Software Requirements Specification

for

Gaia-X Federation Services

Trust Management Infrastructure IDM.TRAIN

Published by

eco – Association of the Internet Industry (eco – Verband der Internetwirtschaft e.V.)
Lichtstrasse 43h
50825 Cologne, Germany

Table of Contents

List of Figures

List of Tables

1 Introduction

To get general information regarding Gaia-X and the Gaia-X Federation Services please refer to [TAD], [TF] and [PRD].

1.1 Document Purpose

The purpose of the document is to specify the requirements of the Identity Management and Trust Subcomponent “Trust Management Infrastructure for Gaia-X (TRAIN)” with the intention of a European wide public tender for implementing this software. The main audience for this document is attendees of the public tender, which are able to supply an open-source software solution for the area of identity and document verification with the purpose to provide digital support for existing certification bodies within Gaia-X.

1.2 Product Scope

The purpose of this product is to provide components for establishing and verifying the root of trust for participants in the distributed Gaia-X ecosystem and credentials issued by these entities. This is achieved through the introduction of trust lists combined with anchoring of pointers in the DNS. The conceptual framework for this is described in [GX.TRUST].

Gaia-X Federations and other entities are supported in the sovereign publication and administration of trust lists for their specific trust frameworks. Verifying entities are supported in their sovereign trust decisions. To achieve this, the following functionalities MUST be developed:

- Trust Framework Configuration
- Trust List Management
- Zone Manager Handler
- Trusted Content Resolver (Extended Universal Resolver) + Libraries
- DNS Zone Manager

Please note that the libraries are intentional for different languages such as GO, Java, Python and Javascript. It’s also intentional to create the libraries as helpers for using the extended universal resolver, by adding content resolver steps, validation routines for VC and other assistive functionalities.

⚠ If it’s required to do code restructurings, modifying existing solutions by adding new microservices etc. (e.g., the universal resolver) then this is explicitly required.

Please note, that it is explicitly required to deliver the software up and running. Responsibility for existing code cannot be shifted to previous development teams.

1.3 Definitions, Acronyms and Abbreviations

The IDM and Trust Architecture Overview Document [IDM.AO] MUST be considered and applied as the core technical concept that includes also the Terminology and Glossary.

1.4 References

[IDM.AO]	**Gaia-X WP1[1] (2021), Architecture Overview**
	Please refer to annex "GX_IDM_AO"
[GX.TRUST]	**Trust Management Infrastructure for Gaia-X, Concept Document**
	Please refer to annex "Trust Management Infrastructure for GX"
[LIGHTEST.TSPA]	**LIGHTest TSPA Source Code**
	https://github.com/H2020LIGHTest/TrustSchemePublicationAuthority
[LIGHTEST.ZM]	**LIGHTest Zone Manager Source Code**
	https://github.com/H2020LIGHTest/ZoneManager
[PRD]	**Gaia-X Policy Rules and Labelling Document (2022)**
	https://docs.gaia-x.eu/policy-rules-committee/trust-framework/latest/
[RFC2119]	**Network Working Group (1997) Key words for use in RFCs to Indicate Requirement Levels**
	https://tools.ietf.org/html/rfc2119
[TAD]	**Gaia-X Architecture Document (2022)**
	https://docs.gaia-x.eu/technical-committee/architecture-document/latest/
[TDR]	**Gaia-X Federation Services Technical Development Requirements**
	Please refer to annex "GXFS_Technical_Development_Requirements"
[TF]	**Gaia-X Trust Framework (2022)**
	https://docs.gaia-x.eu/policy-rules-committee/trust-framework/latest/
[TRAIN]	**TRAIN (Trust Management Infrastructure) - Website**
	https://train.trust-scheme.de/info/

[1] Please refer to appendix B for an overview and explanation of the Work Packages (WP).

[TRAIN.ESSIF]	**ESSIF Gitlab for TRAIN with Sourcecode**
	https://gitlab.grnet.gr/essif-lab/infrastructure/fraunhofer
[VC.DataModel]	**W3C (2022), Verifiable Credentials Data Model v1.1**
	https://www.w3.org/TR/vc-data-model

1.5 Document Overview

The document describes the product perspective, functions, and constraints. It furthermore lists the functional and non-functional requirements and defines the system features in detail. The listed requirements are binding. Requirements as an expression of normative specifications are identified by a unique ID in square brackets (e.g. **[IDM.ID.EX.Number]**) and the keywords MUST, MUST NOT, SHOULD, SHOULD NOT, MAY, corresponding to RFC 2119 [RFC 2119], are written in capital letters (see also [IDM.AO] - Methodology).

2 Product Overview

2.1 Product Perspective

TRAIN provides a trust management infrastructure for Gaia-X Federation Services (GXFS). The TSA can use the Trusted Component Resolver Libraries to verify the institutional trust of verifiable credentials. The Organizational Credential Manager (OCM) can use the Trusted Component Resolver Libraries to validate the trust of the organizational verifiable credentials before storing them in the wallet. The Notary (via the Notarization API) uses the TSPA Connector to enroll trusted entities into the trust framework and add them via the TSPA/Federator to the Trust List.

