EU Cloud Accelerator Bergamo

Technical Workshop

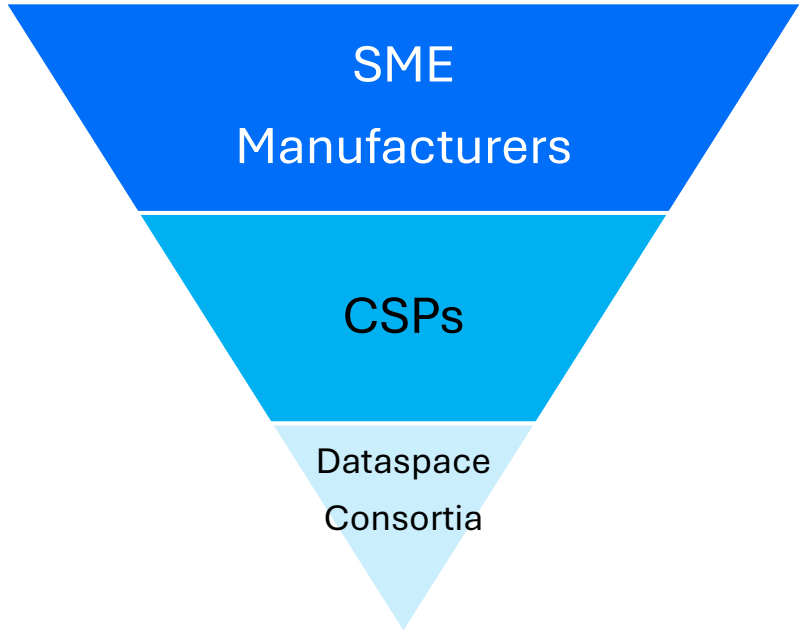
October 6-7, 2025

Developing and Deploying Hosted Dataspace Services

Introduction

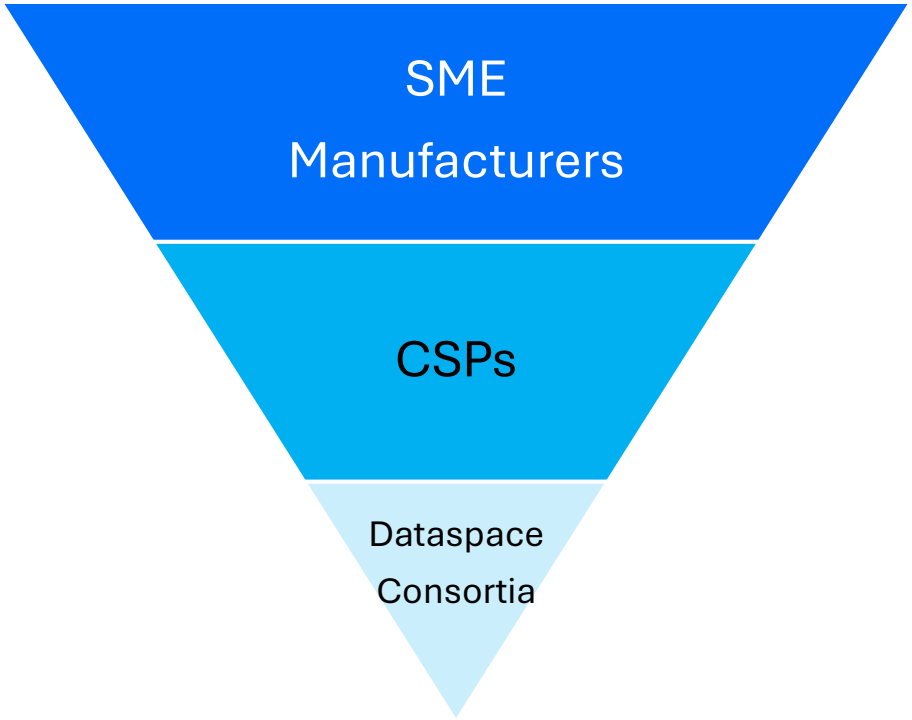
Dataspace Architecture Vision

- Enable Small and Median Enterprises (SME) to quickly participate with minimal technical expertise
- CSPs leverage their capabilities to offer solutions
  - Experts at automated service offerings and apps
  - Competitive advantage through close customer relationships
  - Built on in-house or federated infrastructure



Pyramid Technical Challenge

- Ease-of-use to pull through small companies
- Cost-effective operations
- Relatively easy to administer
- Leverage existing cloud infrastructure



Workshop Goal

- Develop a *Hosted Dataspace Service (HDS) Architecture Blueprint*
  - Technical solution for hosted SME dataspace services
    - Requirements analysis
    - System components
    - End-to-end processes
  - Generalized for industrial dataspaces – not specific to one
- Identify gaps
  - This is version 1 and will be subject to iterations
- Apply the Blueprint to Catena-X and others (Decade-X)

