PoC Federated event distribution

basic data infrastructure

Topsector Logistiek

Colofon

PoC Federated event distribution

Author
H. Alblas

January 2024

Table of Contents

BDI: set of agreements

The BDI set of agreements gives parties a good balance between control (who is allowed to see or use what data?) and efficiency (automated handling), by means of:

- **Digital trust**
  - control of own data by 'accessing data at the source';
  - certainty about identity;
  - high security requirements;
  - contractual basis for computers that communicate with each other without human intervention.

- **Agreements about each other's 'language' and terminology to make automatic processing possible**
  - semantics.

- **Timely automatic alerts ('events')**
  - Getting a signal automatically when needed makes work very efficient;
  - If something has happened or changed in reality that may be relevant.

The last aspect, 'event-driven' communication, is another way of structuring the logistics of information exchange between IT systems of companies and government agencies.

Instead of sending messages ('fire and forget'), all parties involved receive a signal that something relevant has happened ('publish event'). That event contains metadata and a link to the source of the data.

The question was whether existing middleware for completing the publish/subscribe function is suitable, and can easily be applied in corporate environments across firewalls. The subsidiary questions were:
1 What are the core functional requirements in this context for exchanging events and streams?
2 How does existing open-source event and stream middleware score on these core functional requirements?
3 How is an event/stream modelled?
4 How does distributing events/streams with open-source technology work in a PoC setup together with iShare and the DIAC (Distributed Information Access Control) request-reply functionality?

The findings are:
- It is possible to use existing middleware to set up publish/subscribe event distribution that is suitable for the BDI.
- It is possible to have that run over normal open ports in corporate firewalls, greatly reducing the barrier to deployment.
- The Apache Pulsar WebSocket extension was not yet sufficiently developed to handle tokens securely. This triggered a change request on the Apache Pulsar WebSocket extension with which a WebSocket connection is closed when a token has expired. With this Pulsar Improvement Process, the modification becomes part of the official Apache Pulsar project.
- Apache Pulsar includes a component for archiving pulses. This can be investigated and could strengthen the irrefutability requirement further.

Table of contents

1 Introduction

1.1 BDI

The BDI set of agreements (www.bdinetwork.org) aims at easy, effective and controlled data sharing in business ecosystems ('federated data spaces').

In a highly developed economy, companies have a high degree of specialisation. In their network of relationships, they need to communicate with each other with high frequency to achieve the result, such as keeping intercontinental supply chains running smoothly. And they need to provide timely information to the government agencies that oversee this.

The BDI set of agreements gives parties a good balance between control (who is allowed to see or use what data?) and efficiency (automated handling), by means of:

- **Digital trust**
  - control of own data by 'accessing data at the source';
  - certainty about identity;
  - high security requirements;
  - contractual basis for computers that communicate with each other without human intervention.

- **Agreements about each other's 'language' and terminology to make automatic processing possible**
  - semantics.

- **Timely automatic alerts ('events')**
  - Getting a signal automatically when needed makes work very efficient;
  - If something has happened or changed in reality that may be relevant.

The last aspect, 'event-driven' communication, is another way of structuring the logistics of information exchange between IT systems of companies and government agencies.

Instead of sending messages ('fire and forget'), all parties involved receive a signal that something relevant has happened ('publish event'). That event contains metadata and a link to the source of the data.
Every party that receives the event evaluates it and can report back to the source if required. Based on the digital identity of the party making the report, the source can:

- Verify whether this specific party is involved, and in what role.
- Determine what data this party may then receive.

.
The iSHARE Trust Framework is the basis for this.

This approach is much more effective and efficient, and gives the data owner much more control.

In principle, this approach can use existing systems to distribute events based on publish/subscribe mechanisms. The question is whether the existing open-source solutions are suitable for large-scale professional application. One of the core questions here is whether the need for modifications to corporate firewalls can be avoided.

2 Issue

2.1 Main question

Does suitable open-source middleware exist for federated distribution of events and streams using publish & subscribe?

2.2 Subsidiary questions

1 What are the core functional requirements in this context for exchanging events and streams?
2 How does existing open-source event and stream middleware score on these core functional requirements?
3 How is an event/stream modelled?
4 How does distributing events/streams with open-source technology work in a PoC setup together with iShare and the DIAC (Distributed Information Access Control) request-reply functionality?

