openIDL - Architecture - Tenets and Decisions

This page describes the decisions we have taken and the tenets that drive them.

Table of Contents

- Architecture Tenets
  - The System Must be Manageable
  - The System Must be Cloud Agnostic Whenever Possible
  - Infrastructure as a Service before Self Managed Infrastructure
  - The System Must be Transparent
  - The Privacy of the Data Distributed Nodes is Paramount
- Architecture Decisions
  - SECURITY
  - DATA ARCHITECTURE
  - APPLICATION DEVELOPMENT
  - DEV / OPS

Architecture Tenets

Here we discuss the tenets of the architecture. Which things are most important and why. When we make architecture decisions below, they take these tenants into account.

The System Must be Manageable

We will have a distributed network using distributed ledger technology in the guise of Hyperledger Fabric. The surrounding components such as APIs, UI Applications, Databases, Extraction Patterns and Data Transformation and Enrichment must all be managed. That is, when we have the need to add some new functionality, we want all the participants to have the necessary components. To make this work, the node must be manageable. For this purpose we intend to use GitOps, where the configuration is managed by a configuration held in git which is used to update the node components automatically.

The System Must be Cloud Agnostic Whenever Possible

The openIDL components will most often be deployed to the cloud. While we will use AWS, other participants may choose another cloud provider. For this reason, we want the deployment to work in any one of the possible clouds without undo alteration or duplication. There will be specific areas where the cloud provided services must differ. In these cases, we will use a layer to normalize the interface if possible. We must always be careful to balance cost and complexity with flexibility.

Infrastructure as a Service before Self Managed Infrastructure

Whenever possible when there is a choice between implementing infrastructure that we manage and that which is managed automatically by the cloud provider, lean toward cloud provider managed solutions.

The System Must be Transparent

The use of Infrastructure as Code will make the system's configuration self documenting. Anywhere this does not fully explain, extra documentation must be provided. This is generally found in the README.md files in git.

The Privacy of the Data Distributed Nodes is Paramount

The data made available to the openIDL system must have privacy managed by the data owner. The owner must control all "access" to the data and be confident that no data is shared with other nodes without their knowledge.

Architecture Decisions

SECURITY

SUBJECT	Secret Management
STATUS	Open

DECISION	The secrets are held in: GitHub Secrets and Vault The secrets are managed by: The secrets are accessed from Iac …?
DISCUSSION	The management of secrets is complicated. Below are some requirements for the solution. If we can tick off all these, we'll have a winner. Must be able to manage: - carrier secrets - api keys - aais secrets - common secrets - cloud provider secrets - database secrets - hlf network secrets like certs - application secrets - distributed secrets Must: - rotate passwords - be encrypted - permissioned so only visible to specific individuals or ci/cd - manageable - update / delete / create / view - auditable - know what changed and that no breaches have occurred - be accessible from IaC - terraform - be accessible from IaC - helm - be accessible during CI/CD - be cloud agnostic for use - be multi-cloud - have a health check of the system - at startup and intervals - provide logging and notifications of updates - exhibit CIA - confidentiality, integrity, access - have a user interface for managing the secrets Options: - tools o vault o aws secrets manager -

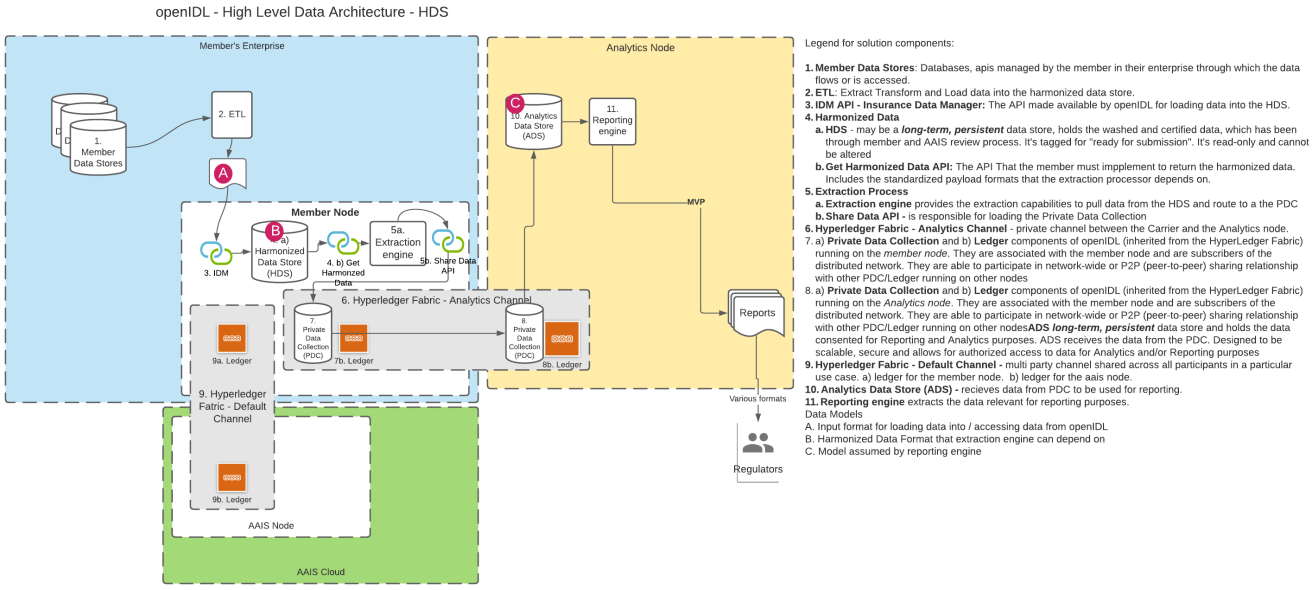
SUBJECT	Automation of Hyperleger Fabric Network Setup
STATUS	Open
DECISION	Use Blockchain Automation Framework (BAF)