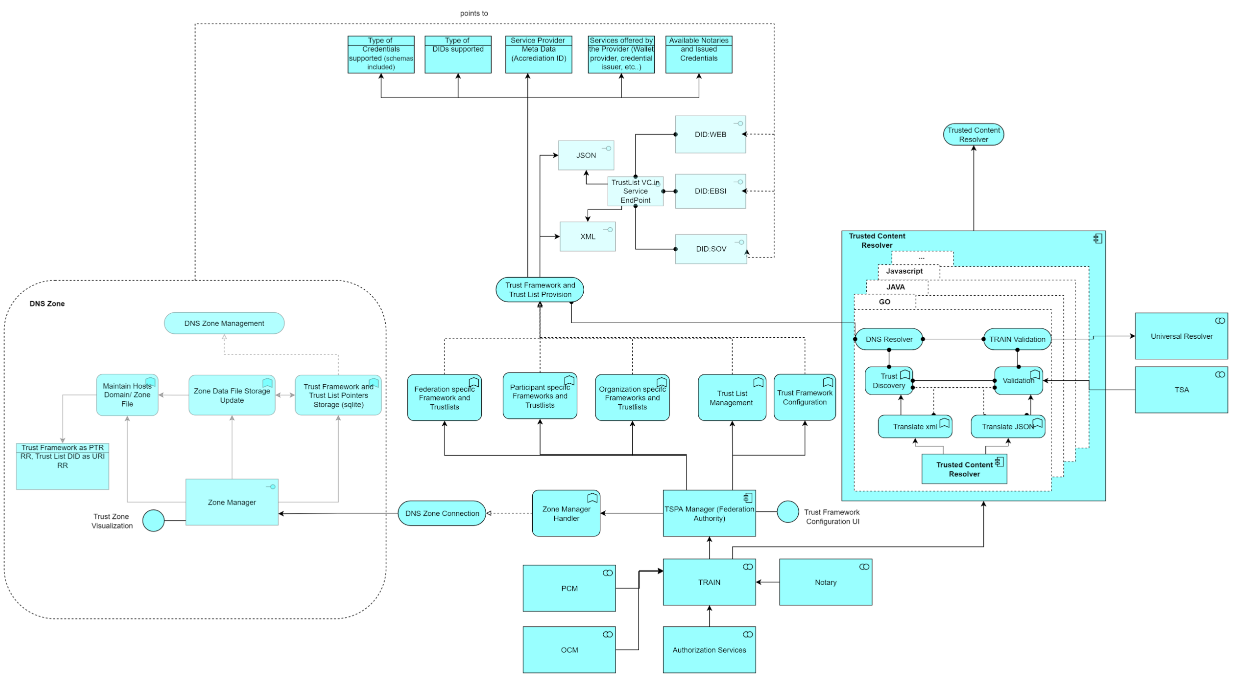

points to
Type of Credentials supported (schemas included)
Type of DIDs supported
Service Provider Meta Data (Accrediation ID)
Services offered by the Provider (Wallet provider, credential issuer, etc..)
Available Notaries and Issued Credentials
DID:WEB
JSON
TrustList VC in Service EndPoint
DID:EBSI
XML
DID:SOV
Trusted Content Resolver
Trust Framework and Trust List Provision
DNS Zone
DNS Zone Management
Maintain Hosts Domain/ Zone File
Zone Data File Storage Update
Trust Framework and Trust List Pointers Storage (sqlite)
Trust Framework as PTR RR, Trust List DID as URI RR
Zone Manager
Trust Zone Visualization
DNS Zone Connection
Zone Manager Handler
Federation specific Framework and Trustlists
Participant specific Frameworks and Trustlists
Organization specific Frameworks and Trustlists
Trust List Management
Trust Framework Configuration
TSPA Manager (Federation Authority)
Trust Framework Configuration UI
Trusted Content Resolver
...
Javascript
JAVA
GO
DNS Resolver
TRAIN Validation
Trust Discovery
Validation
Translate xml
Translate JSON
Trusted Content Resolver
Universal Resolver
TSA
PCM
TRAIN
Notary
OCM
Authorization Services

Figure 1 TRAIN: Overview

A bigger overview of Figure 1 can be found at the end of this document in annex A.

2.2 Product Functions

The core functions of IDM.TRAIN are:

- Provision of a Trust Framework and Trust lists (TSPA Manager is responsible for this functionality):
  - Allows for configuration of a Trust Framework
  - Allows for Trust List Management
  - Provides federation/organization/participant specific Trust Lists in different formats

- Anchoring a Trust Framework and Trust List into the DNS (Zone Manager is responsible for this functionality):
  - Allows for global discovery based on an established and trusted infrastructure
  - Trust Frameworks are anchored in DNS Pointer Resource Record (PTR RR)
  - Trust List URI DID is anchored in DNS URI Resource Record (URI RR)
  - DNSSEC allows for chain of trust

- Enrollment of trusted entities into the Trust Framework (TSPA connector is responsible for this functionality):
  - Notary (via the Notarization API) uses the TSPA connector to enroll trusted entities to the Trust Lists

- Verifying the Institutional Trust of Verifiable Credentials:
  - Trusted Content Resolver is responsible for this functionality
  - Allows Global Discovery of Trust Frameworks through DNS Resolver
  - The content from terms of Use of the verifiable credential will be used for trust discovery
  - Verification of issuer details of the credential with the information of the trust list
  - Integrity of VC must also be verified
  - Integrity of the chain of trust of DNSSEC must be validated.