3 Elaboration

3.1 Functional requirements

What are the core functional requirements for exchanging events and streams?

When using the BDI, organisations should have the ability to subscribe to new relevant data from data owners.

For this purpose, existing and established open-source software packages were compared. The comparison criteria were drawn up from the premise that it should be possible to start using the software in a production environment 'directly'.

- **Development Status:** The software release must be stable, not in Beta.
- **OSS licence:** The software must be open-source and reusable.
- **Pub/Sub:** The publish/subscribe pattern must be supported.
- **WebSocket support:** WebSocket support for communication over standard ports.
- **Guaranteed delivery:** Guaranteed delivery must be supported.
- **Client footprint:** Client must be simple and lightweight.
- **Authentication support:** It must be possible to add iShare as an authorisation mechanism.
- **Client language:** The client must be available in multiple programming languages.

3.2 Score

How does existing open-source event and stream middleware score on these core functional requirements?

The following open-source event and stream middleware was compared:

- RabbitMQ
- ActiveMQ Artemis
- SwiftMQ
- Qpid
- AMQ Broker
- Mosquitto
- Amlen Broker
- NanoMQ
- EMQX
- Pulsar
- Cyclone DDS
- CATIA Magic
- CoreDX DDS
- Fast DDS
- HiveMQ
- Storm
- Kafka
- ZeroMQ

Easily being able to extend authentication with iShare is a hard requirement, and this eliminated many options. Apache Pulsar came out on top in the comparison. The full overview is given in the appendix.

3.3 Modelling

How is an event/stream modelled?

Events are modelled as being a pulse, in which it is only announced that data in the source system has been updated. The pulse contains enough information for automated data retrieval. Here, authorisation is checked in accordance with iSHARE protocols upon retrieval. For example, this makes it possible to transmit pulses to a wide audience, because only authorised parties can retrieve the actual data.

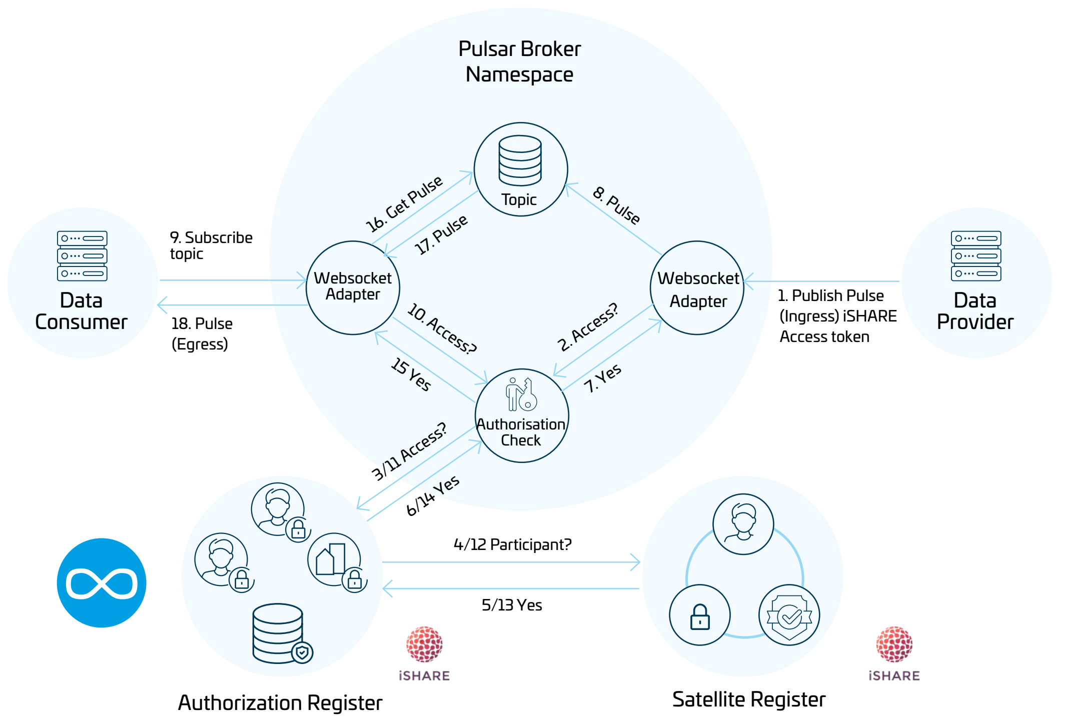
The pulse should be set up such that different technologies can be used for data retrieval. This reveals a contrast between a fully specified model on the one hand, and a formatting that can be used directly by a client on the other. It was decided to fully specify the model, in which the endpoints and data identifiers were modelled separately from each other.

