

Gaia-X Federation Services

**Technical Implementation of GXFS Open-Source
Software and QA Requirements**

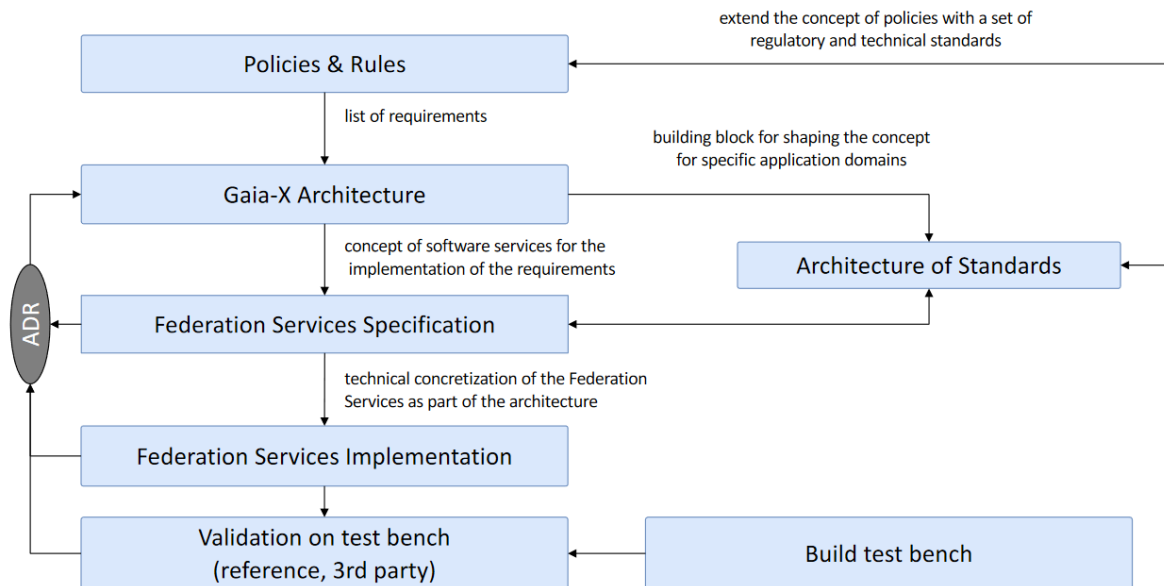
Structure

Abstract	3
Open-Source Software	4
Runtime environments.....	4
Languages	4
Documentation.....	4
QA requirements	5
Target hosting environments for QA.....	5
DevOps tools	5
Architecture Decision Records	6
ADR-001: JSON-LD as the Exchange Format for Self-Descriptions.....	6
Summary	6
Context	7
Alternative Technologies.....	7
Decision Statements.....	7
Consequences.....	7
ADR References	7
External References.....	7
ADR-002: REST as the Interface Technology for Federation Services	8
Summary.....	8
Discussion	8
Decision Statements.....	9
Consequences.....	9
Notes	9
ADR References	9
External References.....	9
ADR-003: Lifecycle of Self-Descriptions	10
Summary.....	10
Context	10
Decision Statements.....	10
Consequences.....	11
ADR References	11
External References.....	11

Abstract

The Federation Services are necessary to enable a Federation of infrastructure and data, provided with open-source reference implementation. This will open up technology where applicable, while existing certifications and standards for accreditation will be recognized.

The following figure shows the relation to Gaia-X documents:



Please refer to the Gaia-X Architecture Document:

https://gaia-x.eu/pdf/Gaia-X_Architecture_Document_2103.pdf

and the Policy Rule Document:

https://gaia-x.eu/pdf/Gaia-X_Policy%20Rules_Document_2104.pdf

Open-Source Software

The Gaia-X Federation Services are Open-Source Software and Open Specifications. For the GXFS project the following rules are mandatory.