2.3 Product Constraints

⏩ **[IDM.TRAIN.00000] The document IDM.AO is the common basis for this functional specification**

The architecture document [IDM.AO] is an essential part of this specification and a prerequisite for understanding the context. The specifications and requirements from the Architecture Document [TAD] MUST be taken into account during implementation. ⏪

⚠ *Please note, that part of the functional requirements require Business Analysts to analyze the requirements of Trust Frameworks and different Trust List formats. Inputs from Business Analysts will be required for the development team set up.*

⏩ **[IDM.TRAIN.00001] The document GX.TRUST is the basis for the TRAIN Trust concept**

The Trust Management Infrastructure for Gaia-X document [GX.TRUST] is an essential part of this specification and a prerequisite for understanding the concept for the Gaia-X Trust Infrastructure. It MUST be taken into account during implementation. ⏪

⏩ **[IDM.TRAIN.00002] Implementation Requirement for TSPA and Zone Manager**

The Code Base of TSPA [LIGHTEST.TSPA] and Zone Manager [LIGHTEST.ZM] MUST be taken as reference and MUST be extended based on the functionalities. For example: Trust Framework Initialization and enrollment in DNS follows the format of **_scheme._trust.federation.company1.de** for both PTR and URI records. ⏪

2.4 User Classes and Characteristics

User Class	Description	Frequency	Expertise	Privilege Level	Product Usage
Administrator	Sets up Zone Manager, Trust Framework Configuration, Trust List Initialization	Low	High	High	Maintenance
Notary	Uses TSPA Connector to manage the Trust List	High	High	High	Administration of enrollment of trusted entities to the Trust List
TSA / Developers	Uses Trusted Content Resolver to verify the institutional trust of the verifiable credential	High	High	Low	Verification of inclusion of issuer details in the Trust Framework
PCM	Uses the trust list to verify the trust of the issuer/verifier	High	High	Low	Before VC verification or connection establishment
AAS	Uses the trust list to decide if	High	High	Low	Cyclic/During Startup

	a public key must be part of the domain				
OCM(s)	Uses the trust lists to decide if a connection is trustworthy	High	High	Low	On Connection establishment

Table 1 User Classes and Characteristics

2.5 Operating Environment

[IDM.TRAIN.00003] Kubernetes Environment

The product MUST be operable on standard Kubernetes based environments without any hardware restrictions. The reference environment for demonstration and development purposes MUST be on a SCS cluster (Sovereign Cloud Stack) provided by the contractor. In addition, the developed components must also run in a restricted developer local environment.

2.6 User Documentation

[IDM.TRAIN.00004] Administration Documentation

The documentation MUST contain:

- Installation Manuals for Zone Manager, TSPA Manager
- Trust Framework Configuration and Trust List Management
- Description of Deployment/Compile Process
- Description of the Automatic Tests / Verification
- How to build the products from source code

[IDM.TRAIN.00005] User Documentation

The documentation MUST contain:

- Short Software Description (why and for what, when to use, how use, where to use)
- Usage guide (Trusted Content Resolver)
- Usage guide for installation of libraries with GO, Javascript, Java and Python
- Usage guide to integrate with Notarization Service (NOT)
- Usage and integration guide for TSA, OCM, PCM
- GDPR design decisions
- Security concept
- Operations concept for all components
- Blueprint guides how to setup a federation in usage of all components (step by step)
- FAQ
- Keyword Directory

2.7 Assumptions and Dependencies

An understanding of the overall Gaia-X architecture and philosophy and the Trust Management Infrastructure Concept is necessary. Please refer to [IDM.AO] and [GX.TRUST].

The control over minimum of 2 fully qualified domains or subdomains is strictly required for the product in combination with an authoritative DNSSec Server.

3 Requirements

3.1 External Interfaces

3.1.1 User Interfaces

[IDM.TRAIN.00006] Trust Framework Configuration

The product MUST provide a Web UI which allows the administrator to add new trust frameworks and corresponding DIDs.

[IDM.TRAIN.00007] Trust Zone Visualization

The product MUST provide a Web UI which visualizes the content of a trustzone (e.g., zones, pointers, DIDs etc.) for external/public users.

3.1.2 Software Interfaces

[IDM.TRAIN.00008] Universal DID Resolver[2]

The DIF Universal Resolver is the product basis that is used for resolving trusted content of DIDs. For building the trusted content resolver, this software is the basis that is to be enhanced.