The data model, and an example of this, can be found on Topsector Logistiek's GitHub

3.4 Distribution

How does distributing events/streams with open-source technology work in a PoC setup together with iShare and the DIAC request-reply functionality?

Figure
Distribution model



1. The data provider connects to the Pulsar Broker via WebSockets with an iShare token
2. The Broker performs an authorisation check to verify that the iShare token has publish permissions on the requested topic.
3. The authorisation plugin connects to the external authorisation register, and checks the authorisation
4. The authorisation register checks whether the organisation named in the iShare token is registered in the satellite (register of onboarded BDI participants)
5. The satellite confirms that the organisation is registered
6. The authorisation register confirms that the organisation from the iShare token is authorised to publish on the topic
7. The authorisation plugin confirms that the data provider may make the connection, and therefore may publish
8. The pulse from the data provider is persisted on the broker (and sent out to any subscribers)
9. The data consumer connects to the Pulsar Broker via WebSockets with an iShare token
10. The Broker performs an authorisation check to verify that the iShare token has subscribe rights to requested topic.
11. The authorisation plugin connects to the external authorisation register, and checks the authorisation
12. The authorisation register checks whether the organisation named in the iShare token is registered in the satellite
13. The satellite confirms that the organisation is registered
14. The authorisation register confirms that the organisation from the iShare token is authorised to subscribe to the topic
15. The authorisation plugin confirms that the data consumer is allowed to make a connection, and with that receive messages from the specified topic
16. The broker checks whether there are historical messages saved on the topic.
17. There is a saved message
18. The message is sent out via the WebSocket connection to the data consumer

3.5 Testing and use of standard ports in firewalls

Event exchange can be challenging in corporate IT environments due to firewalls with tight settings. For this reason, a test was conducted to run the event exchange through the ports for standard web traffic, namely port 443. The Apache Pulsar WebSocket extension was used for this. The disadvantage of the WebSocket extension is that it has fewer features than the native pulsar endpoint, which uses port 6651. One of the features was considered essential for the publication of this report, and suggested as an addition to the open source project.

In the test setup, the Data Consumer was present within CGI's corporate network, the Data Provider within Connekt's network, and the Broker within Azure. No modifications were made to CGI's and Connekt's firewalls. On the Azure instance, the default HTTP and HTTPS ports (80, 443) are open for incoming traffic. Using the WebSocket extension, it was possible to send and receive Pulses with the default firewall settings.

4 Findings

- It is possible to use existing middleware to set up publish/subscribe event distribution that is suitable for the BDI
- It is possible to have that run over normal open ports in corporate firewalls, greatly reducing the barrier to deployment.
- The Apache Pulsar WebSocket extension was not yet sufficiently developed to handle tokens securely. This triggered a change request on the Apache Pulsar WebSocket extension with which a WebSocket connection is closed when a token has expired. With this Pulsar Improvement Process, the modification becomes part of the official Apache Pulsar project.
- Apache Pulsar includes a component for archiving pulses. This can be investigated and could strengthen the irrefutability requirement further.
- The event model is modelled independently of technology. However, this adds complexity when it is used, because more parsing is required. In the next version, the model can be updated to reduce complexity.
- For a smooth demo, better integration is necessary on the source side. When data is modified, a correct pulse must be sent out automatically: that integration is now left out of consideration.

