Pulsar DNS VGU

subtitle

Document Version:	0.9
Document Status:	CONCEPT
Document Date:	July 2024

Preface

This preface provides an overview of this document. It includes the purpose and scope of this document, used references and a reading guide to this Software Architecture Document (SAD).

Goal

The goal of this Software Architecture Document (SAD) is to describe the Software Architecture related to the Basic Data Infrastructure (BDI) usage in the Digital Infrastructure Logistics (DIL) "Vertrouwde Goederen Uitgifte" (VGU) proving ground. The SAD should give the stakeholders and people designing and developing the system a clear overview of the assumptions and mechanisms.

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. The document is limited to that which is relevant from an architectural point of view for the development of the information system and should not be viewed as a complete collection of analysis and/or design elements

This SAD shows the principles based on which the software architecture has been designed and the choices available based on those principles in relation to the software to be developed and the products to be used.

The document has multiple objectives in relation to the above:

- It shows how principles of the architecture can be realised;
- It supports the discussion and decisions for making the choices within the software architecture;
- It generates the documentation for the maintenance of the system.

Reading guide

Chapter 1 (Architectural Representation) describes how the architecture is displayed. Subsequently, chapter 2 (Architectural Goals, Assumptions and Constraints) describes the goal and according to which starting points, constraints and assumptions the architecture has been developed and where it has been derived from.

The functionality that is viewed as being the determining factor for the architecture is described in chapter 3 (Use Case View). Chapter 4 (Logical View) describes the structure and organization of the design of the system. Chapter 5 (Implementation View) enumerates subsystems, their organization, layering and dependencies. The process organization of the system is outlined in chapter 6 (Process View). How the system can and/or will be distributed across physical systems is depicted in chapter 7 (Deployment View). A separate Security View in chapter 9 describes all security aspects of the design. The Data View in chapter 8 describes the persistent data storage perspective of the system.

Chapters 10 (Implementation) describes how the system has been implemented, the results of the tests described throughout the document, and the implementation learnings.

Finally, terms and abbreviations used in this document are mentioned in Appendix A.

1 Architectural representation

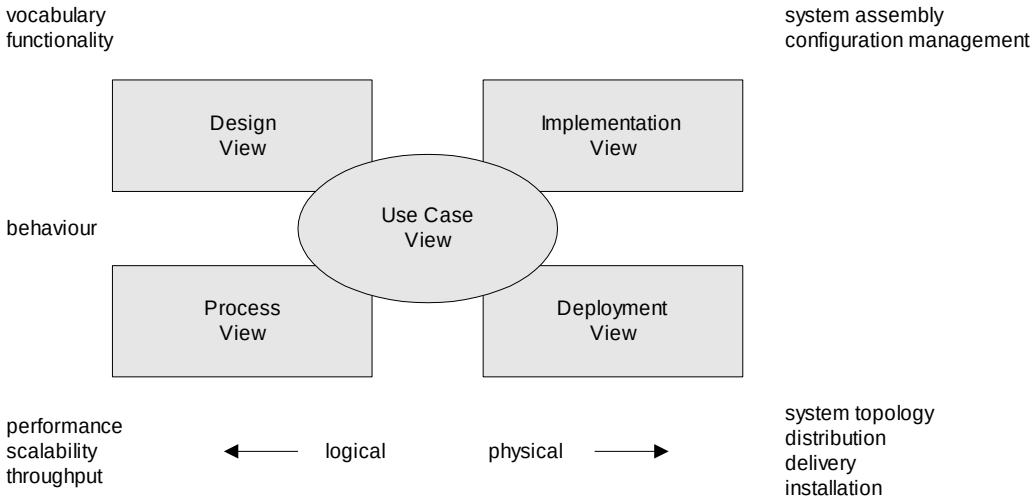
This chapter describes what software architecture is for the current system, and how it is represented with several views. The views that are necessary are enumerated, and for each view the types of model elements it contains are explained.

1.1 View Model of Architecture

The method of design used for the visualisation of the blueprints of the system is based on the “4+1 View Model of Architecture”, which is based on [KRUCHTEN1995]].

A viewpoint, as its name implies, is a notional position from which some aspects or concerns about the system (or the set of models representing the system) are made visible, implying the application of a set of concepts and rules to form a conceptual filter. The term "perspective" is used similarly, to describe a way of viewing and understanding models that best serves the many different orientations and concerns of the diverse stakeholders. Views are projections of models, which show entities that are relevant from a particular viewpoint or perspective.

This document describes the architecture as a series of views: Use Case View, Logical View, Process View, Deployment View, Implementation View and (optionally) Data View. Each view has a focus on a specific aspect of the system and is targeted for a specific audience..



Figuur 1: ·View Model of Architecture

- A **Use Case View** is a subset of the Use-Case Model, presenting the architecturally significant use-cases of the system. It describes the set of Use Cases that represent some significant, central functionality. It also describes the set of Use Cases that have a substantial architectural coverage or that stress or illustrate a specific, delicate point of the architecture.
- A **Logical View** is a subset of the Design Model, which presents architecturally significant design elements. It describes the most important classes, their organization in packages and subsystems, and the organization of these packages and subsystems into layers.
- An **Implementation View** describes the decomposition of the software into layers and implementation subsystems in the implementation model.