[IDM.TRAIN.00009] DNS Resolver

An Open-Source DNS Resolver MUST be used to discover PTR, URI, DNSSEC Resource Records (RR). And this MUST be integrated with Trusted Content Resolver.

[IDM.TRAIN.00010] Authoritative DNSSec Name Servers

DNS Name Servers such as NSD[3] and KNOT DNS[4] MUST be used and tested for DNS Zone installation and configuration.

[2] https://github.com/decentralized-identity/universal-resolver
[3] https://www.nlnetlabs.nl/projects/nsd/about/
[4] https://www.knot-dns.cz/

⏩ **[IDM.TRAIN.00011] Cryptographic Libraries**

Open-Source Cryptographic libraries compliant with the BSI[5] and EU DSS[6] rules including BSI TR-02102-1[7] MUST be used for VC/trustlist signing, VC / trustlist proof validation and DNSSEC Trust Chain Validation. ⏪

3.1.3 Communications Interfaces

⏩ **[IDM.TRAIN.00012] Eventing**

If the use of events within the software architecture is required, it is mandatory to use software abstraction according to CloudEvents specification[8] for publishing and subscription. The minimal supported protocol binding MUST be HTTP[9] and JSON[10] Protocol Binding. ⏪

⏩ **[IDM.TRAIN.00013] Eventing Infrastructure**

The event broker for the eventing MUST abstract the storage and delivery infrastructure. In Kubernetes environments, the broker MUST be uniform across all Lots. (e.g., KNative[11]) ⏪

3.2 Functional

⏩ **[IDM.TRAIN.00014] Trust Framework Configuration**

Description
This functionality MUST allow the creation of trust frameworks, the creation and configuration of DIDs with well-known did configurations[12], instantiation of trust lists, the envelopment of trust lists in Verifiable Credentials with proof and configuring the enveloped VCs in the service end point of DID Documents. The Sequence Diagram (Figure 2, System Features Section) elaborates the process. Each and every process MUST have separate API endpoints to instantiation.

Constraints
The following constraints MUST be fulfilled:

- Allow creation and mapping of multiple trust frameworks (multiple federations/domains)
- Allow referencing of trust frameworks from other domains

[5] https://www.bsi.bund.de/EN/Themen/Unternehmen-und-Organisationen/Standards-und-Zertifizierung/Kryptografische-Vorgaben/kryptografische-vorgaben_node.html
[6] https://ec.europa.eu/digital-building-blocks/wikis/display/DIGITAL/Digital+Signature+Service+-++DSS
[7] https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102.html
[8] https://cloudevents.io/
[9] https://github.com/cloudevents/spec/blob/v1.0.2/cloudevents/bindings/http-protocol-binding.md
[10] https://github.com/cloudevents/spec/blob/v1.0.2/cloudevents/formats/json-format.md
[11] https://knative.dev/docs/

[12] https://identity.foundation/.well-known/resources/did-configuration/

- Configure DID as URI record with corresponding trust frameworks
- Verify the DID with well-known DID configuration before enrolling in URI record
- Creation of digital signed Trust List with different formats - json and xml
- Allow creation of VC with trust list end point as credential subject
- Allow signature of VC- and storage of VC Trust Lists
- Each process MUST have separate API endpoints

Interfaces

- Zone Manager
- Trust List Storage
- VC Storage
- Zone Data Storage
- Zone Manager Connector

Input

1. Trust Framework enrollment
2. DID enrollment
3. Envelope Trust List Endpoint
4. Instantiating Trust List (Storage at Web Server or IPFS)

Output

1. Creation of Trust Framework as PTR record in DNS
2. DID enrolled as URI RR mapped with corresponding Trust Framework
3. Verifiable Credential with Trust List endpoint as Credential Subject
4. Trust List published in storage with retrievable API endpoint

Acceptance Criteria

The following acceptance criteria MUST be met:

1. A request update of trust frameworks and DID configuration is successfully reflected in the DNS Zone File (200)
2. An instantiation of a trust list is reflected in the trust list storage with possibility to retrieve via API endpoints
3. Creation of a VC is allowed with ability to sign the credential
4. A wrong context or missing data leads to an exception (400)
5. An audit entry is created
6. An error is provided if a record is in progress by the operator
7. Should be able to reference Trust Frameworks from other Domains ◄◄

►► **[IDM.TRAIN.00015] Trust List Management**

Description

This functionality MUST allow CRUD (create, read, update, delete) operations on the trust list at the Trusted Data Store. It must also allow for identifying different types of requests based on the enrollment inputs, for example: federation, organization, participant, service provider. If the organization or federation enrolls with its own trust framework, the trust framework name MUST be enrolled in the DNS as PTR record using the Zone Manager Handler.

