# 5 DS2 IDT Broker (IDT)

## 5.1 DS2 IDT Broker (IDT)

| | |
|---|---|
| **Owner(s):** | ICE |
| **DOA Task:** | T6.3 |
| **Tier:** | 1 |
| **Nature:** | System |
| **Results:** | K6.3 |

*The IDT Broker creates a federated environment that allows collaboration and sharing of data among different organizations, communities or jurisdictions with a shared interest in a particular area or domain. This allows for the pooling of resources and expertise to create more comprehensive and diverse datasets. IDT will 'network' with other IDTs allowing the federation approach as described in Section 1 and be overlayed on the Infrastructure of T6.1. This networking will be provided out-of-the box to enable connection to other IDTs.*

### 5.1.1 Introduction

**Purpose:** IDT is the core enabler of DS2 who purpose is to be deployed in front of participants data source/spaces and network connected to any other IDT-enabled data source. As such its aim is to run all DS2 modules, including the DS2 Connector, the core module for Inter-Dataspace communication and data transfer, and the Containerisation module for DS2 module deployment. The IDT contains the core Kubernetes runtime to run all containerised modules and a series of additional open-source software for module management.

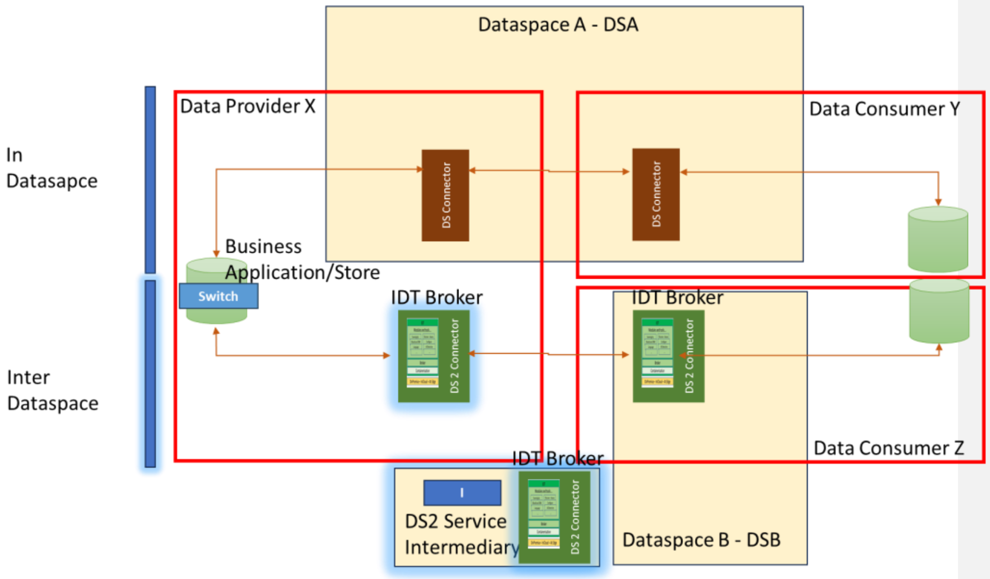
**Description:** The IDT contains the Kubernetes runtime that is the core service to run all modules in a containerised way. Those modules descriptors are uploaded to the DS2 Portal Marketplace and then, using the IDT Kubernetes UI, are deployed to the Kubernetes runtime. The Containerisation module kicks-in and then converts those module descriptors to full module charts effectively deploying the DS2 modules. The Kubernetes UI, alongside the Management and Monitoring Controller, will be used to manage and monitor all DS modules running on a participants IDT. Additional services will also be run as part of the IDT such as certificate management, ingress network traffic management using traditional ingress controllers with ingress resources to expose modules and eventually transitioning to service mesh and gateway API, storage management and possibly other useful open-source tools. The other key component of the IDT is the DS2 Connector used for Dataspace like communications following current IDSA and Gaia-X standards. An additional IDT UI will be provided for module navigation and Connector management.

Note, under the additional information section the Dashbutton is also identified and detailed.

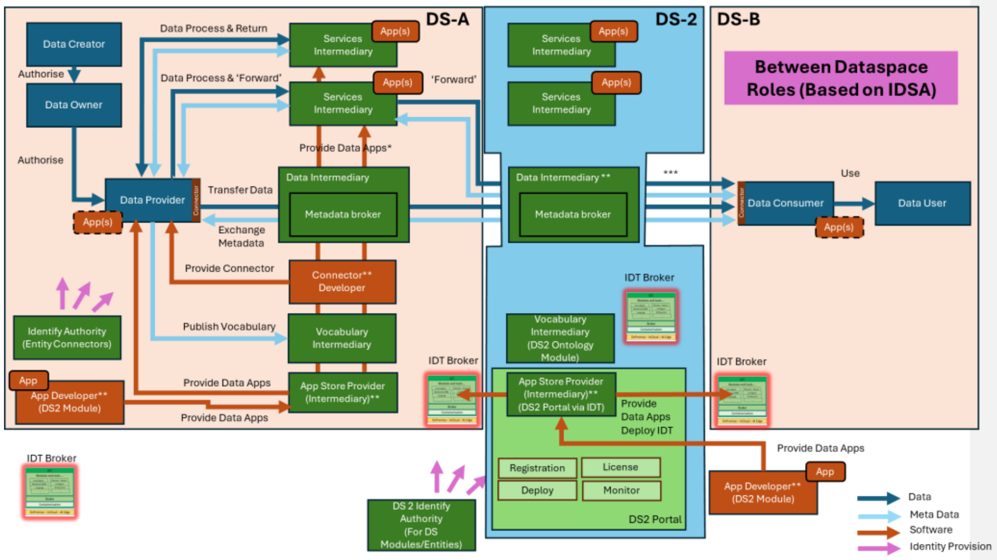
### 5.1.2 Where this component fits

#### 5.1.2.1 Big Picture

IDT is the core DS2 software, containing the DS2 Connector for Dataspace like communication and data exchange, thus it is required at all participants for any DS2 enabled connection and inter-Dataspace operations. The IDT also runs the Containerisation module, required to deploy all DS2 modules.