Appendix A

Opensource middleware score matrix

PRODUCT	STAN-DARD	SUPPLIER	STATUS	OSS LICENTIE	PAID ENTERPRISE VERSION	PUB/SUB	STREA-MING	WEB SOCKET	GUARAN-TEED SUPPORT	CLIENT FOOTPRINT DELIVERY	SECURITY	AUTHEN-TICATION SUPPORT	CLIENT SERVER LANGUAGE
RABBITMQ	AMQP 0.9	VMWare	GA	MPL 2.0	Yes	Yes	No	No	Yes	H+	SSL/TLS	SASL	Java, .NET, Python & more
ACTIVEMQ ARTEMIS	AMQP 1.0	Apache Foundation	GA	AL 2.0	No	Yes	No	No	Yes	H+	SSL/TLS	SASL	Java, .NET, Python & more
SWIFTMQ	AMQP 1.0	Apache Foundation	GA	AL 2.0	Yes	Yes	No	No	Yes	M	SSL/TLS	SASL	Java
QPID	AMQP 1.0	Apache Foundation	GA	AL 2.0	No	Yes	No	Yes	Yes	M	SSL/TLS	SASL	Java, .NET, Python & more
AMQ BROKER	AMQP 1.0	Redhat	GA	-	Yes	Yes	No	No	Yes	L	SSL/TLS	SASL	Java, .NET, Python & more
CYCLONE DDS	DDS	Eclipse Foundation, Zettascale, Adlink	GA	EPL 2.0	No	Yes	Yes	No	No	H	SSL/TLS	PKI Crypto	Java, .NET, Python & more
CATIA MAGIC	DDS	Dassault Systems	GA	-	Yes	Yes	Yes	No	No	L	SSL/TLS	PKI Crypto	Java
COREDX DDS	DDS	Twin Oaks Computing	GA	-	Yes	Yes	Yes	No	No	M	SSL/TLS	PKI Crypto	C, C++, C#, Java
FAST DDS	DDS	eProsima	GA	AL 2.0	Yes	Yes	Yes	No	No	H	SSL/TLS	PKI Crypto	C++, Python
HIVEMQ	MQTT 5.x	HiveMQ	GA	AL 2.0	Yes	Yes	Yes	Yes	No	M	SSL/TLS	SASL	Java, .NET, Python & more
MOSQUITTO	MQTT 5.x	Eclipse Foundation	GA	EPL 2.0	Yes	Yes	Yes	Yes	No	H	SSL/TLS	SASL	Java, Python
AMLEN BROKER	MQTT 5.x	Eclipse Foundation	Incubation	EPL 2.0	No	Yes	Yes	No	No	M	SSL/TLS	SASL	C
NANOMQ	MQTT 5.x	EMQ	GA	MIT	Yes	Yes	Yes	No	No	H	SSL/TLS	Basic/JWT	C
EMQX	MQTT 5.x	EMQ	GA	AL 2.0	Yes	Yes	Yes	No	No	H+	SSL/TLS	HTTP/JWT/ LDAP	Java, .NET, Python & more
ZEROMQ		iMatix	GA	LGPLv3	Yes	Yes	Yes	No	No	M	SSL/TLS	SASL/ZAP	Java, .NET, Python & more
KAFKA		Apache Foundation	GA	AL 2.0	No	Yes	Yes	No	Yes	H+	SSL/TLS	SASL/mTLS/ Basic	Java, .NET, Python & more
STORM		Apache Foundation	GA	AL 2.0	No	Yes	Yes	No	Yes	M	SSL/TLS	SASL	Java, .NET, Python & more
PULSAR		Apache Foundation	GA	AL 2.0	Yes	Yes	Yes	Yes	Yes	H+	SSL/TLS	Basic/JWT/ OAuth 2.0/ Kerberos/ Athenz	Java, .NET, Python & more

Appendix B

Pulse example

```
prefix ex:      <http://example.com/id/>
prefix def:     <http://example.com/def/>
prefix bdi:     <https://ontology.bdinetwork.org/model/def/>
graph <ex:sampleData>{
  ex:message5 a def:Pulse .
  ex:message5 def:timestamp '18-02-1900 16:33:00'^^xsd:dateTime ;
        def:apiEndpoint <https://api.somewhere.nl/v1/> ;
        def:sparqlEndpoint <https://sparql.somewhere.nl/sparql/> ;
        def:targetProperties
          [
            a def:KeyValuePair ;
            def:key bdi:type;
            def:value bdi:Vrachtwagen ;
          ] ,
          [
            a def:KeyValuePair ;
            def:key bdi:id;
            def:value <https://picobellobv.nl/truck1>;
          ] .
}
```

Topsector Logistiek
Ezelsveldlaan 59
2611 RV Delft
+31 15 251 65 65
www.topsectorlogistiek.nl

Topsector
Logistiek