- The **Process View** describes the tasks (processes and threads) involved in the system's execution, their interactions and configurations, as well as the allocation of objects and classes to tasks. This view is relevant to the actual sizing of the system.
- A **Deployment View** describes the physical network and hardware (server) configurations on which the software is deployed and run. It also describes the allocation of tasks (from the Process View) to the physical nodes.
- A **Data View** describes the persistent data storage perspective of the system. This view only used when there is persistent data, or the translation between the Design Model and the Data Model is not trivial.
- Other views, like a **Security View**, may be added when certain specific aspects of the design are relevant to stakeholders or are architecturally significant.

1.2 Architecture blueprints

The graphical depiction of an architectural view is called an architectural blueprint. For the various views described above, the blueprints are composed of the following diagrams from the Unified Modeling Language (UML):

View	Diagrams
Use Case View	Use Case diagrams depicting use cases, actors, and ordinary design classes; sequence diagrams depicting design objects and their collaboration.
Logical View	Class diagrams, state machines, and object diagrams.
Implementation View	Component diagrams.
Process View	Class diagrams and object diagrams (encompassing task-processes and threads).
Deployment View	Deployment diagrams.
Data View	Data model diagrams and object diagrams.

Tabel 1: ·Architecture blueprints

2 Architectural Goals, Assumptions and Constraints

2.1 Goals

This project aims at augmenting the existing “Vertrouwde Goederen Uitgifte” demo

The goal of the project is to implement a logistics event-based approach into the existing demo environment based on Apache Pulsar and DNS service discovery. This project aims to enhance the current system capabilities and align with best practices in event-driven message system deployment and service discovery.

The BDI Building Block “Pub/Sub Service” will be a central part of this demo. It is envisioned that additions to this building block and its events will be made.

Additionally the BDI Building Block ‘Semantics’ is used as reference to develop the pulse and event data model.

3 Use Case viewpoint

3.1 Architectural significant use cases

The following diagram shows the context of the Trusted Issuance of Goods use case, with a focus on the Gate-in and Gate-out event.

Figuur 2: Use case diagram

3.1.1. Use case Warehouse employee

3.1.1.1. User story

As a Warehouse employee **I want** to send a notification to the Shipper when the goods depart from the warehouse **so that** I can inform them that the responsibility of the goods has been handed over to the transport company and **so that** I can directly send the invoice

3.1.1.2. Test scenario / Acceptance criteria

1. An order is created, and send to the WMS & TMS.
2. As Warehouse Employee I mark the shipment as departed.

In the ERP system the status of the specific order must change to “Departed” without any user interaction.

3.1.2. Usecase Sales agent

3.1.2.1. Userstory

As a Sales agent **I want** to get a notification when the goods depart from the warehouse **so that** I can focus on the orders that are delayed, and **so that** I can inform my customer about the expected arrival date of their goods.

3.1.2.2. Test scenario / Acceptance criteria

In the order execution process a warehouse employee can set the status of the order to ‘Departed’ in his WMS. In the ERP system the status of the specific order must change to ‘Departed’ without any user interaction.

The WMS must send a Gate-out Event to the ERP system.

3.1.3. Usecase Transport Planner

3.1.3.1. Userstory

As a Transport Planner **I want** to get a notification from the loading address when the truck has finished loading **so that** I can escalate on the loading locations where the truck is waiting too long

3.1.3.2. Test scenario / Acceptance criteria

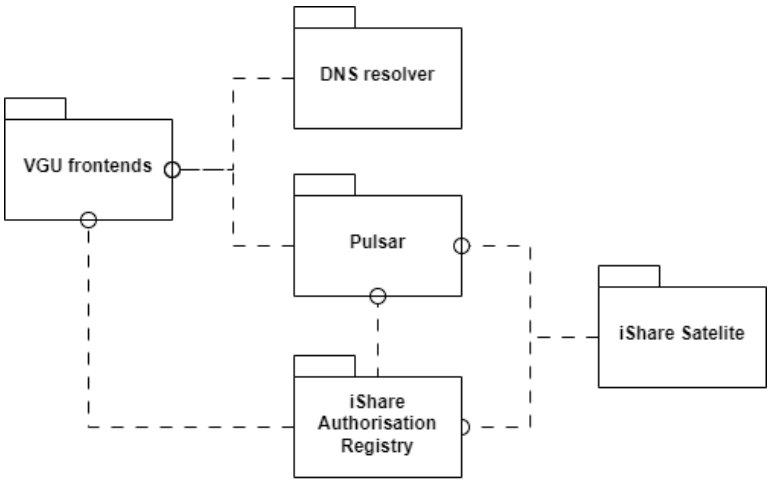
In the order execution process a warehouse employee can set the status of the order to ‘Departed’ in his WMS. In the TMS the status of the specific order must change to ‘Departed’ without any user interaction.

4 Logical View

4.1 System overview

4.1.1. High Level Componenten overzicht

The event system consists out of a message broker, Apache Pulsar in this case, an Authorization Registry, a Participant, and a DNS discovery mechanism. The pulsar broker and the VGU frondends share the same Authorisation registry.