Constraints

The following constraints MUST be fulfilled:

- Support JSON and XML trust-list update
- Have separate endpoints for create, read, update, delete functionalities
- Support multiple forms of storage on IPFS and HTTPS WebServer
- Identify and enroll the trust framework into DNS Zone file as PTR RR

Interfaces

- Database of Trusted Content Storage
- Zone Manager Handler
- Zone Manager
- Notarization API

Input

A confirmed request record. This MUST include:

1. DID of the organization
2. URI of the trust list
3. URI of the schema
4. Metadata of the entity (to be specified further according to business analyst analysis)
    - Legal Name
    - Certification details
    - Assurance Levels
5. Configure supported DID methods
6. Type of other digital credentials supported (example: x509)
7. Different services offered by the organization

Output

1. In create operation: a new trust list entry MUST be created
2. In read operation: the API endpoints MUST have the flexibility to read the whole trust list and also separate entities using UUID
3. In update operation: the change requested by the user MUST be reflected in trust list
4. In delete operation: the trust list entry of the entity MUST be deleted from the list
5. Must contain auditing mechanisms for the trust list. Please refer to the trust concept document [GX.TRUST] for different mechanisms listed there.
6. Must be able to integrate with the Notarization API

Acceptance Criteria

The following acceptance criteria MUST be met:

1. A request update has been successfully reflected in the trust list (200)
2. A wrong context or missing data leads to an exception (400)
3. Audit entry created

4. Error, if record is in progress by the operator
5. Integrate with the Notarization API

[IDM.TRAIN.00016] Trusted Content Resolver (Trust Discovery)

Description

This functionality MUST allow for the resolution of the trust list to find the issuer details in the trust list. The resolver MUST base on the DIF Universal Resolver[13] and MUST provide additional functionality to iterate recursive over DID Documents by resolving references in service endpoints during the standard resolving. The Resolver MUST integrate with the TSA via libraries. The resolving MUST be controllable by giving a list of endpoint types which are considered during the resolving by a defined range of actions. Responding content references MUST be collected and provided to the user either as list or as Callback during/after the resolving of standard documents. For instance, when a DID is resolved, the extended universal resolver collects additional DIDs from the service endpoint section (selected by types) and searches there again for other content. During this process, the defined content types are collected as reference. E.g., a list of URLs to type “gx-trusted-issuer” grouped by DID for a later processing by the libraries.

Constraints

The following constraints MUST be fulfilled:

- Resolve DNS PTR queries
- Resolve DID URI queries
- Navigate to corresponding service type on the DID Document to fetch the corresponding trust list based on user inputs
- Handle multiple Trust Framework Pointer as array
- Validate DNS Name against DNSSEC
- Provision MUST be made to allow user to configure their own DNS Resolvers
- Support different user defined trust list content types for corresponding trust list discovery

Interfaces

- DNS Resolver
- Universal DID Resolver[14]
- DNSSEC validator

Input

1. Trust Framework Pointer (e.g., example.federation1.de) + Types to be considered
2. Issuer details from the VC/VP (e.g., DID/URI)
3. ServiceType of the trust list (e.g., issuance service, verifier service)

Output

1. Corresponding DID mapped to Trust Framework Pointer

[13] https://github.com/decentralized-identity/universal-resolver
[14] https://github.com/decentralized-identity/universal-resolver

2. DID Document of the DID
3. Trust List VC endpoint

Acceptance Criteria

The following acceptance criteria MUST be met:

1. Use standardized DNS resolvers
2. Use standardized DID resolver
3. Navigate multiple trust framework pointers
4. Discover different trust list formats
5. Support different service types
6. Optimized search mechanisms (e.g., Merkle Tree)
7. MUST Allow configuration of different DNS Name Servers
8. MUST allow configuration of user defined service content type (e.g., gxfs-trusted-issuer)
9. The complete discovery process MUST be able to integrate with the TSA via libraries in GO, JAVA, Javascript, Python ◀◀

▶▶ **[IDM.TRAIN.00017] Validation**

Description

This functionality MUST validate the output of the trust discovery functionality of the Trusted Content Resolver. The validation functionality MUST validate the association of DID with a well-known DID configuration. And MUST also be able to validate the integrity of the VC. Then it MUST also be able to validate the issuer details from the trust lists extracted from service endpoints. The validation functionality MUST integrate with the TSA via libraries.

Constraints

The following constraints MUST be fulfilled:

- Validate the association of a DID with a well-known DID Configuration
- Verify the proof of a VC with a public key from a DID Document
- Display the metadata information, public keys, schema from the trust list
- Support multiple signature proofs

Interfaces

1. Trust Discovery

Input

The inputs for validation are based on the output of the trust discovery functionality

1. Corresponding DID mapped to Trust Framework Pointer
2. DID Document of the DID
3. Trust List VC endpoint

Output

1. Validation result of VC
2. Validation result of Issuer details (present/not present)

3. If Issuer details are found, display meta data information, public keys, schema from the trust list

Acceptance Criteria

The following acceptance criteria MUST be met:

1. VC validation mechanism supports multiple signature proofs
2. Standardized open-source libraries for validation in GO, JAVA, Javascript, Python
3. OCI compliant containerization (Dockerized)
4. Resolve different trust strategies
5. Resolve multiple lists
6. Implement the resolving strategies
7. The complete validation process of the Trusted Content Resolver MUST be able to integrate with the TSA via libraries in GO, JAVA, Javascript, Python ◂◂

