it all 'just works'.

Enhancing the STAC schema toolset

So I think there's a few ways we'd ideally go beyond just generalizing how STAC does things. The first one is to be compatible with GeoParquet. GeoParquet is a much more naturally default format for vector data on the cloud than GeoJSON is. It worked well for STAC, particularly because we had both the static STAC and the STAC API options. I originally imagined that large data sets would naturally be stored in a database and use a server that clients would query. But the fully cloud native approach has been quite appealing, and a number of very large datasets are just on object stores, and consist of millions of individual JSON files (next to the actual data files).

We've recently started to standardize on how to represent a full STAC collection in GeoParquet with the STAC GeoParquet Spec. I'll hold off on a deep dive on that, but it's pretty cool to be able to just query the entire STAC catalog without needing an API.

So for non-'asset' data, where the geometries and properties are the data, not metadata, GeoParquet makes much more sense than GeoJSON as the main distribution format. But my hunch is that it likely will still make sense to define data schemas in nice human-readable JSON Schemas. I need to learn more about what types of validations exist in the Parquet world, and maybe there's some awesome set of tools that fit all our needs, but a quick scan didn't reveal a lot. So I suspect we'll still want to do STAC style definition of JSON Schemas, but perhaps extend the validation tooling to be able to validate GeoParquet files using a set of JSON Schemas. Parquet does use JSON in its metadata definitions, so I think there are likely some compatibilities to leverage. But some smarter people than me will need to dig deeper, and my hunch may well be wrong. Or maybe we'll abandon GeoJSON and JSON Schema entirely.

There is some argument for defining the core schemas completely abstractly, in something like UML, since formats will continue to change and we should be adaptable to that. But from my experience that introduces an unnecessary layer of abstraction. With STAC I actually started a SpatioTemporal Asset Metadata (STAM) spec, see https://github.com/radiantearth/stam-spec, to try to make abstract definitions that could map to JSON but also other formats (like GeoParquet though it didn't exist yet, or as Tiff tags in a GeoTIFF). But it was a pain to try to maintain both and just didn't add much. We do have an alternate instantiation with STAC GeoParquet, but it feels totally fine to have the JSON be canonical. And if JSON fades in 10 years I'm sure we could translate the core definitions into whatever new thing is the next trend.

Just as I was nearing the end of my first draft I decided to check Overture and was quite pleased to find docs.overturemaps.org/#key-specifications where their first point is that they will

document everything in JSON Schema first. So nice to be thinking the same way. They say the final format for Overture deliveries hasn't been set yet, but the most recent releases were GeoParquet. Unfortunately they then endorse sillyCamelCase, so we'll have to be mortal enemies—snake_case_forever! ;)

I think there's likely lots of other ways to really enhance things, making higher level tools that make it even easier to define a data schema without having to figure out how to right JSON Schemas. And tools to create compelling applications with ease that rely on solid schemas.

Measuring Success

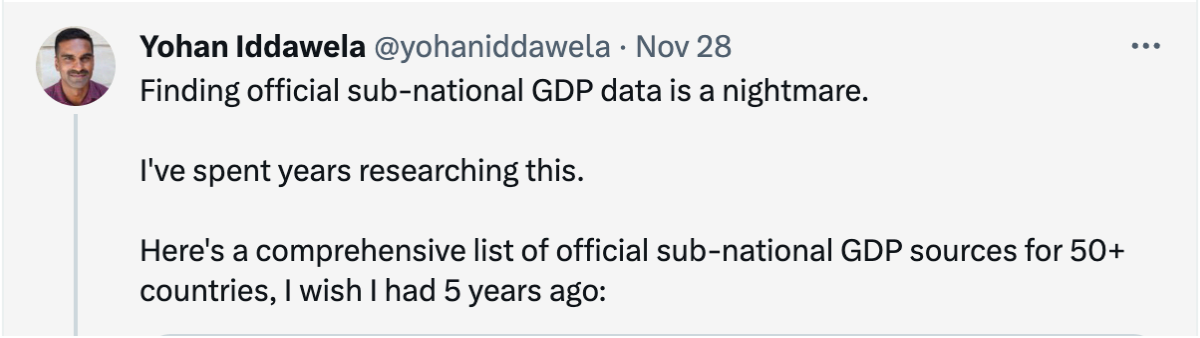
One final idea to generalize and hopefully really enhance is STAC Index. STAC Index provides a list of all public STAC Catalogs, and you can use a STAC Browser to easily browse the full extent of any of them. One of the original ideas of STAC Index was to crawl all the catalogs and provide lots of interesting stats on them. Tim Schaub built a crawler to do this and reported on the results in the State of STAC blog post, but ideally it would be a continuous crawling and reporting of stats.

I believe the key to a Cloud-Native Spatial Data Infrastructure is to make it really easy to measure how successful adoption has been. Not just in total of number of GeoParquet datasets, but to be able to get some real nuance. Things like number of rows in GeoParquet, stats by particular data schemas, global data coverage by data type, etc. I think we'll still use STAC, but likely just at the 'collection' level, to provide metadata on GeoParquet files. So STAC Index should be enhanced to really be the 'Cloud-Native SDI Index', and help be clear KPI and scoreboard to really measure adoption. I should probably spend a whole blog post on the topic of measuring standard adoption at some point, as I have a suspicion that making it really easy to measure the actual adoption is a powerful level to drive adoption.

Let's do this!

So my next major project is to try to pick one or two foundational data sets and just try to make a flexible data schema for them. One obvious one is buildings, and Overture is doing incredible work there. They're truly crushing it on figuring out a core, flexible data schema with global ideas, but I think it'd be awesome to build on what they're doing in two ways. The first is to try to build validation tooling, and experiment with making other building datasets 'overture compatible'. And the second would be to try out 'extending' their core schema with some additional fields. Perhaps something like building color, or roof type. I suppose the ideal would be to find some better attributed building dataset and make it Overture compatible—try to conflate the geometries, merge the common attributes, and make a schema for the extended attributes.

I just flipped to twitter and read this cool post on sub-national GDP.



This does seem like another opportunity—use Overture locality schema for sub-national boundaries at the core but add an attribute for GDP, and harmonize all the datasets he listed.

Overture has been doing such solid work on the core that I think it would be nice to do a parallel effort on some type of data that isn't in their core, to be able to try different approaches, compare and hopefully cross-pollinate. The other data set I'm really intrigued by is agricultural field boundaries, and a flexible schema for properties of the field. <If anyone has interest in working on this get in touch—I'm working with Taylor Geospatial Engine to try to bring together a bunch of experts to make some real progress here, and to hopefully build out some of the core tooling to enable more of this>.

I'd also love to hear of other opportunities for collaborations on data schemas, and other success stories where there are interoperable data standards. I think it'd be interesting to try to adapt some of the successful ones into cloud-native formats, just to see if it works and if it adds value. So if anyone wants to work on that don't hesitate to get in touch.