Figuur 3: System overview

Componentname	High Level description
VGU frontend	The demo frontends for the WMS, TMS and ERP systems.
Pulsar	The Apache Pulsar event broker
iShare compliant Autorisation Registry	An iShare compliant Authorisation Register provided by iShare or Poort8

Tabel 2: High Level Componenten

4.1.2. Interfaces overview

4.1.2.1. Apache Pulsar interface - URL

Exchanging events via the event broker happens via the standardised WebSocket protocol (IETF rfc 6455). The connection URL is Apache pulsar specific:

- wss://*broker-service-url*:*port*/ws/v2/:role/persistent/:*tenant*/:*namespace*/:*topic*

url section	value

broker-service-url	“apachepulsar-container-app.orangebush-8d078598.westeurope.azurecontainerapps.io”
port	“433” (can be left blank, as 433 is the default port for wss)
role	Either “producer” or “consumer”
tenant	“public”
namespace	The namespace is equal to the EORI of the owner of the topic. The namespace must start with “EU.EORI”
topic	The topic name depends on the usecase and can be chosen by the topic owner. The topic name must not start with “__”.

Example url: wss://apachepulsar-container-app.orangebush-8d078598.westeurope.azurecontainerapps.io/ws/v2/producer/persistent/public/EU.EORI.NLSMARTPHON/testtopic

4.1.2.2. Apache Pulsar interface – Headers

One header must be added when setting up the websocket connect:

Header name	**Header value**
Authorization	“Bearer {access token}” The access token should be obtained from the token endpoint, as describe in section 9
Delegation-trail	“{EORI number}” The EORI number of the party who delegated their access rights to the topic. This value is used to contact the AR for validating if the client is allowed to connect to the topic on their behalf.

4.1.2.3. Apache Pulsar interface – Message format

Messages to publish must be placed in an envelope, named the Apache Pulsar Message standard. The Broker will only delivery the content of the envelope to the consumer.

```
{
    'payload' : ‘content’
}
```

Figure 4 - Apache Pulsar Message Standard

4.1.3. External User Interface overview

The events in the VGU demonstration will be sent from, and received by the VGU frontends developed by DIL proving ground. DIL proving ground will implement the event exchange in the ERP, WMS, and TMS system.

User Interface	High Level omschrijving
ERP	A demonstration version of an Enterprise Resource Planning system. This system is used by the owner of the goods to keep track of orders.
WMS	A demonstration version of a Warehouse Management System. This system is used by the warehouse to keep track of the orders to be fulfilled.
TMS	A demonstration version of a Transport Management System. This system is used by the transport company to keep track of the transport order to be executed.

Tabel 3: Externe User Interfaces

5 Implementation View

The implementation view details the high level system architecture, with a focus on the Apache Pulsar Broker. The Apache Pulsar Broker consists out of 3 components, the Core, the websockets interface, and the iShare authentication plugin. The core and the websockets interface are standard components provided by the Apache foundation. The iShare authentication plugin is a custom extension following the Apache Pulsar extension protocol.