HDS Architecture Topics

Tenant Management	• Infrastructure and Service Provisioning
Onboarding	• Automated SME signup • Credential issuance
Operational Experience	• SME Experience (data loading, sharing) • CSP Management and Control • Industrial Use Cases • White labeling/Marketplace

Working Session Structure

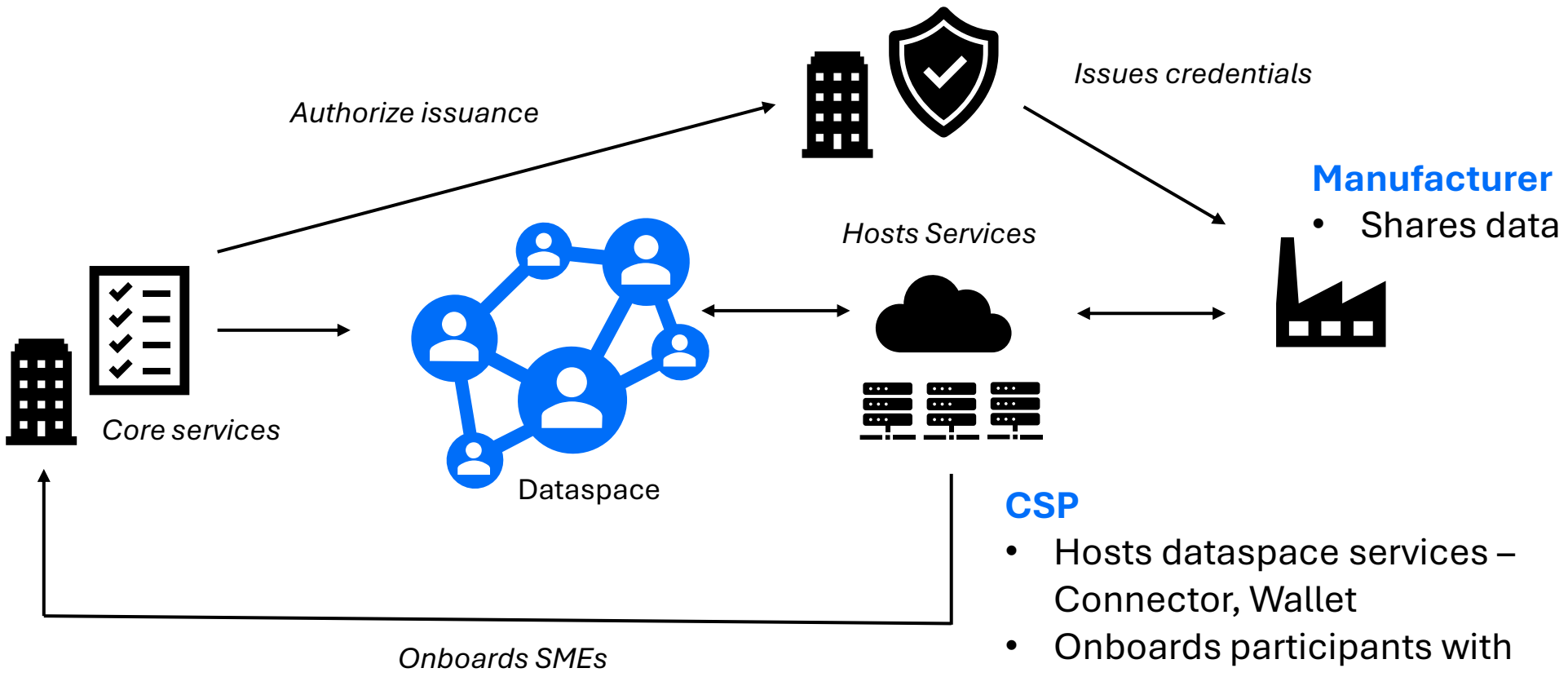
- Choose a break-out group you are interested in
- 45-minute breakout design session
  - Refine system components based on an architecture proposal
  - Develop sequence flow and system architecture
  - Highlight gaps and issues
- Presentations
  - 10-minute whiteboard presentation by each group
  - Requirements, process flows, gaps
  - Next steps