▸▸ **[IDM.TRAIN.00018] Translate xml**

This functionality of the Trusted Content Resolver is responsible for detecting and translating the format of the trust list. If the format of the trust list is XML, this functionality is responsible for finding the right attributes and passing the information to the validation functionality. ◂◂

▸▸ **[IDM.TRAIN.00019] Translate JSON**

This functionality at Trusted Content Resolver is responsible for detecting and translating the format of the trust list. If the format of the trust list is JSON, this functionality is responsible for finding the right attributes and pass the information to the validation functionality. ◂◂

▸▸ **[IDM.TRAIN.00020] Zone Manager Handler**

Description

This functionality part of TSPA is responsible for configuring the TSPA with Zone Manger. It MUST allow publishing the Trust Framework and the DID in the DNS Zone file via the Zone Manager.

Constraints

The config file MUST be configured with the Domain Name of the Zone Manager and the password in order for the Zone Manager Handler to be able to communicate with the zone manager.

Interfaces

- Zone Manager
- Trust Framework and Trust List Pointer Storage
- Trust Framework Configuration

Acceptance Criteria

The following acceptance criteria MUST be met:

1. The TSPA MUST be able to configure the Zone Manager using the Zone Manager Handler.

[IDM.TRAIN.00021] Federation specific Framework and Trust Lists

Description
This functionality is responsible for providing the trust list data model for federation specific use cases.

Interfaces
- Trust Framework Configuration
- Trust List Management

Input
- No input

Output
The Trust List Data model MUST cover the following aspects in detail:
1. Business rules
2. Federation meta data (e.g., legal name, etc.)
3. Accommodate different identifiers (e.g., LEI - Legal Entity Identifier)
4. Accommodate assurance levels
5. Accommodate different digital identities (e.g., DID, PKI)
6. Different services offered by the federation
7. Accommodate auditable history information URI in trust list

Acceptance Criteria
The following acceptance criteria MUST be met:
1. Analysis with existing standards (e.g., ETSI, NIST, etc.)
2. Trust list data model in JSON and XML format with proper semantics
3. Address three use cases (i.e., Automotive, IIOT, Dataspaces)

[IDM.TRAIN.00022] Participant specific Frameworks and Trust Lists

Description
This functionality is responsible for providing the trust list data model for participant specific use cases.

Interfaces
- Trust Framework Configuration
- Trust List Management

Input
- No input

Output

The Trust List Data model MUST cover the following aspects in detail:

1. Business rules
2. Participant meta data (e.g., legal name, etc.)
3. Accommodate different identifiers (e.g.: LEI)
4. Accommodate assurance levels
5. Accommodate different digital identities (e.g., DID, PKI)
6. Different services trusted by the participant
7. Accommodate auditable history information URI in trust list

Acceptance Criteria

The following acceptance criteria MUST be met:

1. Analysis with existing standards (e.g., ETSI, NIST, etc.)
2. Trust list data model in json and xml format with proper semantics
3. Address three use cases for different participants ◂◂

▸▸ **[IDM.TRAIN.00023] Organization specific Frameworks and Trust Lists**

Description

This functionality is responsible for providing the trust list data models for organization specific use cases.

Interfaces

- Trust Framework Configuration
- Trust List Management

Input

- No input

Output

The Trust List Data model MUST cover the following aspects in detail.

1. Business rules
2. Organization meta data (e.g., legal name, etc.)
3. Accommodate different identifiers (e.g., LEI)
4. Accommodate assurance levels
5. Accommodate different digital identities (e.g., DID, PKI)
6. Different services offered by the organization
7. Accommodate auditable history information URI in trust list

Acceptance Criteria

The following acceptance criteria MUST be met:

1. Analysis with existing standards (e.g.., ETSI, NIST)
2. Trust list data model in json and xml format with proper semantics
3. Address three use cases for different organizations ◂◂

[IDM.TRAIN.00024] Zone Manager

Description
This functionality MUST allow publishing the Trust Framework and the DID in the DNS Zone file. It integrates with the TSPA Manager using the Zone Manager Handler.

Constraints
- Zone Manager MUST be extended on the code base [LIGHTEST.ZM]
- Trust Frameworks MUST be published as PTR records
- DIDs corresponding to Trust Frameworks MUST be published as URI records
- Zone Manager MUST provide DNSSEC configurations
- Zone File MUST be re-signed on every new update
- MUST allow trust framework to point to multiple other trust frameworks

Interfaces
- Zone Manager Handler
- Trust Framework and Trust List Pointers Storage (sqlite)
- TSPA Manager
- DNS Servers (NSD & KNOT)

Input
1. Trust Framework
2. DID Enrollment corresponding to Trust Framework

Output
1. Update of Trust Framework and DID in the DNS Zone file

Acceptance Criteria
The following acceptance criteria MUST be met:
1. A request update has been successfully reflected in the sqlite storage and Zone file (200)
2. A wrong context or missing data leads to an exception (400)
3. An audit entry is created
4. An error, if record is in progress by the operator
5. MUST integrate with the TSPA Manager using the Zone Manager Handler
6. MUST be tested with NSD & KNOT DNS Servers

3.3 Nonfunctional Requirements

3.3.1 HTTP Requirements

[IDM.TRAIN.00025] HTTPS

All HTTP Endpoints MUST be protected by TLS 1.2 (all protocol version numbers SHOULD be superseded by upcoming standards). Each endpoint of the product MUST support TLS certificates which are configurable by the administrator of the system.