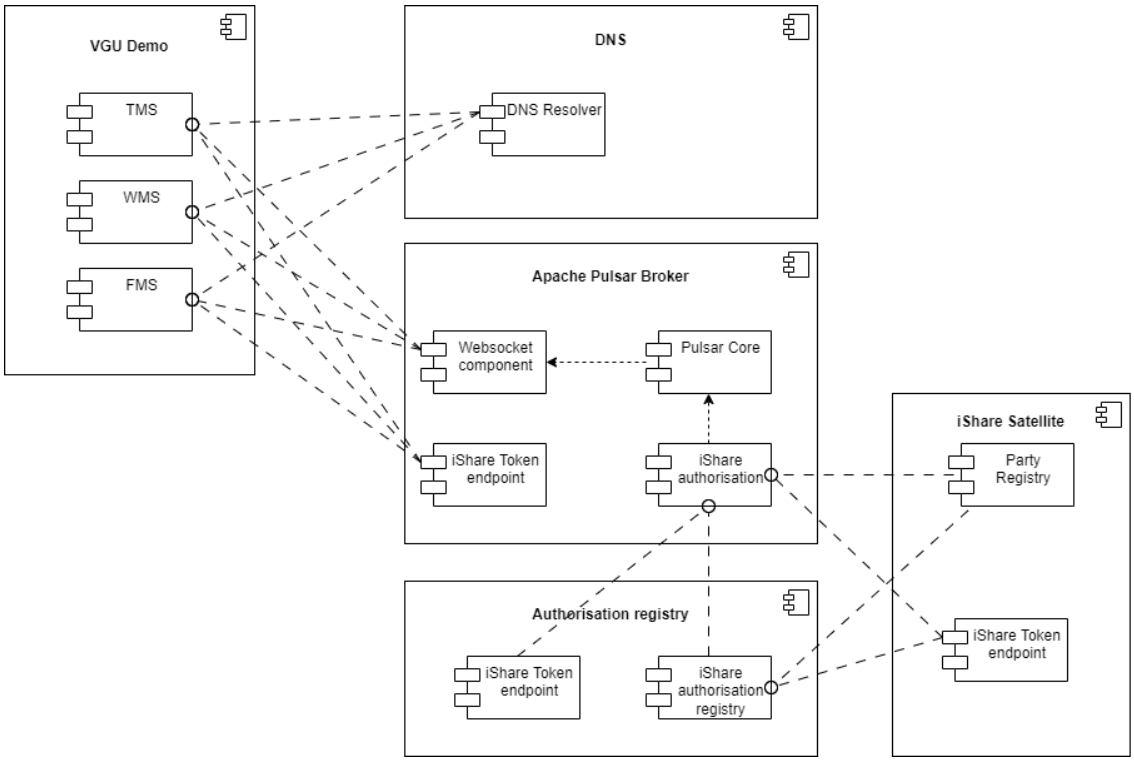


Figure 5 - Implementation view

6 Process view

The process flow in Figure 6 describes the sequence of steps to be taken to include event messaging in the existing VGU demo. This includes setting up topics, subscribing and publishing to topics, and setting up the Authorization policies. All steps above the horizontal bar ("Goods are loaded…"), are required to setup the event messaging channels, in order to be able to actually send and receive events. Sending and receiving events happens below the horizontal bar, this is the part which should happen automatically during the demonstration.

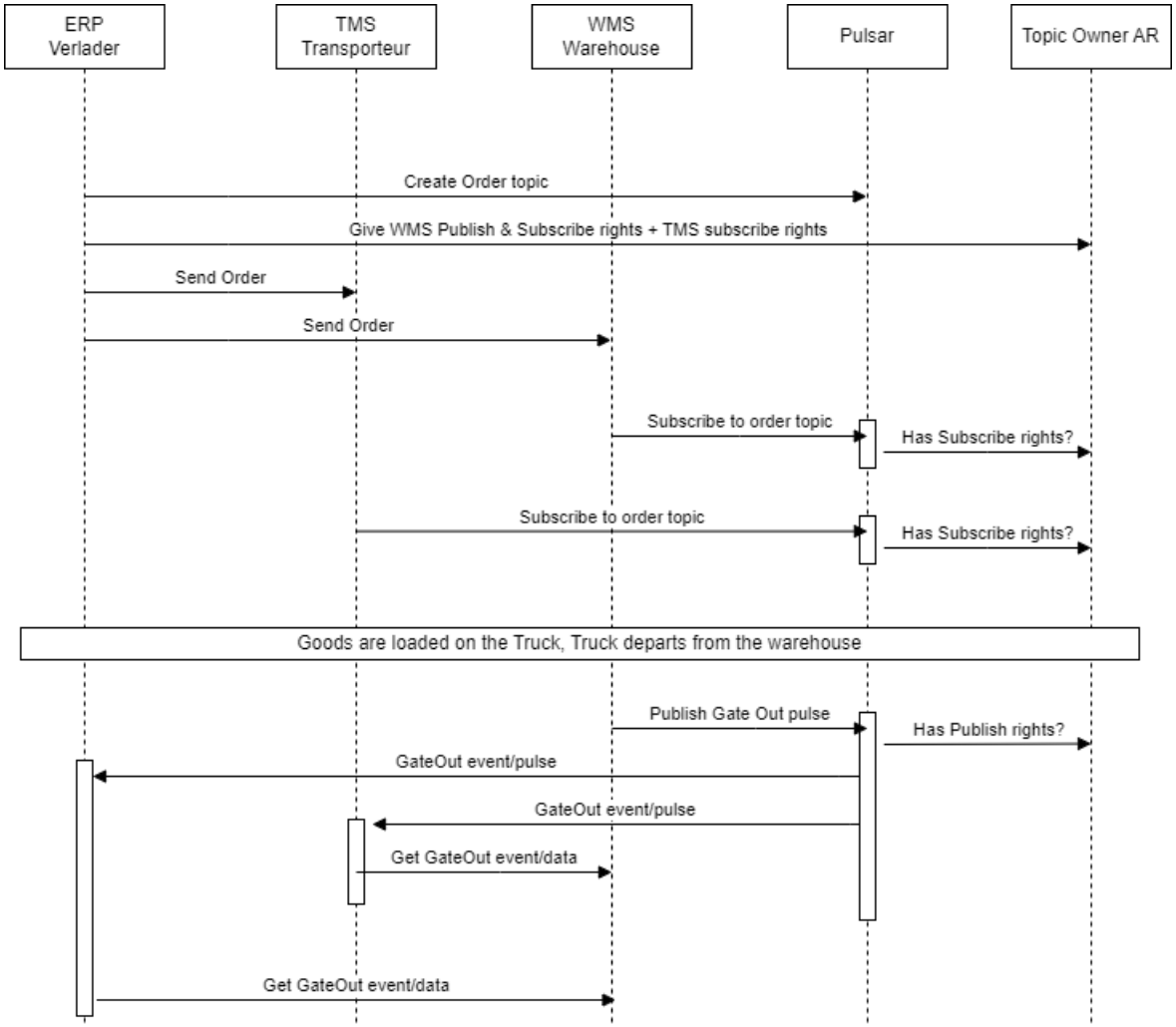


Figure 6 - Process overview

7 Deployment view

7.1 Demo environment

The demo components are deployed on the Azure tenant of Topsector Logistics. This tenant is not hardened, and therefore not suitable for sending sensitive data. It is only designed to be used for demonstration purposes, with unsensitive data. Azure Container Apps is used for running the various components, which are packaged in docker images.