- Compliance with Apache License, Version 2.0
 - Text version: <https://www.apache.org/licenses/LICENSE-2.0.txt>
 - SPDX short identifier: [Apache-2.0](#)
 - OSI Approved License: <https://opensource.org/licenses/Apache-2.0>
- The project should consume third-party content under one of the approved licenses listed here: <https://www.eclipse.org/legal/licenses.php#approved>. Exceptions must be documented.
- A project git repository will be made available, and all contractors are obliged to use this repository as master on a daily base. A parallel contractor repository is not allowed.
- All work must be done in compliance with Eclipse Contributor Agreement 3.1.0 or higher: <https://www.eclipse.org/legal/ECA.php>

Runtime environments

- Isolation: Dockers (Alternatives cri-o, podman)
- Container Orchestration: Kubernetes
- Eventing: CloudEvents.io
- Mesh Services: ISTIO.io

Languages

- Preferred: shell script, python, java
- With prior approval: nodejs, rust, C++, go [nodejs is a runtime and not a language, TypeScript or JavaScript are the languages of nodejs]

Documentation

- Generator
 1. readthedoc
 2. mkdocs
- Product Documentation
 1. AsciiDoc
 2. RTS
- Architecture Modelling
 1. Archimate
 2. UML

QA requirements

- helm charts for Kubernetes (<https://helm.sh/>) - In addition we consider specific Operators (CNCf Operator) for runtime administration
- docker-compose for docker (https://hub.docker.com/_/composer)
- Ansible to configure hosts (<https://www.ansible.com/>)
- Terraform to spawn infrastructure (<https://www.terraform.io/>)

Target hosting environments for QA

- Infrastructure/IaaS: <https://opennebula.io/> or <https://www.openstack.org/> or similar
- Object storage: <https://www.zenko.io/> or similar (Ceph with Rook supports Block)

DevOps tools

- Git
- GitFlow
- Jira

Architecture Decision Records

Architecture Decision Records (ADR) document important decisions. This ensures the synchronization between Work Packages. This is especially important as new members join Gaia-X. Accepted ADR are appended to the Architecture Document. Future releases of the Architecture Document state up to which ADR changes have been incorporated. ADR are written in the RestructuredText format. This is both human-readable, fit for source versioning and can be translated into most other document format. The Primer on Restructured Text explains the formatting rules. See the file 000_adr_template.rst for an example.

ADR-001: JSON-LD as the Exchange Format for Self-Descriptions

adr-id: 001

revnumber: 1.0

revdate: 06-07-2020

Status: accepted

Author: Self-Description WP, Catalogue WP

stakeholder: Self-Description WP, Catalogue WP

Summary

GAIA-X needs to define a serialization format for the exchange of Self-Descriptions. The exchange format allows the self-descriptions to be serialized into files for transportation between Services and Catalogues. JSON-LD is selected as the Self-Description serialization format.

The following is a minimal example for the JSON-LD format:

```
{
  "@context": "https://json-ld.org/contexts/gaia-x.jsonld",
  "@id": "http://dbpedia.org/resource/MyService",
  "provider": "http://dbpedia.org/resource/MyProvider",
  "name": "MyService"
}
```

Linked Data Proofs 1.0 (<https://w3c-ccg.github.io/ld-proofs/>, currently in draft status).

An example for a signed document with Linked Data Proofs is this:

```
{
  "@context": "https://www.w3.org/2018/credentials/examples/v1",
  "title": "Hello World!",
  "proof": {
    "type": "Ed25519Signature2018",
    "proofPurpose": "assertionMethod",
    "created": "2019-08-23T20:21:34Z",
    "verificationMethod": "did:example:123456#key1",
    "domain": "example.org",
    "jws": "eyJ0eXAiOiJK...gFWFOEjXk"
  }
}
```

Context

The Architecture Document (June 2020) states that Self-Descriptions are expressed in an extensible format.

JSON is an established data serialization format. It is both human-readable and machine-interpretable.

JSON-LD combines JSON with semantic technologies (ontologies) from the Linked Data community. Specifically, the RDF-Standard is referenced. So the schema-definitions (and tooling) from the established RDF format can be reused.