[IDM.TRAIN.00026] HTTP Protocol Definitions

All HTTP Endpoints MUST follow [RFC7231] and [RFC5789], but it MAY be chosen which of the protocols are necessary to realize the functionality. For problem reports the [RFC7807] MUST be used in combination with Standard HTTP Error Codes.

3.3.2 Logging Requirements

[IDM.TRAIN.00027] Data Minimization

The data minimization principle is expressed in Article 5(1)(c) of the GDPR and Article 4(1)(c) of Regulation (EU) 2018/1725, which provide that personal data must be "adequate, relevant and limited to what is necessary in relation to the purposes for which they are processed". The data shall be stored for a period of time in accordance with national requirements and, as a minimum, shall consist of the following elements:

(a) node's identification

(b) message identification

(c) message data and time

All logged data/information MUST be documented in the GDPR design decisions for a GDPR review.

[IDM.TRAIN.00028] Logging Frameworks

The product MUST support logging frameworks e.g., graylog, fluentD or logstash to support logging and analysis by enterprise infrastructures. The supported framework MAY be chosen for the first version, but it MUST support the common open-source logging solutions with OpenTelemetry[15] support. The final solution MUST be aligned with the other subcomponents. It MUST be sketched in the operations concept how the support of multiple solutions is given in the future.

3.3.3 Performance Requirements

[IDM.TRAIN.00029] Up/Down Scale

All components MUST be able to scale out/down their functionality for an undefined amount of instances. This requires a parallel execution possibility which will be tested later on by performance tests.

[15] https://opentelemetry.io/

⏩ **[IDM.TRAIN.00030] Docker and Kubernetes Deployment**

Trust Framework and Trust List Provision service MUST provide a OCI containerized (dockerized) version for example DID with a DID Document that can be published locally and also a sample trust list enveloped in a VC that can be run on a local server for testing purposes.

A OCI containerized version of the Trust Framework and Trust List Provision service MUST be able to be integrated with DNS Zone Manager.

An implementation of the DNS Zone Manager MUST be tested with core DNS Kubernetes and also the service MUST be OCI containerized and able to resolve queries from the Trusted Content Resolver. ⏪

⏩ **[IDM.TRAIN.00031] Integration Functionality**

The DNS Zone Manager MUST be able to integrate Trust Framework and Trust List Provision Services. ⏪

⏩ **[IDM.TRAIN.00032] Resolving Functionality**

The Trusted Content Resolver MUST be able to resolve DNS Queries and MUST be able to fetch and verify trust framework and trust list provision services. ⏪

3.3.4 Safety Requirements

⏩ **[IDM.TRAIN.00033] Major Releases**

All used software components MUST use the major releases with Long Term Support (LTS). If no LTS is available, all components MUST use the latest major releases with security hardening. ⏪

3.3.5 Security Requirements

⏩ **[IDM.TRAIN.00034] CVE Patches**

All software components MUST have applied CVE patches, which are available for major releases. ⏪

⏩ **[IDM.TRAIN.00035] Risk-Management and Pentesting Requirements**

An adequate security risk management process is applied compliant to EU ENISA Risk Management in usage of a Tread Modelling Process like PASTA[16] or OWSAP[17]

[16] https://threat-modeling.com/pasta-threat-modeling/

[17] https://owasp.org/www-community/Threat_Modeling_Process

All software components MUST be compliant to the requirements of the Pentesting like BSI IS-Penetrationstest[18]

3.3.6 Software Quality Attributes

[IDM.TRAIN.00036] Software Quality Requirements

All software components MUST be compliant to the requirements within the quality assurance repository[19]. This includes testing on different layers (unit, component, integration), branch model that support stages, patch management, and code quality verification with adequate reporting. In addition, all requirements and quality attributes MUST demonstrated by automated behavior driven testing (BDD) methodology.

3.3.7 Business Rules

[IDM.TRAIN.00037] Software Consistency

The used technologies MUST have consistency. Standard technologies e.g., databases MUST be abstracted over JDBC, authentication over OIDC etc.

[IDM.TRAIN.00038] Cherry Picking

All components and the entire software architecture MUST be checked for the necessity for deployment of each single feature, to allow enterprise deployment customization.

3.4 Compliance

[IDM.TRAIN.00039] GDPR Audit Logging

The seven EU GDPR principles MUST be considered: Data handling and logging MUST fulfill lawfulness, fairness and transparency; purpose limitation; data minimization; accuracy; storage limitation; integrity and confidentiality (security); and accountability.