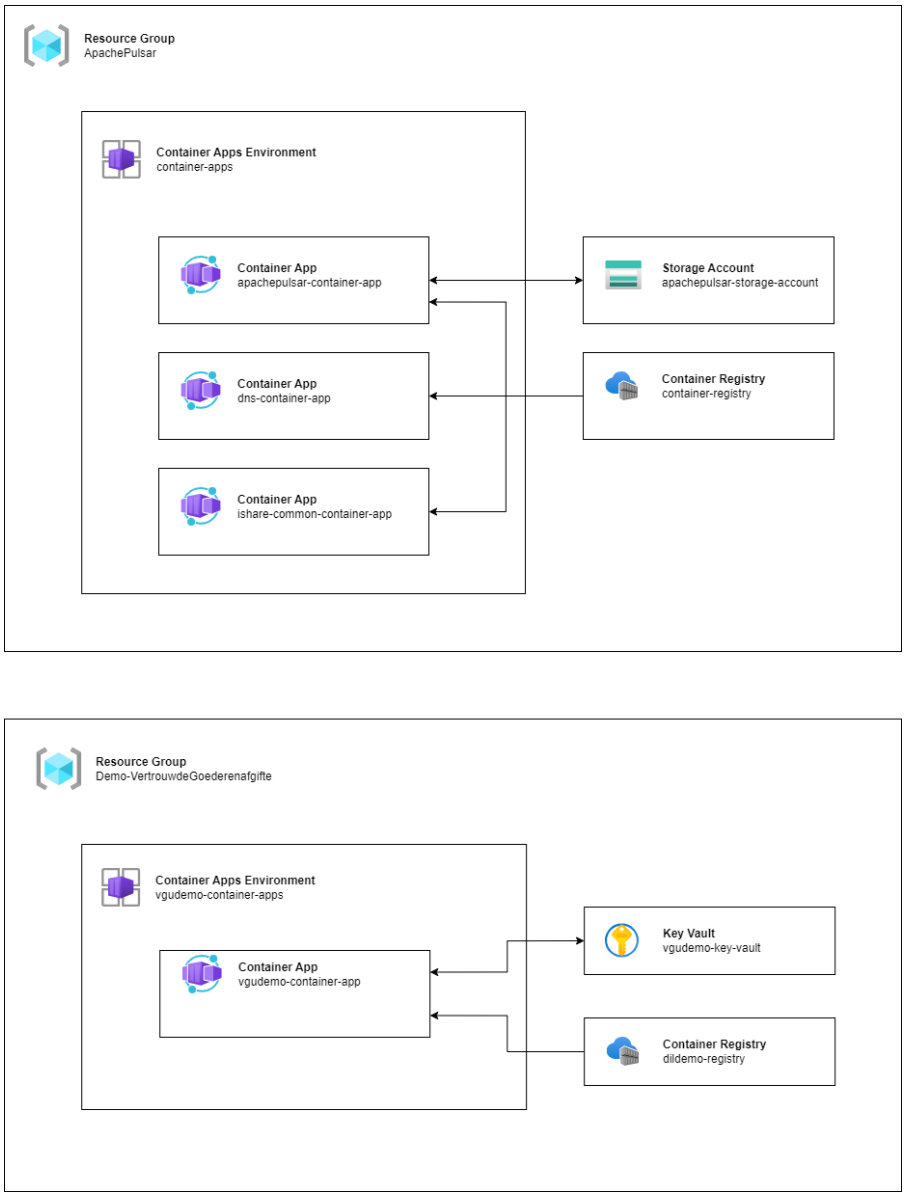


Figure 7 - Azure deployment view

8 Data view

8.1 Pulse data model

{

"@context": {dcat": "http://www.w3.org/ns/dcat#"},

"@type": " dcat:Dataset",

"dcat:downloadURL": "*https://example.nl/to/1234*"

