Multi Channel Data Architecture

This is the data architecture for the data channels such as the Utility Emissions Channel of the Operating System for Climate Action.

The Idea

Writing software requires getting data from external sources, which usually means accessing an API by sending a GET or POST request to a URL with authentication keys and a request message and parsing the XML or JSON that comes back. While this approach is fully automated, it comes with a lot of potential problems:

- You have to build a different API for every source.
- The API is periodically unavailable, so you’re down as well.
- The API could change and break your app.
- The API could simply be taken away from you, to "upgrade to a better API", to "improve security", etc. that leaves you scrambling for an alternative.
- You have to store a local copy of the data retrieved from the API, in case it goes away.
- You had to work out a procedure for audit and verification of the data obtained from the API.
- You worry about security because now you have a big app with lots of data and authentication keys.

As a result, these API integrations are expensive and time consuming to build and maintain, so they don’t scale well in scope. When we’re talking about building a Hyperledger-based blockchain app for Carbon Accounting and Certification WG, which could require integrating data from a large number of sources, we came upon a better way to do it with the blockchain itself.

Instead of each source providing us with an API, they could put their data on the blockchain. This could be a Hyperledger permissioned channel that is only open to their customers and other trusted parties. The data could be the carbon emissions of a product or a particular invoice, for example.

Instead of a big app consuming data from all these sources, an app could simply traverse through multiple channels. Instead of having a single access key from each data source, it could request that the user of the app pass it their own authentication key or token. Then it would pass that token through to the data source's channel and obtain the data it needs. It would then perform the calculation with this data. Finally, it could open a channel of its own and put the results on this channel. This channel could then be open to its customers, so they could in turn use the data for their own carbon emissions calculations.

An additional advantage is that the sources contributing the data simply have to store the data in the channel. They do not have to be responsible for its long-term keeping and availability. Thus, this allows specialty companies, such as CO2 emissions auditing companies, to audit the data from a larger entity, such as a utility, shipping company, airline, or manufacturer, and then store it in the channel for everyone else to use. This is similar to the offline scenario where a building engineer files a report for a building developer with the city’s building department: Once the report has been filed, the building engineer needs no longer to be responsible for making it available. The city planning department is responsible for making it available to the building developer and his sub-contractors and customers, as well as other government agencies.

There could be many channels of data from electric utilities, shipping companies, airlines, and suppliers of industrial commodities such as plastics or steel, each placed on the channel by a trusted CO2 emissions auditing company. Our app is a carbon calculator. The user of our app would pass in his keys to access the channels, for example from his utility emissions channel, based on his status as their customer. We would pass those keys to the channel and get the emissions from the utility, which we could then use for our calculations. The entire process is run as a serverless micro service.

The advantages of this approach are:

- The auditing company doesn’t have to worry about keeping an API up and running. Once it has done its analysis, it’s written to the blockchain channel and no longer requires a server to provide it, just like web content deployed on a CDN.
- The format is standardized by agreement between the provider and users of the data.
- The content of the data cannot be altered once it is written to the blockchain, removing concerns about audit.
- Even if the auditing company changes the format or stops providing data in the future, data from the past will always be available.
- The app consuming the data can be much smaller. For example, instead of storing an authentication key for many sources and data from many users, it can use keys provided by a customer’s wallet. Once it’s finished with the data obtained from those keys, it can put the results on its channel and then delete the data obtained from the blockchain.
- Both the data source and data consumer could then be run as serverless micro services.

The channels could be maintained by a third party entity, which charges a small fee for each access through a token. The data source provider could give out these tokens when it allows a data consumer permission to access its channel. In return, the third party entity serves as a neutral party tasked with keeping up the operations of the data channel.
This general architecture could be used for a variety of business data integrations, specifically the ones from Operating System for Climate Action.

See slides illustrating this.

**A Physical Analogy**

A physical world analogy is engineering reports filed with city building departments. There are many engineering companies who would draw up building plans according to building codes. These are reviewed, approved, and filed by the city building department, and then they are available to owners of the building as well as government agencies. Here we are proposing to replace the equivalent of the centralized city building department with a blockchain hosted and run by the engineering companies themselves.