Roles

Governance Authority
- Rules

Operating Entity
- Core services
- Legal onboarding
- Member list

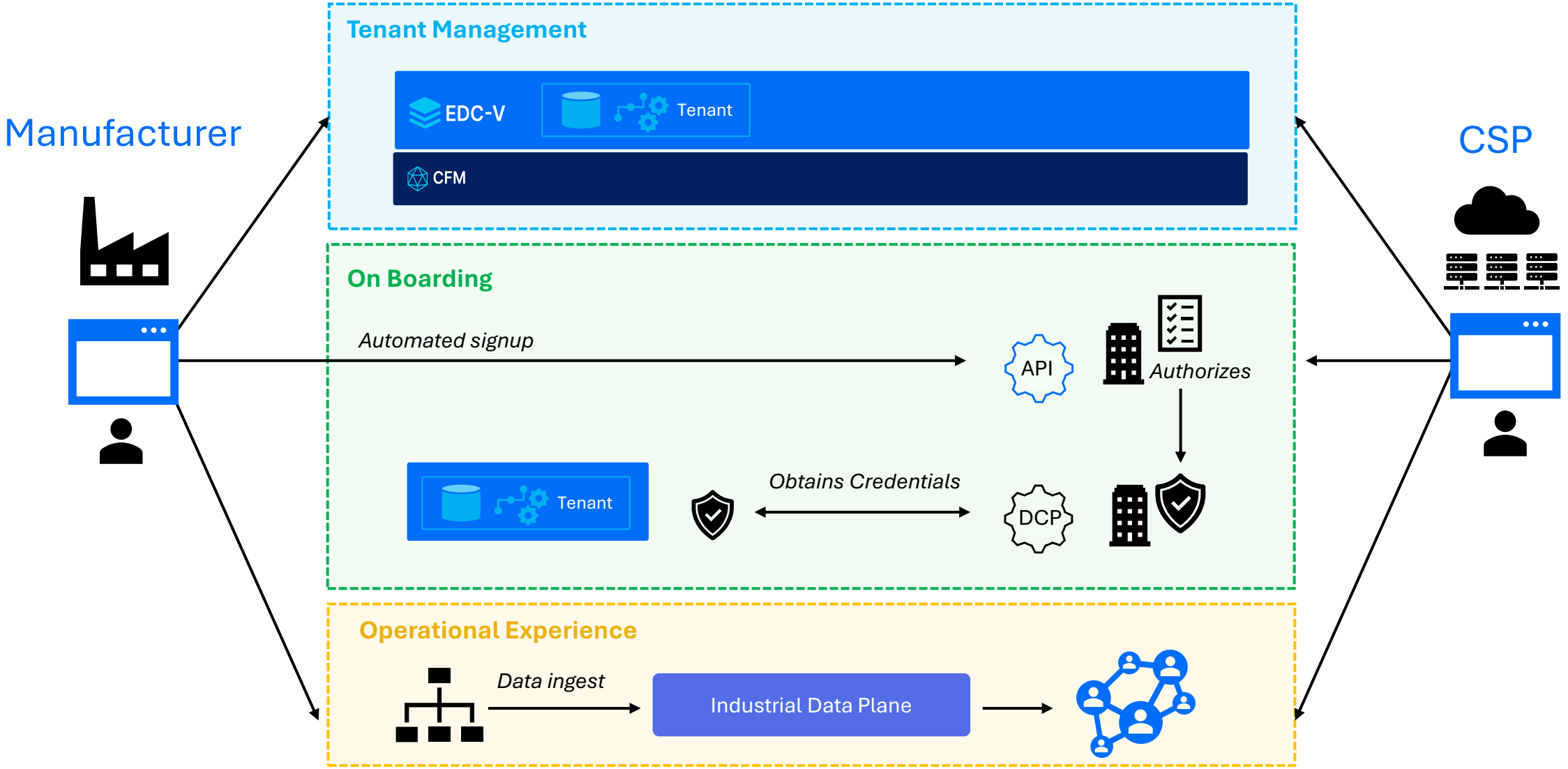
Credential Issuer
- Issues company credentials