}

8.2 Event data model

Data models for event exchange in the logistics sector are scarce. In existing datamodels for the logistics sector, such as UNCEFACT, DCSA and OTM, events are included, but only as an additional element to an existing concept. One universal event data standard which is investigated is the open EPCIS standard. EPCIS is initiated by GS1 for exchanging supply chain related events. In the VGU demo usecase this data model will be used to exchange events.

```
{
    "@context": [
      "https://ref.gs1.org/standards/epcis/2.0.0/epcis-context.jsonld",
      {"bdi": "http://www.bdi.org/bdivoc/",
    "gs1": "https://www.gs1.org/voc/"}
    ],
    "type": "EPCISDocument",
    "schemaVersion": "2.0",
    "creationDate": "2024-07-11T15:03:17.32Z",
    "epcisBody": {
      "eventList": [
        {
          "type": "ObjectEvent",
          "eventTime": "2024-07-11T15:24:15+02:00",
          "eventTimeZoneOffset": "+02:00",
          "eventID": "ni:///sha-
1;d38c4adc56affffbf842322d8414a6ad6e86d30a?ver=CBV2.0",
          "epcList": [
            "https://opd.org/OPDU205271422G1"
          ],//Container nummer
          "action": "OBSERVE", //Mist in flow laurens #35
          "bizStep": "departing",
          "disposition": "in_transit",
          "sourceList": [
            {
                "type": "location",
                "source": {
```

```
                    "@type": "gs1:Place",
                    "gs1:additionalLocationID" : {
                        "@type": "gs1:LocationID_Details",
                        "gs1:locationID_Type":"9F462958+HFJ",
                        "gs1:locationID": "gs1:LocationID_Type-
OPEN_LOCATION_CODE"
                    }
                }
            }
          ],
          "destinationList": [
            {
                "type": "location",
                "source": {
                    "@type": "gs1:Organization",
                    "gs1:organizationName": "Winkel van Sinkel",
                    "gs1:additionalOrganizationID" : {
                        "@type": "gs1:OrganizationID_Details",
                        "gs1:organizationID":"NL012345678",
                        "gs1:organizationID_Type": "gs1:OrganizationID_Type-
EORI"
                    }
                }
            }
          ],

          "bizLocation": {
            "id": "https://securestorage.nl/warehouse1"
          },
          "bizTransactionList": [
            {
              "type": "po",
              "bizTransaction": "https://smartphone.nl/po12345"
            }
          ],
          "bdi:eventClassifierCode" : "bdi:ACT"
        }
      ]
    }
}
```

Figure 8 - EPCIS GateOut event sample

8.3 Topic structure

A topic can be compared with a whatsapp group chat. Everyone in the group can read all the messages. Only people who are authorized can send messages to the entire group. The scope, and therefore the number of participants, is mostly determined by the confidentiality of the information exchanged. The more confidential the data is, the smaller the group size should be. Opposed to the whatsapp analogy, in the Apache pulsar implementation, subscribers of a topic do not know who the other subscribers are. Additionally, subscribers also do not know who the publisher is of a message. Only when additional information is collected at the source, via an iShare secured API endpoint, the identity of the requesting party becomes known.

In the scenario where all information is send via the event, and no additional information needs to be collected at the source, both the publishers and the subscribers can remain anonymous to each other. Only the host of the broker/topic is aware of the senders and receivers of the event.

The scope of a topic can be different in every usecase. For the VGU demo usecase a number of options can be considered:

Topic scope subscribers	Usecase
All organizations	Low sensitive data of which every organization needs to be made aware of
One organization	Highly sensitive data which is only relevant for one specific party
All organizations related to one specific customer order	Order execution orchestration, in which every party related to this order is allowed to see the chain of event triggers from start to finish
All organizations related to one product stream	Order execution orchestration for which there are fixed suppliers for a long time period.

For the VGU usecase, different suppliers need to exchange information with each other. For that reason it is advisable to keep the scope of the topic as small as possible. In order to keep things manageable, one should start with One topic per customer order, where every supplier who needs to interact with that order is a part of.

8.4 Access policy shape

8.4.1. Direct access policy

The iShare authorisation plugin checks at the Authorisation Registry of the owner of the topic, if the requesting client is allowed to publish or subscribe. While the communication between the authorisation plugin and the AR is standardised by iShare, the content of the delegation is not. The access policy in the AR must conform to the example shown in Figure 9. in order to successfully allow the client to publish or subscribe to a topic.

```
{
  "delegationEvidence": {
    "notBefore": 1721285715,
    "notOnOrAfter": 1847447885,
    "policyIssuer": "EU.EORI.policyIssuer",
    "target": {
      "accessSubject": "EU.EORI.client#EU.EORI.topicOwner#topicName"
    },
    "policySets": [
      {
        "policies": [
          {
            "target": {
              "resource": {
                "type": "http://rdfs.org/ns/void#Dataset",
                "identifiers": [
                  "EU.EORI.topicOwner#topicName"
                ],
                "attributes": [
                  "*"
                ]
              },
              "actions": [
                "BDI.subscribe",
                "BDI.publish"
              ]
            },
            "rules": [
              {
                "effect": "Permit"
              }
            ]
          }
        ]
      }
    ]
  }
}
```

Figure 9 - AR Access policy

8.4.2. Delegated access policy

After the topic owner has given access to a client to subscribe to a topic, the client is able to delegate this access to another party. The client will need to place an access policy in its own AR according to the format of Figure 10. The party who connects to the topic on behalf of the client only needs to mention the clients EORI number in the Delegation-trail header (section 4.1.2.2). No other changes are required.

```
{
  "delegationEvidence": {
    "notBefore": 1721285715,
    "notOnOrAfter": 1847447885,
    "policyIssuer": " EU.EORI.client",
    "target": {
      "accessSubject": "EU.EORI.thirdParty#EU.EORI.topicOwner#topicName"
    },
    "policySets": [
      {
        "policies": [
          {
            "target": {
              "resource": {
                "type": "http://rdfs.org/ns/void#Dataset",
                "identifiers": [
                  "EU.EORI.topicOwner#topicName"
                ],
                "attributes": [
                  "*"
                ]
              },
              "actions": [
                "BDI.subscribe",
                "BDI.publish"
              ]
            },
            "rules": [
              {
                "effect": "Permit"
              }
            ]
          }
        ]
      }
    ]
  }
}
```


Figure 10 - Delegated AR Access policy

9 Security view

9.1 Azure tenant

The Azure tenant of Topsector Logistics is used for the deployment of the Apache Pulsar broker. This tenant is not hardened, which means that no additional security measure are taken to monitor and protect the deployment from malicious activities. Therefore it is important that no sensitive data is used in this demonstration environment.