Its fit (RED BOX) in to the DS-DS environment is as follows



| Where | Status |
|---|---|
| **Within a single Dataspace** for use between participants in that Dataspace only | This could be used in the same model as listed in “Across Dataspace with Intermediary” but due to the focus of DS2 this will not be piloted/validated |
| **Deployed and used by a single participant** to enable the participant in either an In-Data space or Inter- Data space scenario | Yes: Since IDT runs the DS2 modules, those can be used for In-Dataspace purposes as well. |
| **Across Dataspaces without Service Intermediary** | Yes: IDT is the key software for inter-Dataspace connections |
| **Across Dataspace with Intermediary** | Yes: An intermediary service required to run modules will run them using the IDT |
| Other Comments | N/A |

#### 5.1.2.2 Within a single Dataspace (where applicable)

Yes but not in the project for reasons mentioned above

#### 5.1.2.3 Deployed and used by a single participant (where applicable)

The IDT is the required software to run all DS2 modules and DS2 connections, thus it runs at every DS2 participant. Depending on the modules, those can also be used for intra-Dataspace processes.

#### 5.1.2.4 Across Dataspaces (where applicable)

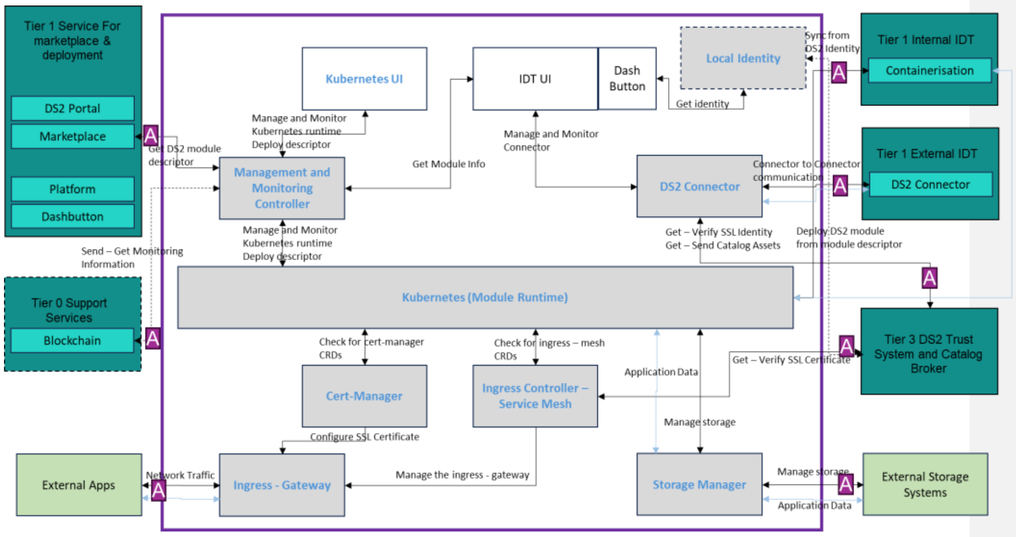
As stated, the IDT runs the DS2 Connector, which is the key piece for the inter-Dataspace connectivity. The IDT along with the DS2 Connector will be run at every participant willing to establish DS2 connections to a different Dataspace and participant.

#### 5.1.2.5 Dataspace Intermediary (where applicable)

A dataspace service intermediary is used to host additional services, typically DS2 modules, thus the IDT is required to run those modules.

### 5.1.3 Component Definition

The figure below represents the actors, internal structure, primary sub-components, primary DS2 module interfaces, and primary other interfaces of the module.



Tier 1 Service For marketplace & deployment
DS2 Portal
Marketplace
Platform
Dashbutton
Tier 0 Support Services
Blockchain
External Apps
Kubernetes UI
IDT UI
Dash Button
Local Identity
Sync from DS2 Identity
Get identity
Manage and Monitor Kubernetes runtime Deploy descriptor
Get DS2 module descriptor
Get Module Info
Manage and Monitor Connector
Management and Monitoring Controller
DS2 Connector
Connector to Connector communication
Send – Get Monitoring Information
Manage and Monitor Kubernetes runtime Deploy descriptor
Get – Verify SSL Identity
Get – Send Catalog Assets
Deploy DS2 module from module descriptor
Kubernetes (Module Runtime)
Check for cert-manager CRDs
Check for ingress – mesh CRDs
Application Data
Get – Verify SSL Certificate
Cert-Manager
Ingress Controller – Service Mesh
Configure SSL Certificate
Manage storage
Network Traffic
Ingress - Gateway
Manage the ingress - gateway
Storage Manager
Manage storage
Application Data
Tier 1 Internal IDT
Containerisation
Tier 1 External IDT
DS2 Connector
Tier 3 DS2 Trust System and Catalog Broker
External Storage Systems
A

Figure 1: Schema for the Module

This module has the following subcomponent and other functions:

- **Kubernetes (Module Runtime):** Kubernetes is the leading technology in container orchestration and the choice and key component of the IDT for deployment and integration of the DS2 modules. This is the core IDT subcomponent that runs and orchestrates the DS2 modules and all the other IDT subcomponents, as containers. This is Open-Source software and the current distribution being used is K3s, a lightweight version of Kubernetes easy to install, half the memory, all in a binary, in less than 100 MB among other enhancements. One of the main advantages is the flexibility of installation, since it can be deployed at a participant edge, onPremise, InCloud, etc.
- **Kubernetes UI:** The Kubernetes UI is open-source software based on Rancher that allows to deploy and manage Kubernetes in a more user-friendly way both onPremise and InCloud. The Kubernetes UI will provide the management interface for platform administrators and the module deployment interface for participants

running the IDT. This interface will be used to deploy the module ChartDescriptors, configuration files that describe how a module runs on Kubernetes, from the DS2 Portal Marketplace in the IDT. The Containerisation module will then transform the descriptors into full Helm Charts and deploy them to the Kubernetes subcomponent. The Kubernetes UI will also provide the monitoring interface to the IDT Kubernetes subcomponent and the DS2 modules.

- **Management and Monitoring Controller**: This is the main interface from the Kubernetes UI to the Kubernetes subcomponent and also for external integrations. The Management and Monitoring Controller is open-source software based on the Kubernetes and Rancher API and the Rancher agents. It is used as the primary interface to the IDT Kubernetes subcomponent for management and deployment of modules. It will also be used as the primary interface for monitoring which will potentially be integrated with the DRM Blockchain module for traceability. In addition, further research on using other modules for monitoring such as Prometheus-Grafana will be conducted for enhanced monitoring.
- **Ingress – Gateway:** The ingress or gateway resource provides the entry point to the IDT Kubernetes subcomponent via the Ingress controller, thus, the IDT network, for all network traffic from external apps, being an external app, any system external to the IDT. It describes how the DS2 modules are exposed outside of the IDT. Initially the modules will use a Kubernetes Ingress resource to expose the modules but further research will be conducted to examine the use of the Gateway API and Service Mesh technology.
- **Ingress Controller – Service Mesh:** Based on Open Source, the Ingress Controller is the Kubernetes controller dealing with Ingress resources, that is, managing the entry point to the IDT and how the DS2 modules are exposed outside of IDT. Further research is expected for replacement of the Ingress Controller with Service Mesh technology and the Kubernetes Gateway API, that adds a transparent layer to provide the IDT with enhanced connectivity, security, control and observability. The use of the Service Mesh could also be a key feature for using more secure communication via mutual TLS protocol (mTLS) in all DS2 communications which provides and additional trust layer. This could also be integrated with the DS2 trust and identity system.
- **Storage Manager:** Open-source software to provide the interface between the IDT Kubernetes subcomponent and the physical storage for DS2 stateful modules. This will use Kubernetes native storage technology to allow highly available stateful module deployments in IDT. When data from DS2 modules need to be persisted in a participant backend storage system, the Storage Manager will be used to map current deployment and Kubernetes Persistent Volumes to external storage systems. This is not a storage system or technology for modules… If DS2 modules need to use storage, the DS2 modules need to provide them by packaging them in their module Chart.
- **CertManager:** Based on Open-source software, it provides management of SSL certificates for secure connectivity ie. HTTPS, with verified signed certificates using Let’s Encrypt CertificateAuthority (CA) and configures them for the Ingress or Gateway resource. The CertManager integrates with the Ingress Controller and/or Service Mesh subcomponents and in addition, further research on integration with DS2 Trust system will be explored.
- **DS2 Connector:** The DS2 Connector in the IDT is the key element that will allow for DS2 transactions and data exchange, following the IDSA and Gaia-X standards. The Open-source Eclipse EDC Connector (or the Tractus-X extension) will be used to

provide interoperability between Dataspaces and secure, trustworthy exchange of data. Following existing Dataspace principles and protocols, the DS2 Connector will use the DS2 Trust system for identity management and will connect to other participants IDT DS2 Connectors in other Dataspaces for data exchange. The Connector will also integrate with the DS2 Catalog or a Dataspace level Metadata Broker for participant and data discovery.

**Commented [SC1]:** Leave this note in - to be seen

- **Local Identity:** This module is optional and it provides local identity, authentication and authorization to access a participant IDT and its modules by the various users types within a company. Based on Open Source Keycloak identity provider software, further research will be done in order to explore the possibility of linking the Local Identity with the DS2 Trust system.
- **Tier 0 Support Service Stack:**
  - **DRM** and **API**: For further exploration integration of the monitoring controller with the Blockchain will be considered.
- **Tier 1 Service Stack for Marketplace and deployment** and **API:** The full stack will be implemented as generically described elsewhere in this document. Exceptions: The IDT uses the DS2 Portal and Marketplace to retrieve the ChartDescriptors of modules and deploy them via the Kubernetes UI. Then the Containerisation module uses the descriptors to deploy the full module Helm Chart. In addition, The DS2 Connector in the IDT integrates with other IDT DS2 Connectors for data exchange.
- **Tier 3 Trust Stack and Catalog** and **API:** The IDT will make use of the relevant parts of the DS2 Trust Stack for certificates in the Ingress Controller – Service Mesh and identities in the DS2 Connector. The IDT will also connect via the DS2 Connector to the Catalog.
- **External Apps:** External Apps refer to any software application external to the IDT and DS2 ecosystem that uses the DS2 Connector in the IDT for any DS2 data transaction. It's the application that can trigger a data exchange via the Connector, either as a consumer or producer.
- **External Storage Systems:** This refers to any external storage system, physical or software defined, that a participant has already in place and where data from the IDT and DS2 ecosystem can be persisted, thus, is mapped via the Storage Manager into the IDT Kubernetes.