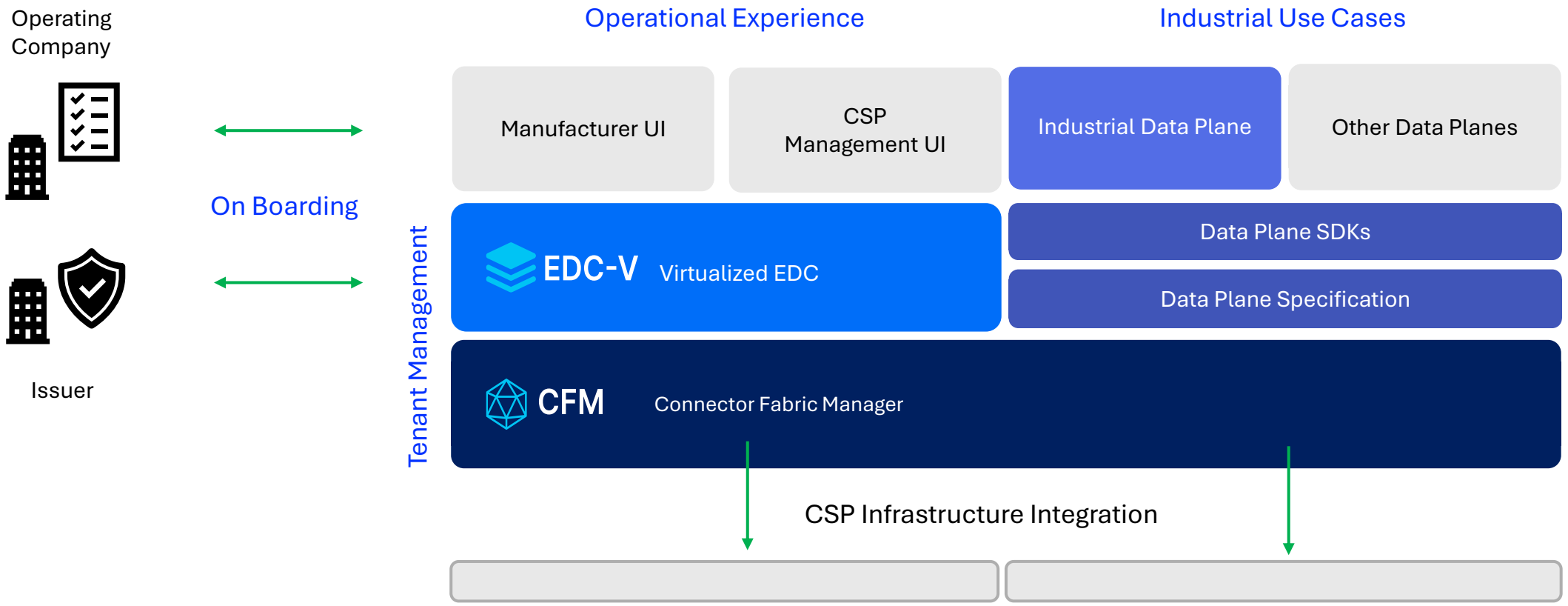
Manufacturer
- Shares data

CSP
- Hosts dataspace services – Connector, Wallet
- Onboards participants with the operating company