Alternative Technologies

Decision Statements

The serialization format for the exchange of GAIA-X Self-Descriptions is JSON-LD 1.1.

The RDF 1.1-standard is used to express an extensible hierarchy of schemas for Self-Descriptions with well-known attributes.

Cryptographic signatures are added to Self-Descriptions according to the Linked Data Proofs 1.0 specification.

Consequences

All Self-Descriptions need to be ready for serialization into the JSON-LD format.

The extensible hierarchy of self-description attributes is expressed in an ontology according to the RDF standard.

The Identifier of GAIA-X Assets and Participants must be IRIs (Internationalized Resource Identifiers [RFC3987]) so that they can be used for cross-referencing between Self-Descriptions in the JSON-LD format.

ADR References

- None

External References

- [JSON-LD] JSON-LD 1.1 - A JSON-based Serialization for Linked Data, <https://www.w3.org/TR/json-ld11>
- [LDP] Linked Data Proofs 1.0, <https://w3c-ccg.github.io/ld-proofs/>
- [RDF] RDF 1.1 Concepts and Abstract Syntax, <https://www.w3.org/TR/rdf11-concepts/>

ADR-002: REST as the Interface Technology for Federation Services

adr-id: 002

revnumber: 0.9

revdate: 23-07-2020

Status: accepted

Author: Catalogue WP

stakeholder: All Federation Services

Summary

GAIA-X defines a set of Federation Services. These Services provide an API for the development of client applications. This "client" could be an application developed by a GAIA-X Participant or a user interface that is part of GAIA-X itself.

For the internal consistency of GAIA-X and the Federation Services, a common technology and design principles should be selected for the Federation Service's API interface.

The ADR proposes REST (HTTP+JSON) as the interface technology and OpenAPI as the interface definition language.

Discussion

Many different protocols are in use for internet-based software interfaces. Following is an (incomplete) overview listing only the most common technologies.

- SOAP Webservices (<https://www.w3.org/TR/soap/>)
- XML-RPC (<http://xmlrpc.com/>)
- REST (HTTP+JSON)
- Object-Oriented Interfaces (Corba, OPC UA, ...)
- Message-Oriented Interfaces (e.g. via Kafka, MQTT, AMQP, DDS, ...)

In principle, the Federation Services could be implemented using either of the mentioned interface technologies. But there are further criteria besides the core functionality to take into consideration. Selecting a widely used technology with a proven track-record ensures that GAIA-X Participants have developers with the skills to immediately make use of the Federation Services. Furthermore, the established technologies are more likely to be widely supported by the different programming languages and environments in the long-term.

From those criteria, the choice has fallen on a combination of REST (HTTP+JSON). It is the mostly widely used technology for web-API development (also used by the current hyperscalers) and has mature tooling.

Based on the choice of REST (HTTP+JSON), several competing formats exist for expressing the interface definitions in a human and machine-readable format.

- OpenAPI (<http://spec.openapis.org/oas/v3.0.3>)
- RAML (<https://raml.org/>)
- Hydra (<https://www.hydra-cg.com/>)
- Custom Format for GAIA-X

Again, all these technologies could perform the task to a sufficient degree. We select the most widely used alternative OpenAPI.

Decision Statements

The GAIA-X Federation Services use JSON-Documents transferred via a HTTP/REST API for their interfaces.

Special use cases (e.g. for streaming data) might receive an exemption to use different interface technologies.

The interfaces of the Federation Services are specified in the OpenAPI Specification (v3 or later). <http://spec.openapis.org/oas/v3.0.3>

Consequences

Developers can reuse tools for the commonly used combination of HTTP+JSON for the Federation Services.

Code stubs to interact with the Federation Services can be auto-generated from the OpenAPI definitions for many programming environments.

The Work-Packages that define the interfaces of the Federation Services use OpenAPI to define the interfaces in a human and machine-readable format.