#### 5.1.4 Technical Foundations and Background

The foundations of the IDT Broker are Open-source software, mainly based on Kubernetes and the Cloud Native Computing Foundation (CNCF) ecosystem tools. The core of the IDT is K3s Kubernetes distribution to orchestrate the DS2 module containers. It includes Rancher, its API, and agents to manage Kubernetes and deploy the DS2 modules from a UI. It relies on nginx Ingress Controller and Ingress resource for exposing the cluster to the internet, and further research will be conducted to replace those by more modern approach such as Service Mesh and Gateway API. It also uses Cert-manager as the tool to manage ssl certificates for secure communication to-from the cluster. Kubernetes native storage system is also provided, first as NFS but further research will be conducted to provide more native storage systems such as OpenEBS, Longhorn … As the key technology for DS2 communications, the IDT broker makes use of a Dataspace Connector, most likely the base Eclipse EDC Connector, which provides standard Dataspace communication across Dataspaces between the different participants IDTs. Further research will be done to explore more advanced connectors such as Tractus-X which is also based on EDC Connector.

| Subcomponent/Component | Owner | License |
|---|---|---|
| Kubernetes | Open Source | Apache 2.0 |
| Rancher | Open Source | Apache 2.0 |
| Ingress nginx | Open Source | Apache 2.0 |
| Service Mesh | Open Source | Apache 2.0 |
| Cert-manager | Open Source | Apache 2.0 |
| Kubernetes NFS provisioner | Open Source | Apache 2.0 |
| Open EBS | Open Source | Apache 2.0 |
| Keycloak | Open Source | Apache 2.0 |
| Eclipse EDC Connector | Open Source | Apache 2.0 |

### 5.1.5 Interaction of the Component

The following table specifies the primary input/output controls/data to blocks which are not part of the module

| With Module/Feature | Receives from/Gives To | What |
|---|---|---|
| Tier1 Portal Marketplace | Receives From | Module ChartDescriptor for module deployment |
| Tier 1 Internal IDT Containerisation | Gives To | Module ChartDescriptor deployed in the IDT |
| Tier 1 Internal IDT Containerisation | Receives From | Module Chart for module deployment |
| Tier 3 Trust System | Receives From/Gives To | Get identities, Validates identities |
| Tier 0 Blockchain | Receives From/Gives To | Sends and Gets monitoring information |
| Tier 1 External IDT DS2 Connector | Receives From/Gives To | DS2 Control and Data plane communication |
| External Apps | Receives From/Gives To | Network traffic and Data |
| External Storage systems | Receives From/Gives To | Persistent Data |

### 5.1.6 Technical Risks

| Risk | Description | Contingency Plan |
|---|---|---|
| Choice of Connector | The choice of the IDT connector is a key decision and will have a huge impact on DS2 and all modules if integration is too coupled | Make sure the right connector is selected and try to decouple integration as much as possible with the integration architecture |
| Local Identity vs DS2 Identity | Identity for login in modules and integration with DS2 trust system | Need to find out if a local identity is needed and if link to DS2 identity is needed |

### 5.1.7 Security

| Security Issue | Description | Need |
|---|---|---|
| Inter-participant trust | The IDT and its connector need to provide secure and trust-worthy connectivity and data exchange | Connect to the DS2 trust system |

|  |  |  |
|---|---|---|
| In-Dataspace | In Dataspace, a participant will need to provide credentials in order to log in to modules | Provide local identity and integrate with DS2 trust system |

### 5.1.8 Data Governance

| Data Governance Issue | Description | Need |
|---|---|---|
| Connector Control plane | The connector is the responsible for contracts and agreements and those are stored in database | Comply with GDPR regarding storing contract and agreements |
| Connector Data plane | The connector is the key software for Dataspace data exchange | Although the connector is the key for data exchange, the data is then transferred and stored in participants backend storage, so no additional needs for the connector |

### 5.1.9 Requirements and Functionality

This module will be used in the following usecases:

|  |  |
|---|---|
| City Scape | ✓ |
| Green Deal | ✓ |
| Agriculture | ✓ |
| Inter-Sector | ✓ |

Their requirements and functions/extensions to achieve them relative to this module, specifically extracted from the use case are as per the table below noting that in many cases further discussion might need to take place between pilot partner and module partner to determine if a fit or the scope of the precise fit.: :

In respect to all use cases the DS2 IDT Broker module is the DS2 core module where all the other modules are deployed including the DS2 Connector, thus it's been identified by all use cases as a Must, as expected.

| WHERE | WHAT | WHY | Run/Design Time | Priority |
|---|---|---|---|---|
|  | **Use Case 1: City Scape** |  |  |  |
| N/A |  |  | R & D | M |
|  | **Use Case 2: Green Deal** |  |  |  |
| N/A |  |  | R & D | M |
|  | **Use Case 1: Agriculture** |  |  |  |
| N/A |  |  | R & D | M |

### 5.1.10 Workflows

The following sub-sections describe the sequence diagrams of the Module

#### 5.1.10.1 Install Chart Descriptor (Install Module)

This feature provides the capability to install a DS2 Module from the Marketplace, first by installing the Chart Descriptor that will trigger the Containerisation module workflow to install the Module Helm Chart. Figure 2 shows the sequence diagram of this feature.