Starting System View



Starting Architecture View

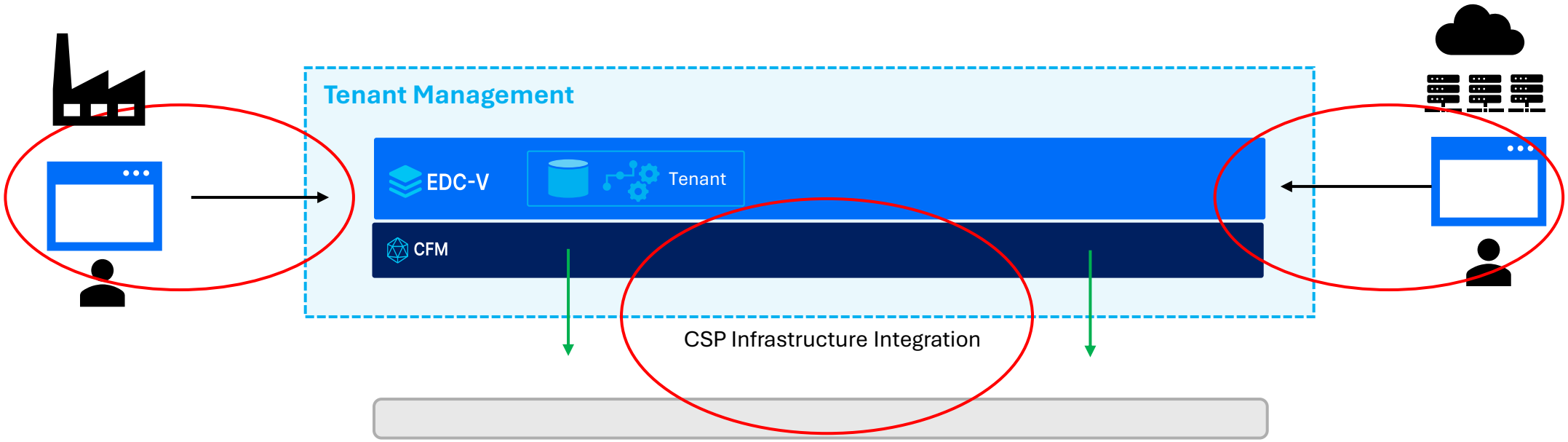


Architecture Blueprint

Working Session

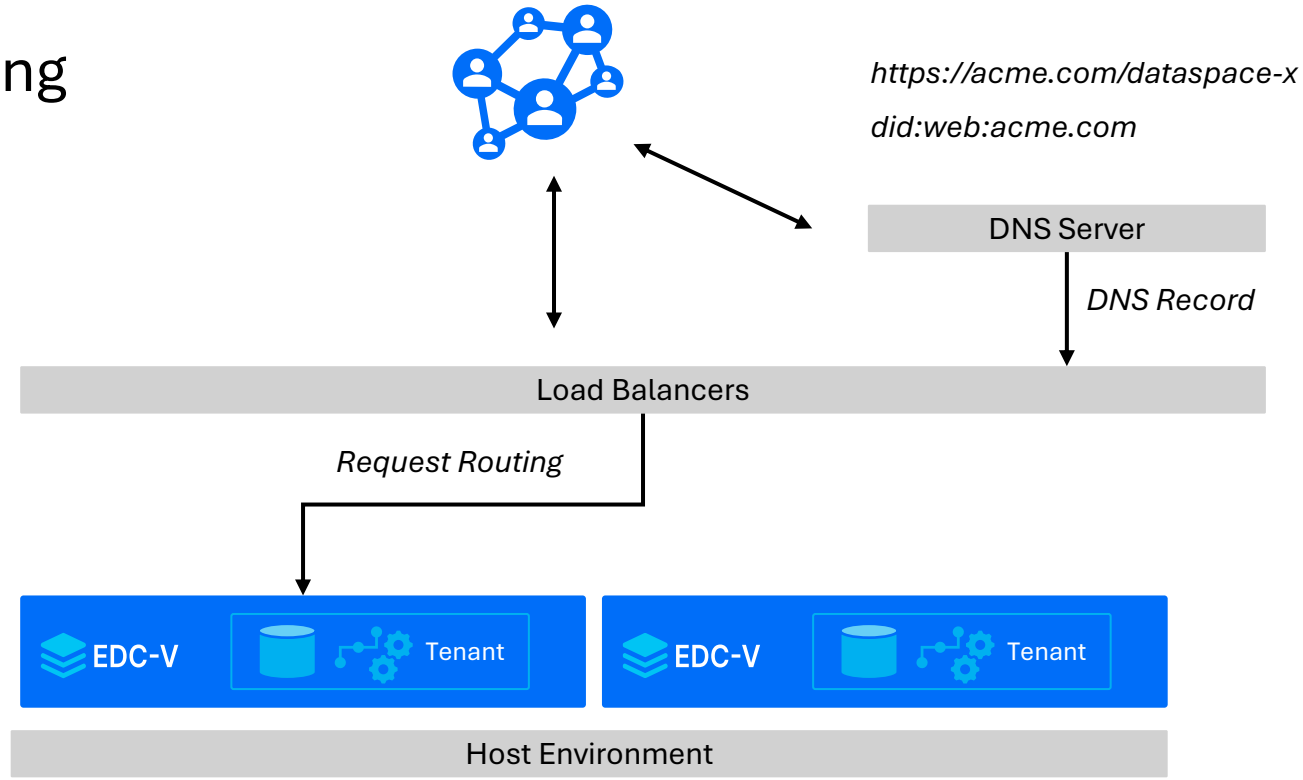
Tenant Management – What's Missing

- Infrastructure and Service Provisioning
  - What are the required infrastructure components (DNS, load balancing, object storage, etc.)?
  - How are these components integrated into the CFM?
- CSP management and control
  - What operational visibility is required to manage this process?



Infrastructure Integration

- Automated provisioning of EDC components
  - Connector
  - Identity Hub
- Load balancing and request routing



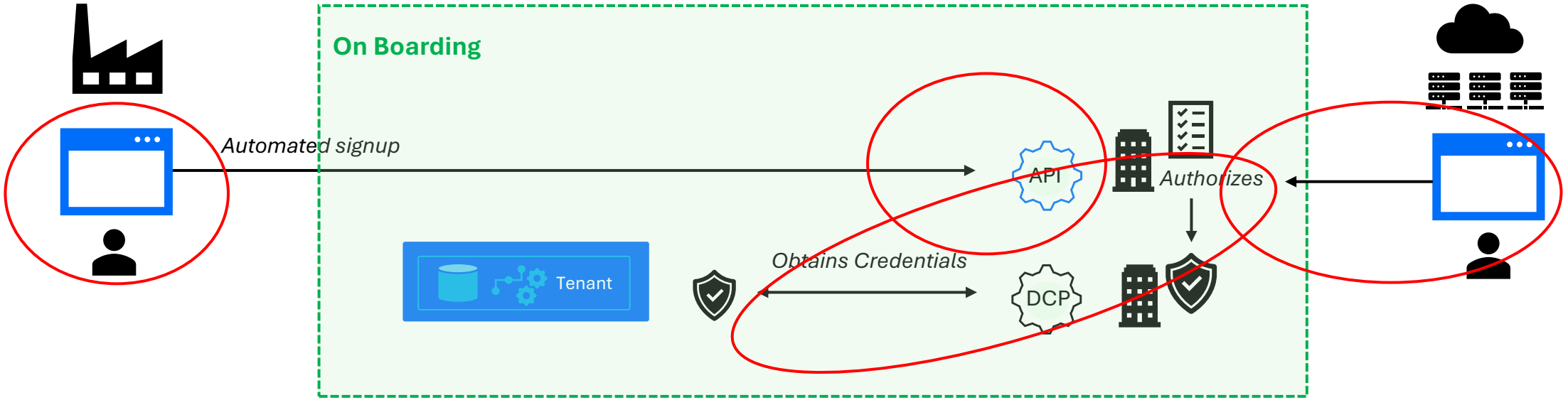
Connector Fabric Manager

- Provide a simple orchestration system to enable EDC tenant provisioning
- Orchestration is a stateful asynchronous execution system
  - CNCF NATS for reliable messaging (https://nats.io)
  - Jetstream for persistence
- Agents execute actions
  - Go framework
  - Write in any language (only need a NATS client)