DISCUSSION	BAF will be used to set up the network automatically. BAF will run on a pod inside the kubernetes cluster so it has access to the required credentials and certificates that are stored in Vault. The Vault instance is running inside the private cloud, so the automation cannot run from GitHub actions.

SUBJECT	User Authentication for Application Access
STATUS	Open
DECISION	User Authentication is Platform Specific or can it use Okta
DISCUSSION	The authentication of users must be cloud specific for access to applications because there is no generic authentication provider. - start with aws strategy - cognito - want to offload identiy to identity provider - can we use okta as the main identity management and link it to the underlying provider thus acting as a common api for the applications?

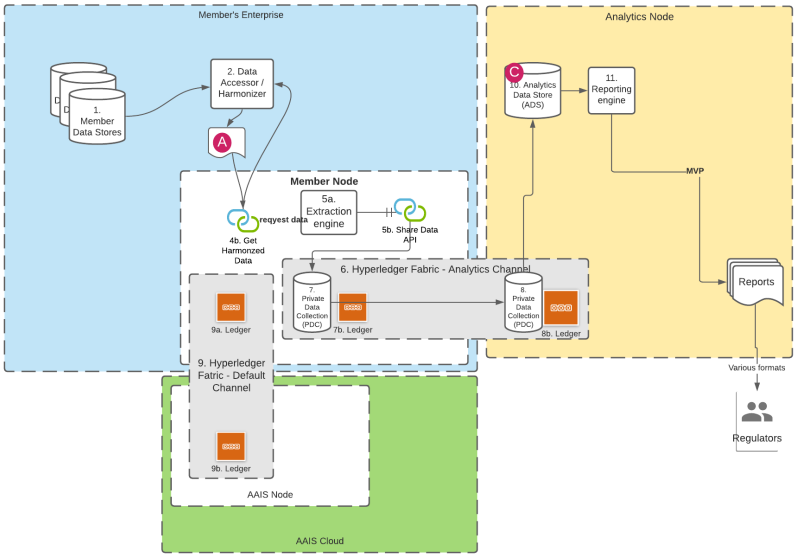
DATA ARCHITECTURE

A. Using the HDS DB



B. Using only the API

openIDL - High Level Data Architecture - No HDS



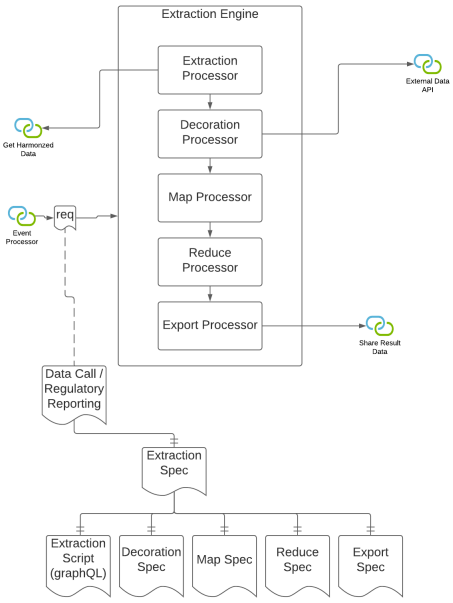
Legend for solution components:

1. **Member Data Stores**: Databases, apis managed by the member in their enterprise through which the data flows or is accessed.
2. **Data Accessor / Harmonizer**: Respond to Request from api to get harmonized data.
3. **N/A - removed IDM**
4. **Harmonized Data**
   a. **N/A - removed HDS**
   b. **Get Harmonized Data API:** The API That the member must impplement to return the harmonized data. Includes the standardized payload formats that the extraction processor depends on.
5. **Extraction**
   a. **Extraction engine** provides the extraction capabilities to pull harmonized data route to the PDC
   b. **Share Data API** - saves data into the Private Data Collection
6. **Hyperledger Fabric - Analytics Channel** - private channel between the Carrier and the Analytics node.
7. a) **Private Data Collection** and b) **Ledger** components of openIDL (inherited from the HyperLedger Fabric) running on the *member node*. They are associated with the member node and are subscribers of the distributed network. They are able to participate in network-wide or P2P (peer-to-peer) sharing relationship with other PDC/Ledger running on other nodes
8. a) **Private Data Collection** and b) **Ledger** components of openIDL (inherited from the HyperLedger Fabric) running on the *Analytics node*. They are associated with the member node and are subscribers of the distributed network. They are able to participate in network-wide or P2P (peer-to-peer) sharing relationship with other PDC/Ledger running on other nodes**ADS *long-term, persistent*** data store and holds the data consented for Reporting and Analytics purposes. ADS receives the data from the PDC. Designed to be scalable, secure and allows for authorized access to data for Analytics and/or Reporting purposes
9. **Hyperledger Fabric - Default Channel** - multi party channel shared across all participants in a particular use case. a) ledger for the member node. b) ledger for the aais node.
10. **Analytics Data Store (ADS) -** recieves data from PDC to be used for reporting.
11. **Reporting engine** extracts the data relevant for reporting purposes.

Data Models
A. Input format for loading data into / accessing data from openIDL
B. Harmonized Data Format that extraction engine can depend on
C. Model assumed by reporting engine

- DA - Extraction Processing
- DA - Harmonized Data Scope
- DA - Harmonized Data Loading / Normalization
- DA - Harmonized Data Format Governance
- DA - Harmonized Data Store
- DA - Harmonized Data Access
- DA - Harmonized Datastore DBMS Implementation
- DA - Harmonized Data Model
- DA - Export Data Model

SUBJECT	**DA - Extraction Processing**
STATUS	Open
DECISION	TBD

DISCUSSION	 Extracton Engine • the extraction engine is called from the ***Event Processor*** as a result of some event like a consent or a data call / regulatory report due event. The payload in the request is the specification. • accesses data through the ***Get Harmonized Data*** API • ***Extraction Processer*** executes ***extraction spec (graphQL)*** against the API • the result of the API call is passed onto the ***Decoration Processor*** • ***Decoration Processor*** reaches out to external data providers to get corrolated data (like census or flood etc) based on the ***decoration spec.*** • Resulting decorated data is passed onto the ***Map Processor*** which uses the ***map spec*** to map data to keys and remove unneeded data • The ***Reduce Processor*** reduces the mapped data down to the aggregated data needed for the report using the ***reduce spec***. • The ***Export Processor*** calls the ***Share Result Data*** API to make the data available for reporting. The format of this data is defined in the ***export spec.*** In openIDL when a data call (or a stat reporting) is "consented to" by the carrier, the data must first be accessed from somewhere and th (usually a regulator) can access or be sent. The transformation of the data from its "harmonized" state to the result state is called the "extraction", "extraction pattern" or "extraction Since accessing the data can take multiple forms (see other architecture decision "Harmonized Data", there is some variability in this de We can assume that the data being accessed for the extraction is "harmonized", meaning for every execution of the extraction on a sing of the data are known and consistent. Creation and Management of extractions can be organization specific. For stat reporting, this is the stat-agent (such as AAIS), for data In either option from the diagrams above, the extraction processor will access the data through an api instead of accessing a database It is proposed that graphQL be considered as the language used to access for extraction and summarization. The extraction processor similar. This means the extraction is more than just a data access. See the diagram above for how this component may be architected.

SUBJECT	**DA - Harmonized Data Scope**
STATUS	Proposed
DECISION	Proposal: data format / schema will be standardized across nodes, for a given use case
DISCUSSION	The data available for extraction must be normalized for multiple extractions across multiple use cases across multiple members. Is this one single model? Is the data at rest in the same model as the data in motion?