Notes

The REST API specifications are described in https://gitlab.com/gaia-x/gaia-x-core/gaia-x-core-document-technical-concept-architecture/-/blob/master/architecture_document/federation_services.rst

ADR References

- ADR-001: JSON-LD as the Exchange Format for Self-Descriptions

External References

- [REST] https://en.wikipedia.org/wiki/Representational_state_transfer
- [OpenAPI] <http://spec.openapis.org/oas/v3.0.3>

ADR-003: Lifecycle of Self-Descriptions

adr-id: 003

revnumber: 1.0

revdate: 05-11-2020

Status: proposed

Author: Catalogue WP

stakeholder: Self-Description WP, Catalogue WP

Summary

There are four possible states for the lifecycle of GAIA-X Self-Descriptions: active, eol (end of life), deprecated and revoked.

Context

GAIA-X Assets (Nodes, Services, etc.) and their Self-Descriptions can be updated over time. Furthermore, there can be Self-Descriptions that are abandoned or even revoked for containing false information. This needs to be explicitly tracked to prevent the Catalogue and GAIA-X participants to work with outdated Self-Descriptions.

The claims of the Self-Descriptions in JSON-LD format can carry cryptographic proofs based on hash signatures. Furthermore, the hash of the JSON-LD file can be used as an identifier to reference to the source of information from the Catalogue. Hence the JSON-LD files are immutable and can only be replaced as a whole. The state of the Self-Descriptions therefore needs to be tracked as metadata outside of the JSON-LD files itself.

End of Life: The Self-Descriptions are the source information for the GAIA-X Catalogue. In order for the Catalogue to contain only up-to-date information it should be "self-cleaning" whereby outdated information is removed automatically. This can be achieved by timeout dates attached to every Self-Description after which they are end-of-life. It is recommended that the automatic timeout date of Self-Descriptions is set rather low, e.g. 90 days. This has proven useful in the context of TLS certificates [LetsEncrypt] where a frequent renewal forces providers that automated update systems are put in place instead of infrequent manual updates.

Deprecated: If two versions of GAIA-X Assets (Nodes, Services, Data etc.) are offered at the same time, then they each have independent Self-Descriptions. In order for the Self-Descriptions to be self-contained, there shall be no partial updates to Self-Descriptions. The entire JSON-LD file is published in an updated version and the old Self-Description is deprecated with reference to the updated Self-Description.

Revoked: GAIA-X needs to protect against bad actors that might not be able or willing to correct false information. Hence a revocation mechanism is put into place by which Self-Descriptions can be marked as non-active. Revocation can be performed by the original issuer of a Self-Description and also by trusted parties.

Decision Statements

The Self-Descriptions in JSON-LD format (ADR-001) have an additional state information that can change over time. The possible states are:

- active

- eol (end of life after a timeout date)
- deprecated (by a newer Self-Description)
- revoked (by a trusted party)

The default state is "active". The other states are terminal, so no further state transitions are made once a Self-Description has become non-active.

Non-active Self-Descriptions might still be available in the JSON-LD format in the historical record. But they are no longer considered for new search queries in the Catalogue, etc.

Self-Descriptions have a timeout date after which they are in the "eol" state. The timeout date is part of the JSON-LD file and considered in cryptographic signatures.

Self-Descriptions in the JSON-LD format are immutable and cannot be modified after they have been published. They can only be replaced by a new Self-Description (deprecated) or given another non-active status.

A Self-Description with wrong or even fraudulent information can be revoked by the original issuer or a trusted party. Who is allowed to revoke is at the digression of the operators of the different Catalogues. Future rules for Catalogue operators by the GAIA-X organization may apply.

Consequences

Having decided on the possible states reveals new questions that need to be answered in the context of the GAIA-X Catalogue.

The specification of the Catalogue has to describe in detail how the information of the Self-Description state is exposed in its API and search query interfaces.

The state information needs to be distributed between Catalogue instances. There might be disagreements / information gaps between Catalogue instances that need to be resolved.

ADR References

- ADR-001: JSON-LD as the Exchange Format for Self-Descriptions

External References

- [LetsEncrypt] <https://letsencrypt.org>