Operational Experience Notes

Onboarding – What's Missing

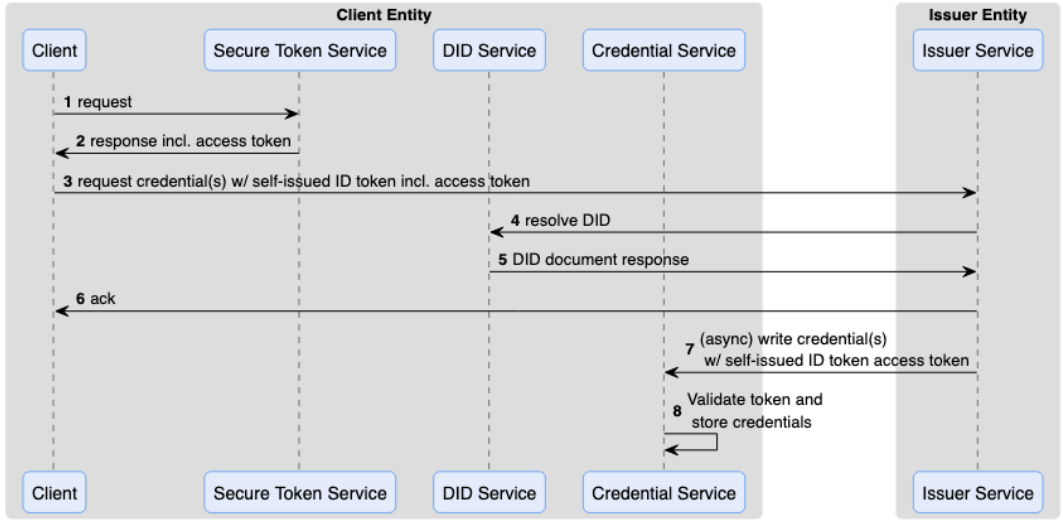
- Automated SME signup
  - How do CSPs get their customers legally onboarded into a dataspace?
- Credential Issuance
  - Once onboarded, how are credentials issued to the SME's wallet?
- CSP management and control
  - What operational visibility is required to manage this process?

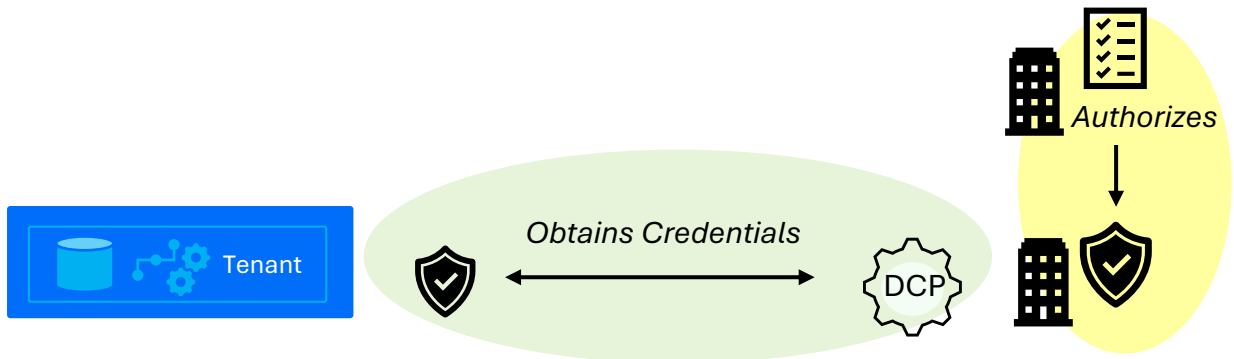


Onboarding Notes

DCP Flow

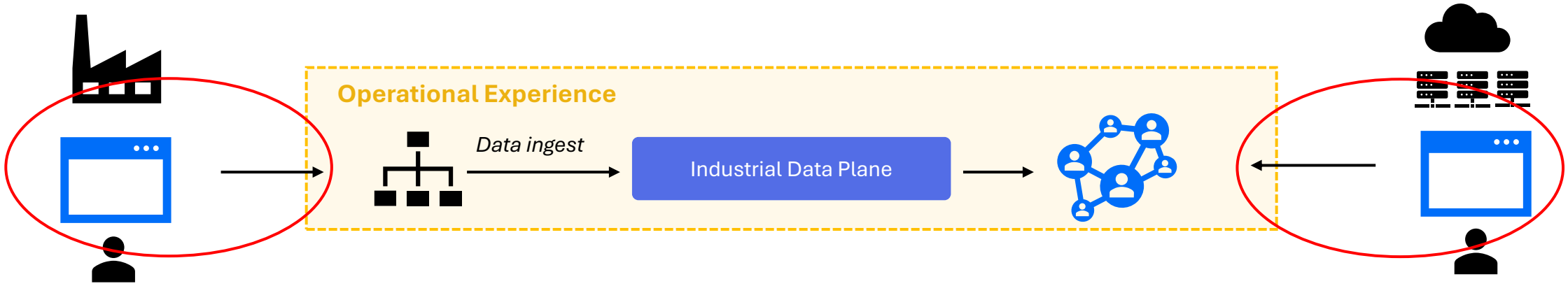
- DCP solves the problem of how the tenant obtains credentials
- Requires the operating company to update the issuer with the participant's DID
- The participant uses a self-issued token as proof to obtain the credentials





Operational Experience – What's Missing

- Assume SME manufacturers don't understand ODRL, policies, connectors, or data spaces
- SME UI
  - How are SMEs going to load data into the system and share it?
- CSP management and control
  - What operational visibility is required to manage this process?