SUBJECT	**DA - Harmonized Data Loading / Normalization**
STATUS	Proposed
DECISION	Proposal: If the HDS is at rest, the loading of that data is the responsibility of the member owner of the node. If the HDS is an API, the maintenance of that API is the responsibility of the Technical Steering Committee and the mapping to other data sources is the responsibility of the member owner of the node.

DISCUSSION	1. Loading data will be via an API, IDM or ...? Will a direct SQL load be allowed? This will use the ingestion model. 2. We should consider using graphQL as the extraction processing language or part of the extraction processing component to replace the current map reduce "extraction pattern"

SUBJECT	**DA - Harmonized Data Format Governance**
STATUS	Proposed
DECISION	Proposal: data schema, enumerations and the data dictionary will be standardized, and endorsed by the RRSC (and other groups per use case)
DISCUSSION	The Technical Steering Committee, Regulatory Reporting Steering Committee and the Data Model Steering Committee are all possible owners of this.

SUBJECT	**DA - Harmonized Data Store**
STATUS	Proposed
DECISION	Proposal: 1. HDS can be persistent or transient. It's a member's decision. See next 2. HDS can be persistent and can be used by member's as a "warehouse on the edge" for sharing data via openIDL 3. Either way, the member is responsible for configuring the API that accesses the data.
DISCUSSION	The data available for extraction must be normalized for multiple extractions across multiple use cases across multiple members. Is this one single model? Is this one database? Is this data at rest and/or available through an API

SUBJECT	**DA - Harmonized Data Access**
STATUS	Proposed
DECISION	Proposal: 1. All access to "Harmonized Data" is through an API 2. Member is responsible for the quality of the data retrieved and for certifying that a "request" for the data is supported.
DISCUSSION	If we determine that a standing harmonized store is not required, then we must establish an API with a standardized payload format that can be used to access the data. The member must "certify" that the data is available and quality in order to consent to a data extraction. The consent and certification can be captured on the ledger. The call to the API will come from the extraction processor. The extraction processor can run on the member node. Can the extraction processor run on the Analytics Node? If the extraction runs outside the member node, how does this work? Can it call the API directly? Must we use HLF to "transport" the data?

SUBJECT	**DA - Harmonized Datastore DBMS Implementation**
STATUS	Open
DECISION	*Proposal:* *If the HDS is a physical db inside the node, then the HDS DBMS must support our chosen access language (graphQL?)* *HDS will be a relational database. It cannot be a noSQL, graph, document DB etc.* technical implementation of a HDS is non-prescriptive i.e. it can be MySQL, MS SQL, Oracle etc.

DISCUSSION	If data is at rest in the harmonized datastore, what is the technology? Does it need to be a single dbms? Should it be noSQL? Can it just be an interface?

SUBJECT	**DA - Harmonized Data Model**
STATUS	Open
DECISION	??
DISCUSSION	The data available for extraction must be normalized for multiple extractions across multiple use cases across multiple carriers. The data must be produced by the carrier from their original sources. Is this an ETL process? What is the format of the load data? Is it the schema of a standing HDS database or is it a messaging format? Is there just one "in transit" model or are there different ones for different contexts? Contexts might be the data call, the line of business, etc.

SUBJECT	**DA - Export Data Model**
STATUS	Open
DECISION	Proposal: The export data model is specific to each use case and must be specified in that data call / regulatory report
DISCUSSION	The extraction process results in an export of data in an agreed format. The format of this must be defined as part of the specific data call or regulatory report.

APPLICATION DEVELOPMENT

SUBJECT	Common UI Code Management
STATUS	Open
DECISION	Single Application Angular UI Variations will utilize angular libraries
DISCUSSION	The library will be a different kind of angular app located in the same super library as all apps that use it. This is the approach for the data-call-ui. (openidl-ui and openidl-carrier-ui) For common / shared libraries we will use an npm registry.

DEV / OPS

SUBJECT	Local Kubernetes Development
STATUS	Open
DECISION	Use Minikube for Local Kubernetes Runtime
DISCUSSION	There are multiple options for local kubernetes deployment. We chose Minikube over Kind because of it's simplicity.

SUBJECT	Infrastructure as Code
STATUS	Open

DECISION	Use a combination of solutions depending on application.
	