9.2 Authentication Pulsar

Authentication on the Pulsar Broker happens via the iShare protocol. The access token of every new connection request is checked if it was issued by the broker. If the access token is signed with the private (iShare) key of the Broker, then the authentication was successful. The access token has an expiration time. A few minutes before the expiration time has passed, the broker will ask the client to send a new access token to the broker. If the client fails to send a new access token, the connection will be terminated.

9.2.1. Authentication usecase

9.2.1.1. User story

As an Event Broker manager **I want** to allow only connections with an access token issued by the organisation representing the Broker manager **so that** unauthorised organisations do not get access to the broker, and do not consume resources by checking AR's.

9.2.1.2. Test scenario / Acceptance criteria

Obtain an access token from the Apache Pulsar Broker manager (happy flow).

Obtain an access token signed by another organisation than the Broker Manager, and attempt to connect to the broker.

Validation:

1. Only connection requests with an access token signed with the private key of the organisation representing the Broker manager should be allowed
2. Connection requests with an access token signed with the private key of any other organisation should be denied

9.2.2. Token expiration usecase

9.2.2.1. User story

As a Event Broker manager **I want** to terminate connections of which the access token has expired **so that** organisations who are unable or unwilling to obtain a new access token do not keep access to the broker.

9.2.2.2. Test scenario / Acceptance criteria

1. Setup a connection to the broker

2. Ignore requests from the broker to renew the access token, and wait until the token expires.

Validation: If all is correct, the client is unable to send or receive messages to the broker after the expiration time of the token.

9.3 Authorization Pulsar

Authorization on the Pulsar Broker happens via the iShare protocol. The topic owner controls the AR which contains the rights to a specific topic. The broker must check for every new subscribe or publish connection in the AR of the topic owner whether the requesting party is allowed to publish/subscribe or not. In a second iteration the Broker should also be able to check whether the requesting party has access via a delegated manner.

9.3.1. User story

As a Topic owner **I want** to control in a digital manner who has access to publish and subscribe to the topic **so that** unauthorised organisations do not get access to the topic information, and **so that** I can issues and revoke permissions seamlessly integrated with my business process.

9.3.2. Test scenario / Acceptance criteria

Only organisations for which a policy exists with their EORI number, read or write permissions and topic name are allow to publish or subscribe to a topic.

9.4 Encryption

All traffic over the public web is encrypted (TLS v1.2 & v1.3). Internal traffic, within the a cloud tenant is not encrypted. The websocket connection is established by using Websockets (ws://) over TLS (wss://). The API endpoint is accessible via HTTPS. The certificate management is handled natively by Azure Container Apps Environment.

10 Implementation

10.1 Implementation guide

Setting up an event publisher and subscriber is relatively simple. Setting up an Apache pulsar cluster with iShare authorisation module and websockets is a more complex.

10.1.1. Setting up a producer/consumer scenario

The producer/consumer scenario consists out of two parties that exchange events. One party is the “Topic Owner”, and in this scenario also the message Producer. The other party is a Subscriber.

Steps to take top setup the producer/consumer scenario:

1. The topic owner needs to determine the Broker where the topic will be located and managed.
2. Every participant needs to be registered in an iShare Satellite
3. The location of the AR of the Topic owner needs to be specified in the iShare Satellite
4. The topic owner should setup access rules for allowing the other parties to publish or subscribe to their topics. This is done in the AR of the topic owner, according to section 8.4.1.
5. The Publishing and Subscribing party need to establish a websocket connection to the pulsar broker, according to section 4.1.2.
6. Events can be published to the topic, according to section 8.1.
7. The subscribing party will receive the event.

10.1.2. Setting up a producer/consumer scenario where access is delegated to a third party

The producer/consumer scenario with delegated access consists out of three parties. One party is the “Topic Owner”, and in this scenario also the message Producer. The second party is delegating party. This party receives access to the topic from the Topic Owner. The delegating party gives the third party access to the topic on their behalf. The third party is the Subscriber who received access to the topic via the delegating party.

Steps to take to setup the scenario:

1. The topic owner needs to determine the Broker where the topic will be located and managed.
2. Every participant needs to be registered in an iShare Satellite
3. The location of the AR of the Topic owner and of the Delegating party needs to be specified in the iShare Satellite
4. The topic owner should setup an access rule for allowing the delegating party to subscribe to their topic. This is done in the AR of the topic owner, according to section 8.4.1.
5. The delegating party should setup an access rule for allowing the third party to subscribe to the topic of the topic owner. This is done in the AR of the delegating party, according to section 8.4.2.