Catena-X Enablement

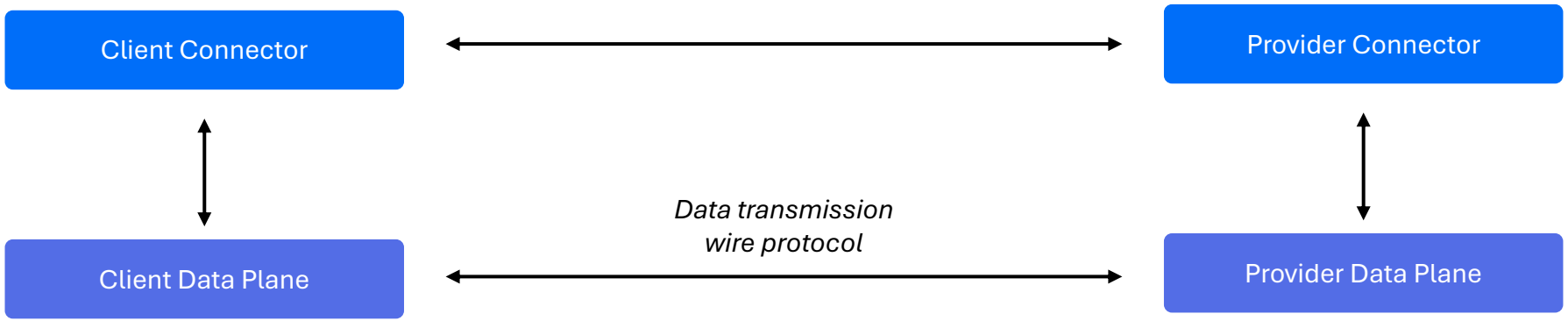
Working Session

Developing Data Planes

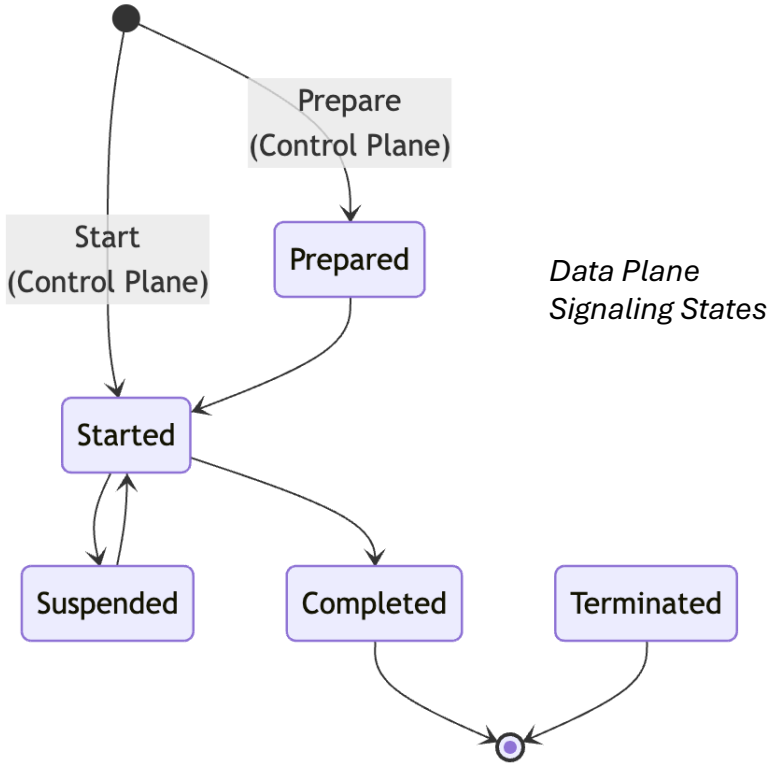
Hands-On Coding with the Eclipse Data Plane SDK

What is a Data Plane?