The main steps/functionalities are as follows:

- Get the Chart Descriptor
- Install Chart Descriptor

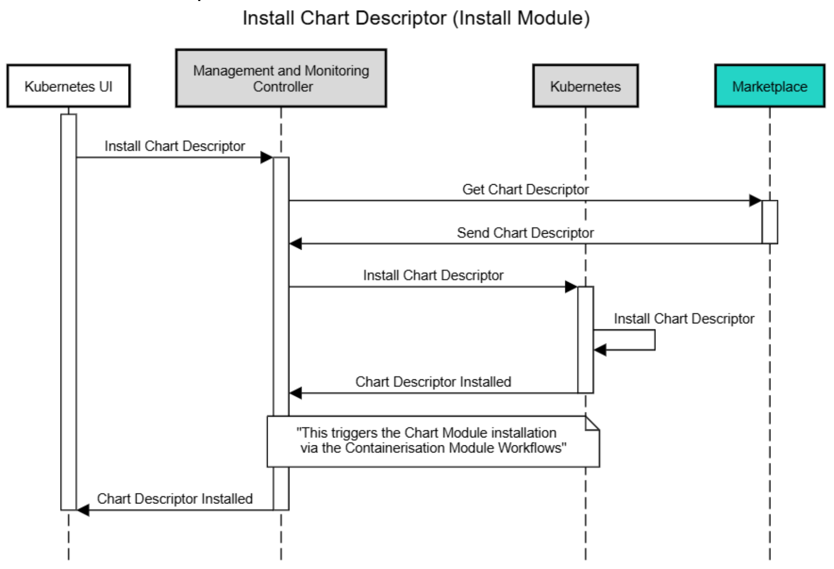

Install Chart Descriptor (Install Module)
Kubernetes UI
Management and Monitoring Controller
Kubernetes
Marketplace
Install Chart Descriptor
Get Chart Descriptor
Send Chart Descriptor
Install Chart Descriptor
Install Chart Descriptor
Chart Descriptor Installed
"This triggers the Chart Module installation via the Containerisation Module Workflows"
Chart Descriptor Installed

Figure 2: Install Chart Descriptor sequence diagram

#### 5.1.10.2 Display Modules Helm Chart Info (Kubernetes UI)

This feature provides the capability to display the information of the Modules Helm Chart installed in the IDT Kubernetes. Figure 3 shows the sequence diagram of this feature.

The main steps/functionalities are as follows:

- Get the Charts Info
- Display the Charts Info

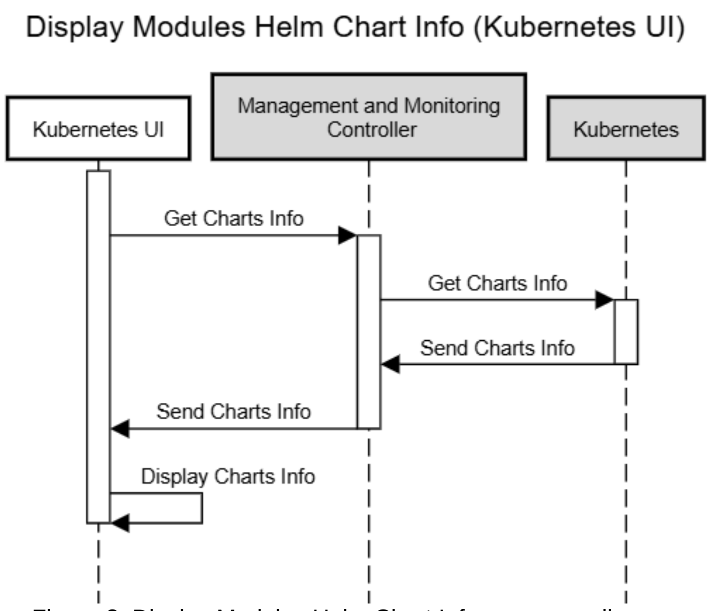


Figure 3: Display Modules Helm Chart Info sequence diagram

#### 5.1.10.3 Display Modules Info (IDT UI)

This feature provides the capability to display the information of the Modules installed in the IDT Kubernetes. It’s different form the previous one in that the information is at the module level vs chart level. Figure 4 shows the sequence diagram of this feature.

The main steps/functionalities are as follows:

- Get the Charts Info
- Display the Module Info

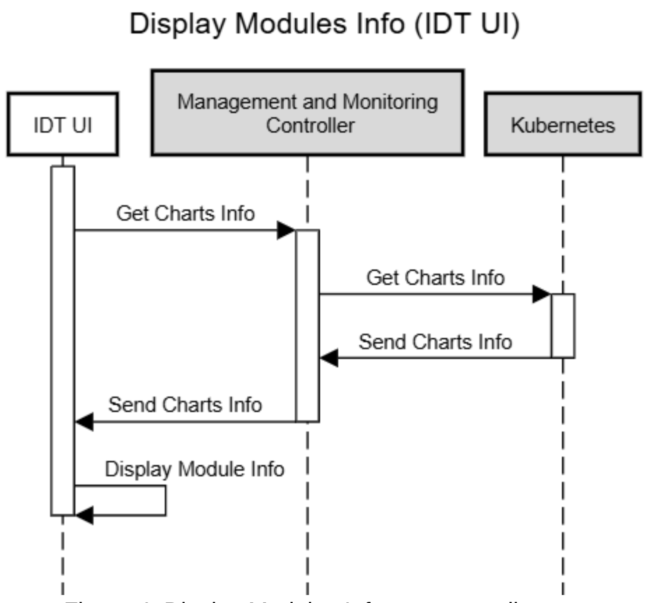


Figure 4: Display Modules Info sequence diagram

#### 5.1.10.4 Manage and Monitor Connector from IDT UI

This feature provides the capability to run management commands and monitor the Connector from the IDT UI. Figure 5 shows the sequence diagram of this feature.

