Data Plane Signaling

Kickoff Meeting

Data Plane Signaling

The Eclipse Data Plane Signaling specification defines an interoperable protocol for DSP control planes to communicate with data planes. The scope of the specification includes:

- A protocol for control planes to manage data transfers using a variety of wire protocols, ranging from HTTP-based to streaming event systems.
- A registry of data transfer types so that other projects can define non-clashing identifiers for those types.
- An associated TCK in the Eclipse Dataspace TCK project
  - https://github.com/eclipse-dataspacetck
- The following are out of scope for Eclipse Data Plane Signaling:
  - The definition of specific wire protocol formats and data transfer types
  - Any data plane implementation
- Eclipse Project
  - https://github.com/eclipse-dataplane-signaling

Three Projects

- **Data Plane Signaling**
  - Interoperability: Plug any data plane into any control plane
- **DCore**
  - Building blocks for creating custom data planes
  - Will Implement the data plane parts of Signaling
- **EDC**
  - Evolving to be only a control plane
  - Will implement the control plane parts of Signaling
  - Data plane components are deprecated and will be removed

DCore

- Multi-language SDKs
  - Rust, Go, Java, .NET, potentially others
    - Language parity for base features
    - Not “ports” – use idiomatic approaches for each language
  - Implement core Signaling features
  - Minimalist, designed to be consumed by other projects that distribute custom data planes
    - Industry-specific (industrial data)
    - Data specific (streaming, HTTP, etc.)
  - Focus on Rust
    - Memory safety, emphasis on high performance, low overhead (other languages are VM-based)
- Multi-language Facets
  - Higher-level feature capabilities that can be plugged into SDKs
  - Not available for all languages – focus on Rust
  - Distributed lock management, token generation, and token cache
- Rust-based HTTP Data Plane
  - Implements Signaling
- Samples

Time Frame

- Mid March
  - Initial specification Candidate Release (we can’t have an official Eclipse release)
  - Dataplane SDK Previews
    - Java, Go, Rust
  - EDC Signaling support
- May
  - TCK Preview
- End June
  - Eclipse Specification Release + TCK + at least one implementation
- Post June
  - ISO submission

Next Steps

- Specification template adoption
  - Conversion to DSP/DCP spec template
- Technical housekeeping
  - Review the initial specification and identify gaps and issues
  - Align schemas, specification text, and DCore implementations
- Data Plane/Control Plane Registration and Auth
  - Define and specify
- Data Transfer Types Registry and Profiles
  - Define a profile template and requirements
  - Do we want to define specific profiles?
    - May require a project scope change