6. The Publishing and Subscribing party (Topic owner and Third party respectively) need to establish a websocket connection to the pulsar broker, according to section 4.1.2. The Third party needs to specify the EORI number of the delegating party in the “Delegation-trail” header (section 4.1.2.2).
7. Events can be published to the topic, according to section 8.1.
8. The subscribing party will receive the event.

10.1.3. Setting up an Apache Pulsar Broker

Either use a SaaS broker, or setup your own.

A docker file to set up your own broker can be found at the Github repository: https://github.com/Topsector-Logistiek/Apache-Pulsar-Auth-Plugin

10.2 Test results

Userstory 3.1.1 up until 3.1.3 feature UI elements as will be created by the BDI proving ground team, and are to be manually asserted.

Userstory 9.2.1 is covered in the unit tests which are part of the code base. All unit tests resulted in a positive outcome.

```
$ pyton test.py
----------------------------------------------------------------------
Ran 2 tests in 13.135s

OK
```

Userstory 9.2.2 is manually asserted, and functioned according to the acceptance criteria.

```
$ python client_secure_consumer.py
Received msg:
{"type":"AUTH_CHALLENGE","authChallenge":{"challenge":{"authMethodName":"token","au
thData":"token"},"protocolVersion":0}}

Send message: {"type": "AUTH_RESPONSE", "authResponse": {"clientVersion": "v21",
"protocolVersion": 21, "response": {"authMethodName": "token", "authData":
"[JWT_ACCESS_TOKEN]"}}}
```

10.3 Deployment learnings

A number of realisations have been formed. These are described in this section.

10.3.1. Topic access revocation

When a client subscribes to a topic, the broker checks if the client has access to the topic. The broker is not notified when the access is revoked in the AR. If this happens after the client has subscribed, he will remain

connected. The pragmatic solution for this issue is to drop the connection after the expiration time of the access token has passed. Requiring the client to setup a new connection, where the access policy is re-checked.

Design requirements for a new event broker implementation should include the following:

- The broker needs to request the client to send a new access token in order to keep the connection alive when the old access token nears its expiration time.
- The broker needs to check the access policy again after the expiration time of the access policy has passed ("notValidAfter"). If the newly requested access policy is expired, access to the described resource should be stopped.
- Optionally the broker can periodically check if the access policy has not been revoked. If this is the case, access to the described resource should be stopped.

10.3.2. Consumers and producers can remain anonymous to each other

One realisation made is that producers and subscribers remain anonymous to each other. Only the topic owner knows who all the publishers and subscribers are. The identity of other producers and subscribers can only be deduced based on the content of the messages sent, and based on the actions performed such as requesting an others party API.

10.3.3. Delegating access

Delegating access to a third party is done differently in the BDI provinggground environment than is documented in the iShare documentation. The current AR's apparently do no support delegating access (in a pragmatic manner). It should be determined whether delegating access in the way as implemented in Section 10.1.2 will become an accepted way to delegate access.

10.3.4. Authorisation Registry Search

The iShare implementation requires a lot of requests, and has a hard dependency on the Satellite. It takes quite some time and resources to query the Satellite for determine the location of a Parties Authorisation Register. The Satellite was down a couple of times during the development of the demonstration, which caused the event exchange to stop working completely. This shows the vulnerability of the system. It is advisable to implement the DNS system for storing the location of the AR of parties. This increases performance, and reduces risk of down time.

This alternative has been implemented in the following way under the sample domain "EU.EORI.PRECIOUSG.bdi-demo.nl":

The main DNS record were all the bdi references can be found:

_bdi.EU.EORI.PRECIOUSG.bdi-demo.nl

The associated TXT-records let us know that they have an ishare compliant authorization registry

_bdi.EU.EORI.PRECIOUSG.bdi-demo.nl. 3600 IN TXT

'_ishare._tcp._bdi.EU.EORI.PRECIOUSG.bdi-demo.nl'

We zoom in on the iShare authorization registry to find:

_ishare._tcp._bdi.EU.EORI.PRECIOUSG.bdi-demo.nl. 3600 IN TXT

'_ishare._tcp.authorisation-registry._bdi.EU.EORI.PRECIOUSG.bdi-demo.nl'

We find the server with the SRV-record:

_ishare._tcp.authorisation-registry._bdi.EU.EORI.PRECIOUSG.bdi-demo.nl. 3600 IN SRV

'1 1 443 ar.isharetest.net.'

10.3.5. iShare Authorisation Registry access subject limitation

The iShare AR only allows one access policy per “Access Subject”, resulting in workarounds with hashtags. It should be determined how to handle this.

Appendix A: Abbreviations

Abbreviation	Description
AR	Authorisation Registry
BDI	Basic Data Infrastructure
DIL	Digital Infrastructure Logistics
DNS	Domain Name System
ERP	Enterprise Resource Planning
SAD	Software Architecture Document
TMS	Transport Management System
UC	Use Case
WMS	Warehouse Management System