The main steps/functionalities are as follows:

- Send the Management Commands
- Run the Commands in the Connector
- Return Commands Status
- Display Command Status
- Get Connector Info
- Display Connector Info

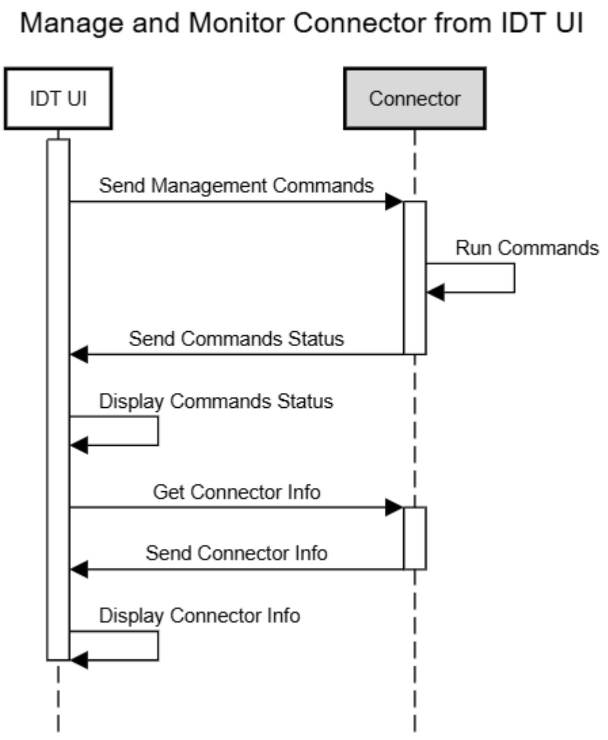


Figure 5: Manage and Monitor Connector from IDT UI sequence diagram

#### 5.1.10.5 Data Transaction via Connector from IDT UI

This feature provides the capability to run typical Dataspace Data Transaction steps from the IDT UI and optionally persist data in the external storage systems. Figure 6 shows the sequence diagram of this feature.

The main steps/functionalities are as follows:

- Send Data Transaction
- Run Data Transaction Steps
- Get Transaction Info
- Display Transaction Info
- Store Transaction Data

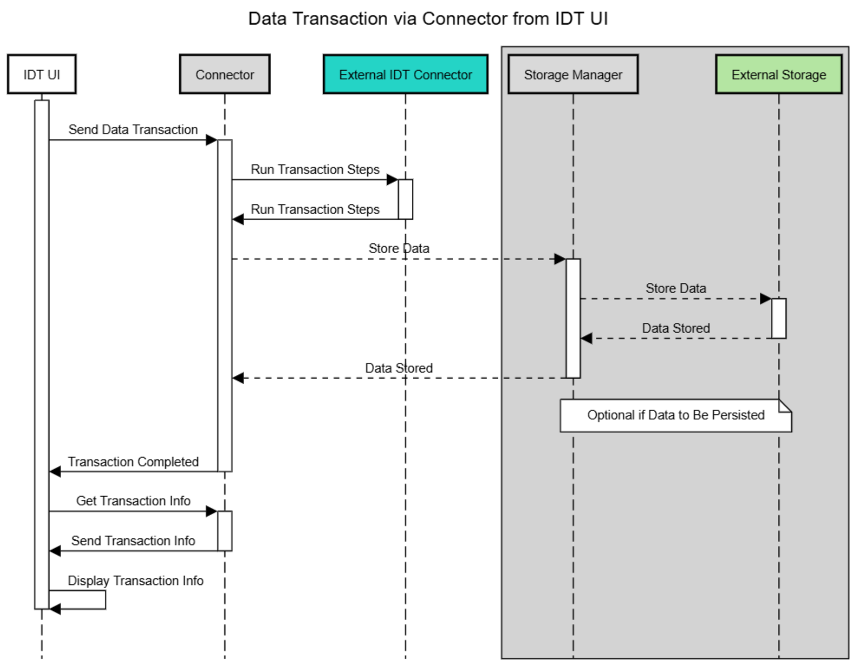


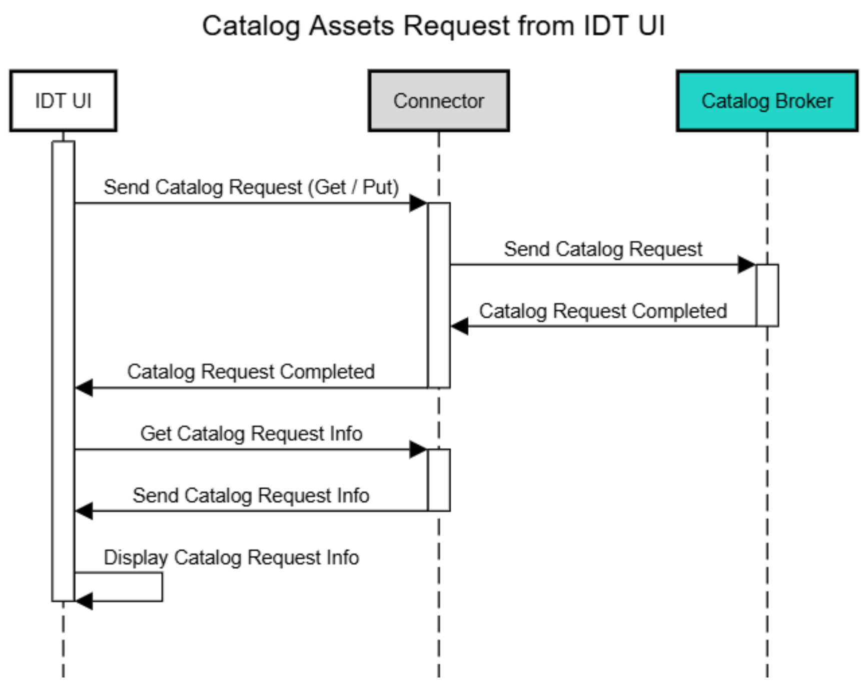
Figure 6: Data Transaction via Connector from IDT UI sequence diagram

### 5.1.10.6 Catalog Assets Request from IDT UI

This feature provides the capability to run typical Dataspace Catalog Transaction steps from the IDT UI to the Catalog Broker: publish asset, retrieve asset. Figure 7 shows the sequence diagram of this feature.