	▪ Cloud
	  ▪ Terraform for Infrastructure Provisioning in the Cloud
	  ▪ GitHub actions for CI/CD and execution of Terraform
	  ▪ Ansible for Deploying ??
	  ▪ Helm for managing Kubernetes
	  ▪ Flux for provisioning and managing distributed nodes
	▪ Local Reference
	  ▪ Bash scripts for provisioning local

DISCUSSION	Provide options for selection upon setup. ▪ Terraform Cloud ▪ Terraform Enterprise ▪ GitHub Actions ▪ Manual All provisioning artifacts are managed in git The customer will have a github / gitlab account that is private to them. We may or may not have access to that repository. To accept updates, the customer will accept a merge/pull request into their repository with our changes. That update in git will automatically trigger the workflow. The workflow may allow automatic provisioning or require an acceptance from the customer. - milestones 1. start with github actions, forked repos and manual execution 2. move to terraform cloud for aais node target will have two options for node owners ▪ use terraform cloud ▪ use terraform on-prem (terraform enterprise)

SUBJECT	Infrastructure as Code
STATUS	Open
DECISION	Use Terraform to provision cloud specific services
DISCUSSION	

SUBJECT	Infrastructure as Code
STATUS	Open
DECISION	Execute Terraform using GitHub actions
DISCUSSION	If possible use GitHub actions - see above for options If there is some reason to use cloud specific - cost of implementation - complexity etc

SUBJECT	Infrastructure as Code
STATUS	Open

DECISION	Use Flux v2 for Deployment of Kubernetes artifacts
DISCUSSION	This technology enables GitOps in build and deployment

SUBJECT	Infrastructure as Code
STATUS	Open
DECISION	Use Helm Charts for Application and Network provisioning in Kubernetes
DISCUSSION	Helm is a very popular way to provision Kubernetes clusters

SUBJECT	DevOps
STATUS	Open
DECISION	Publish Common Libs as images to NPM Registry in GitHub
DISCUSSION	Any common components should be packaged as images and published to the GitHub packages.

SUBJECT	DevOps
STATUS	Open
DECISION	Images should be published in the GitHub packages container registry
DISCUSSION	Since we separate building of images from their deployment, we can build the images into the registry and then refer to that registry when deploying

SUBJECT	Secrets Management
STATUS	Open
DECISION	secret management should be cloud agnostic
DISCUSSION	Notes - the secrets for the each cloud might be managed differently - hashicorp makes a popular opensource solution called vault - if cloud specific, we should have a layer that normalizes the access of secrets so the scripts / config files don't need to change from cloud to cloud

SUBJECT	Secret Management
STATUS	Open
DECISION	Secrets are applied during deployment, not in the image
DISCUSSION	The images used to create the pods in Kubernetes should not contain any private information. This can all be applied during deployment by mounting the file from a secret held outside.

SUBJECT	MongoDB

STATUS	Open
DECISION	The Harmonized Data Store will be deployed inside kubernetes
DISCUSSION	The best practice regarding databases and Kubernetes is to host them outside. As long as the db is mongo and has a uri accessible to the insurance data manager and other apis, it is viable. The terraform to set it up may need different flavors for the different clouds.

SUBJECT	UI Deployment
STATUS	Open
DECISION	The UI will be deployed inside kubernetes
DISCUSSION	There are two main choices for deploying the ui. Here is the discussion about the relative merits for the options.

Item	Cloud Specific	Cloud Agnostic
How	Using S3 and other AWS specific technologies	Deploy as pod inside Kubernetes
Performance	Very good performance	Less performant
Availability	The UI itself is more available, but the api isn't any more available	The UI is subject to the same availability as the API
Cost	Very inexpensive	More cost, TBD
Scalability	Infinite scalability, subject to API	Not as scalable, but good
Complexity	More complex for multi-cloud	Less complex for multi-cloud
Managability	More difficult to manage in remote nodes	Less complex for remote nodes

We deploy the applications inside kubernetes so they are more manageable. This includes the APIs and the UIs. Deploying at the edge is a best practice, but manageability is more important in this case. We can deliver updates to the code as images in a container registry and have them deployed much easier than if we used AWS (or other cloud) specific services.

Q: Why not have a centralized UI?

A: The UI is configured to access the API. iI has to have private access to the API inside the node, not go out onto the internet and have the apis exposed publicly. The apis are private to the member cloud, actually private to the kubernetes cluster.

Because manageability is a very high priority item for the ui components, this outweighs the differences in other aspects.

SUBJECT	Channel Policy
STATUS	Open
DECISION	The Channel Policy will be set to ANY with a specific role required to allow new organizations to join the network
DISCUSSION	The channel policy controls how new organizations are joined to the network. If set to Majority, many of the participants on the network must approve new organizations. If set to Any, then just one is required. We will create a role of Admin which will be required by the policy for any organization to approve new organizations.