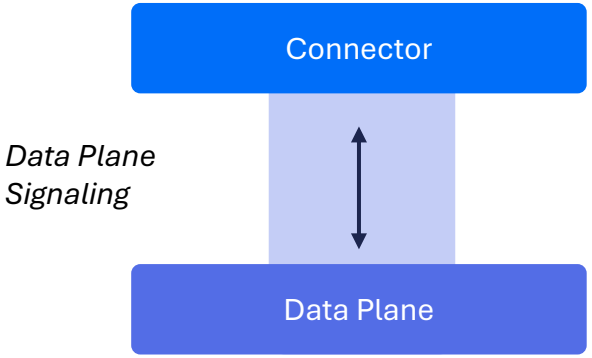
- A system responsible for transmitting data from a provider to a client
  - Perform data transfer on behalf of the control plane
- Use existing wire protocols and technologies
  - HTTP, messaging/streaming platforms, object storage, etc.
- Can be a third-party system, API, application or custom implementation



Developing Data Planes



Data Plane Signaling States



Data Plane Signaling

- **Goal: Grow a data plane ecosystem for different use cases**
- Data Plane Signaling
  - An interoperable standard for control plane/data plane communication
  - Under the Eclipse Dataspace Working Group (EDWG)
    - Along with the other main dataspace standards (DSP, DCP)
  - https://projects.eclipse.org/proposals/eclipse-data-plane-signaling
- Data Plane SDKs
  - Simplify the task of building custom data planes
  - Java, Go, .NET, Rust, Typescript
  - Developed under the Eclipse Data Plane Core Project (DCore)
  - https://github.com/eclipse-dataplane-core

Pull vs Push Transfers

One data transfer type does not fit all

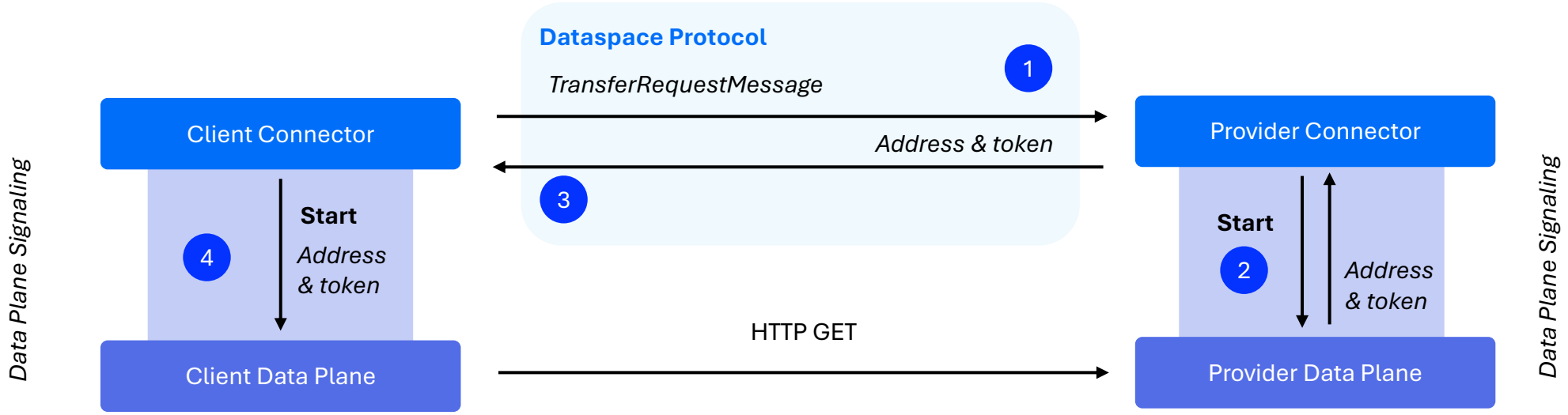
- Pull
  - The data consumer “retrieves” data from the provider
  - Consumer GETs from an HTTP endpoint, consumer subscribes to a provider queue
- Push
  - The data provider “sends” data to the consumer
  - Provider uploads blob data to consumer object storage, provider publishes a message or event to a consumer endpoint, etc.

Let’s Build Some Data Planes

- Based on the Go SDK
  - https://github.com/eclipse-dataplane-core/dataplane-sdk-go
- Explore the HTTP pull and streaming push examples

HTTP Pull Data Plane

- Client HTTP GETs from an endpoint



Using the Data Plane SDK (DSDK)

```
type ConsumerDataPlane struct { // ...}


func NewDataPlane() (*ConsumerDataPlane, error) {
        dataplane := &ConsumerDataPlane{tokenStore: common.NewStore[tokenEntry]()}
        sdk, err := dsdk.NewDataPlaneSDKBuilder().
                Store(memory.NewInMemoryStore()).
                TransactionContext(memory.InMemoryTrxContext{}).
                OnPrepare(dataplane.prepareProcessor).
                OnStart(dataplane.startProcessor).
                OnTerminate(dataplane.noopHandler).
                OnSuspend(dataplane.noopHandler).
                Build()
        if err != nil {
                return nil, err
        }
        dataplane.api = dsdk.NewDataPlaneApi(sdk)
        return dataplane, nil
}
```

Configure storage and custom code callbacks

Configure the Signaling server

Streaming Push Data Plane

- Provider publishes to a consumer queue
- Uses NATS