The main steps/functionalities are as follows:

- Send Catalog Request
- Get Catalog Request Status
- Get Catalog Request Info
- Display Catalog Request Info


Catalog Assets Request from IDT UI
IDT UI
Connector
Catalog Broker
Send Catalog Request (Get / Put)
Send Catalog Request
Catalog Request Completed
Catalog Request Completed
Get Catalog Request Info
Send Catalog Request Info
Display Catalog Request Info
Figure 7: Catalog Asset Request from IDT UI sequence diagram

#### 5.1.10.7 Data Transaction via Connector from External App

This feature provides the capability to run typical Dataspace Data Transaction steps from an External App and optionally use the stored data. Figure 8 shows the sequence diagram of this feature.

The main steps/functionalities are as follows:

- Send Data Transaction
- Run Data Transaction
- Get Stored Data
- Use Data

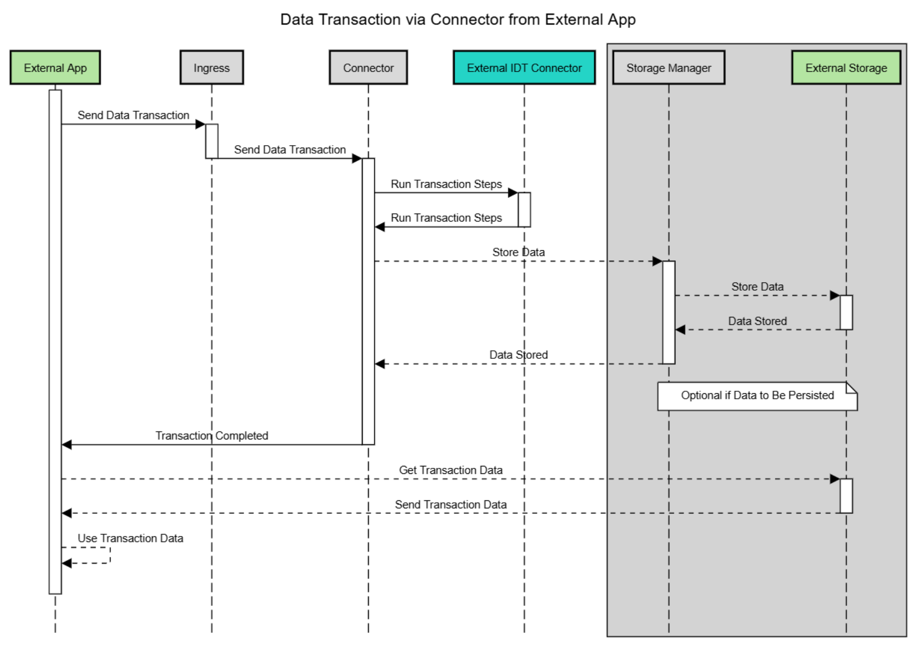


Figure 8: Data Transaction via Connector from External App sequence diagram

#### 5.1.10.8 Configure Https Certificate

This feature provides the capability to install and configure https certificates for https transactions in communication to / from the IDT. Figure 9 shows the sequence diagram of this feature.

The main steps/functionalities are as follows:

- Create Certificate
- Configure Certificate
- Get Certificate Info
- Display Certificate Info

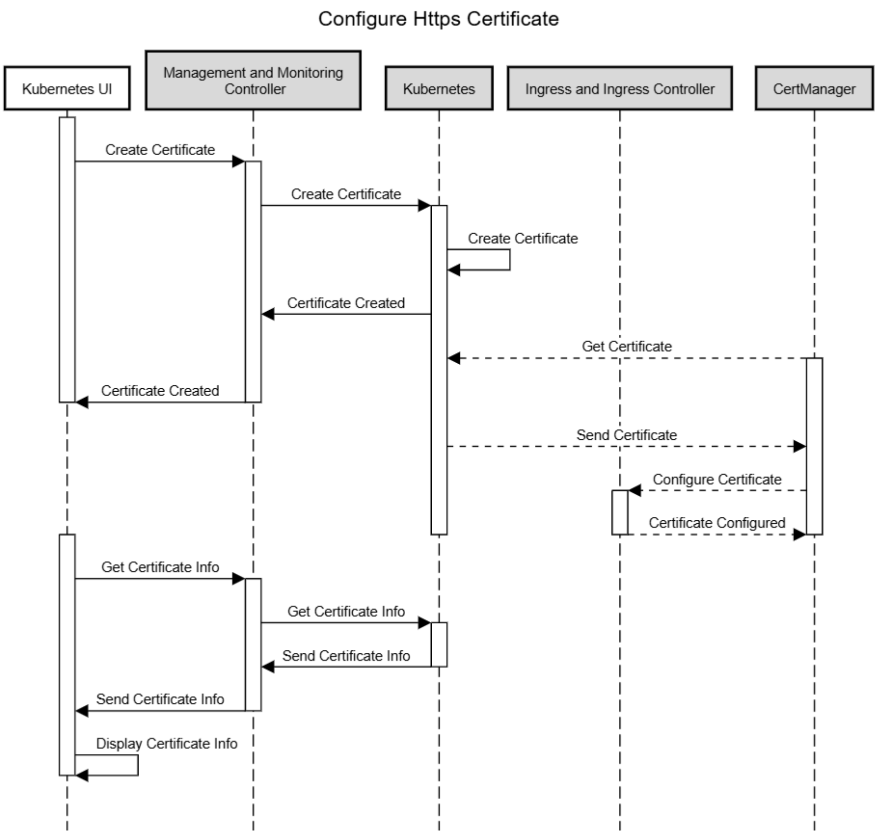


Figure 9: Configure Https Certificate sequence diagram

### 5.1.11 Role, Resourcing, and Milestones

| Sub-component | Main Activity | M18 | M24 | M30 | M36 |
|---|---|---|---|---|---|
| Kubernetes (Module Runtime) | Upgrade to latest version and adapt changes<br>Integrate with Containerisation module<br>Adapt the Module Helm Chart base templates | 🟦 | | | |
| Kubernetes UI | Upgrade to latest version and adapt changes<br>Adapt look and feel | 🟦 | | | |
| Management and Monitoring Controller | Upgrade to latest version and adapt changes | 🟦 | | | |
| Ingress Controller – Service Mesh | Upgrade to latest version and research on using Service Mesh<br>Adapt the Module Helm Chart base templates | 🟦 | | | |
| Cert Manager | Upgrade to latest version and adapt changes<br>Adapt the Module Helm Chart base templates | 🟦 | | | |
| Ingress - Gateway | Upgrade to latest version and research on using Gateway API<br>Adapt the Module Helm Chart base templates | 🟦 | | | |
| Storage Manager | Test, Install and Configure NFS Provider<br>Adapt the Module Helm Chart base templates | 🟦 | | | |
| DS2 Connector | Research on EDC and Tractus-X connectors<br>Integrate them into the IDT Stack<br>Develop required extensions for identity, catalog and research on possible other extensions required | | | 🟦 | |
| Dashbutton and Local Identity | Adapt the dashbutton and local identity for DS2<br>Further enhancements to the dashbutton<br>Dash Button integration in the IDT UI | | 🟦 | | |
| Tier 0: Blockchain & Blockchain API: | Integration with DS2 block chain module | | | 🟦 | |
| Tier 3: Integration with identity system and catalog | Integration with DS2 block chain module | | | 🟦 | |
| **Table Total/DOA Task Total/Resilience** | **Comments:** Blockchain, Ontology are sacrificial. Anticipated no new methods are needed | | | | |

### 5.1.12 Open Issues

The following table summarise open issues/uncertainties that need to be resolved during the next stages or implementation.

| Issue | Description | Next Steps | Lead or Related Component |
|---|---|---|---|
| Connector | From theory→research→design→implementation of service upgrade to chosen connector | Further connector research | None |
| Other modules integration | Still need to review the implementation of blockchain, identity system and catalog for integration | Wait for final design/implementations. Catalog and Identity: Ideally will be using current connector integration adapted to DS2 needs.<br><br>Blockchain: After discussion, it could be done using Prometheus as an intermediate step | WP3 Blockchain WP4 Catalog WP6 Identity System |

### 5.1.13 Additional Information: Dash Button

The Dash button is a web component that needs to be integrated in all DS2 module frontends and provides the following primary features:

- Single Sign On: This is optional and depends on whether the IDT local identity component is used or not. The idea is to provide local authn/authz in an IDT, most likely integrated with DS2 Trust system, where the Dash Button will be the abstraction layer for all frontend developments to authenticate and authorize users. This is accomplished by the Dash Button re-directing to the local identity single sign on feature in order to provide the authentication, and also connecting to the local identity system retrieving the necessary tokens and creating a local cookie that the frontend can later use for authorization.
- Module Navigation: the dash button will also keep track of the different DS2 modules running on the IDT and their URLs, that means, the DS2 modules that have been deployed in a participant IDT, allowing the participant to be able to navigate and connect to all the DS2 modules available using a shortcut menu style that is added in the UI of every module.

Based on the Stencil.js framework, it ensures consistent support across all modern frontend frameworks with minimal configuration and simple integration.

To integrate the Dash Button into a frontend:

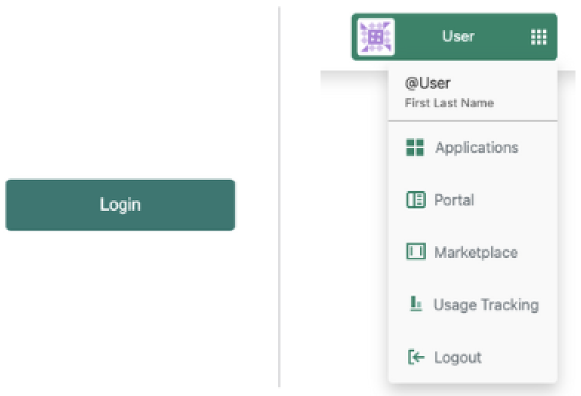
- First import the corresponding Javascript library in the html file:

```
<script type="module" src="https://unpkg.com/dash-button-web@0.0.12/dist/esm/web-component.js"></script>
```

- Then, place the dash button component anywhere in the web application UI, ie. at the right of the top navigation bar. For minimal configuration, you need to pass the keycloak-uri, realm, and client-id attributes as shown:

```
<dash-button keycloak-uri="HTTPS://KEYCLOAK_URL" realm="REALM" client-id="CLIENT_ID" show-post-login-text="false"></dash-button>
```

The Dash Button looks like the following image before and after a participant is logged in the IDT.



**Before and after authentication**

There are more specific configuration options that are explained in more detail in the corresponding DS2 GitHub repository.

There are also specific instructions on how to work with the cookie and the tokens retrieved and configured by the dash button.