[IDM.TRAIN.00040] GDPR Data Processing

Is it necessary to process person-relevant data, it MUST be earmarked to a clearly defined business process, which has to be described in the GDPR design decisions. All relevant data MUST be deleted after the processing, if applicable.

[18] https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Sicherheitsberatung/Pentest_Webcheck/Leitfaden_Penetrationstest.pdf?__blob=publicationFile&v=10

[19] https://gitlab.com/gaia-x/data-infrastructure-federation-services/quality-assurance/-/issues

3.5 Design and Implementation

3.5.1 Installation

⏩ **[IDM.TRAIN.00041] Helm/Argo CD Deployment**

All installations MUST be scripted/templated to ensure automated deployment into an enterprise Kubernetes cluster (K8S), SCS K8S demonstration cluster and local K8S development instance. This MUST be ensured uniform over HELM templates which MUST follow uniform across all lots (orientation on existing deployment pipelines from phase I and with alignment with the contractor). The charts MUST be integrable in a ARGO CD Pipeline defined in the gxfs-integration repository[20]. ⏪

3.5.2 Configuration

⏩ **[IDM.TRAIN.00042] Configuration**

All components MUST support one of the major configuration formats (yaml, json, ini, environment variables) wherever configuration is required. If environment variables are overwriting an actively set configuration, a warning SHOULD be logged. ⏪

3.5.3 Distribution

⏩ **[IDM.TRAIN.00043] Helm Repositories**

All component helm charts MUST be available under a helm repository hosted in the gitlab, with different channels for distribution[21]. ⏪

⏩ **[IDM.TRAIN.00044] Istio Resources**

Additionally, the Charts MUST provide Istio Resource (e.g., Authorization Rules, Virtual Services etc.) following the integration pattern specified in the gxfs-integration repo[22]. ⏪

3.5.4 Service Meshing

⏩ **[IDM.TRAIN.00045] Istio Support**

All HELM charts MUST be provided with Istio support aligned together with the contractor. This consists of Authorization Rules, Virtual Service Definitions and other relevant Istio Definitions which are required for integration in a Istio Environment. ⏪

3.5.5 Standard Technology Stack

⏩ **[IDM.TRAIN.00046] Default Toolstack**

[20] https://gitlab.com/gaia-x/data-infrastructure-federation-services/gxfs-integration
[21] https://gitlab.com/api/v4/projects/41175300/packages/helm/Integration/index.yaml
[22] https://gitlab.com/gaia-x/data-infrastructure-federation-services/gxfs-integration

Each development MUST consider the following standard technologies if nothing else is explicitly requested:

Area	Technology
Service Meshing	Istio
Databases	Redis, Mongo, Postgres
Messaging	CloudEvents
Continuous Integration	Argo CD, Gitlab
Installation Templates	HELM
Container	Docker Images (ARM64/AMD64)
Secret Storage	Hashicorp Vault, k8s Secret
UI Technology	React[23]
Ingress Controller	Nginx
API Testing	Postman (manual),
API Design	OpenAPI

Table 2 Standard Technology Stack

The technology stack is mandatory to avoid integration impact.

3.5.6 Metrics

[IDM.TRAIN.00047] Opentelemetry Support

All helm charts/services MUST provide metrics endpoints in opentelemetry[24] format.

3.5.7 Configurability

[IDM.TRAIN.00048] Configuration Profiles

Environment specific parameters MUST be configurable over the helm templates by using profiles. Each component MUST be delivered minimum for profile:

[23] https://react-bootstrap.github.io/
[24] https://opentelemetry.io/docs/

- DEV, a local environment for round trip development and testing
- Acceptance, a restricted resource environment (with minimal system requirements) which can be deployed in cluster (remote or locally)
- Prod, a scalable environment with fault tolerance, HA settings and security hardening

[IDM.TRAIN.00049] Secret References in Helm Charts

The configuration secrets within Helm Charts MUST use secretRefs to support external Secret Management. Clear text secrets within the Helm Charts are not allowed.

3.5.8 Maintainability

[IDM.TRAIN.00050] Microservice Architecture

For a better scale out, maintainability and decentralization, the product architecture MUST have a microservice architecture. Each microservice MUST NOT be limited on the lines of code or number of days to implement. The service "size" SHOULD be oriented on the fine granular business capabilities. (e.g., Order, ListMenu, Payment).

[IDM.TRAIN.00051] Domain Driven Design

To support the microservice architecture within the maintainability, a domain model MUST be declared before realization. The software description MUST explain which domain model was chosen, which services contain it and how it scales. This MUST be documented in the public code repository to support future enhancements for new developers.

3.5.9 Reusability

[IDM.TRAIN.00052] Enterprise Environments

All components MUST be reusable in different enterprise environments by customization and whitelabeling. This means that all components MUST be able to customize and white label the components by configuration settings (e.g., UIs, text labels, endpoints etc.)

3.5.10 Runtime Stability

[IDM.TRAIN.00053] Readiness Checkups

All components MUST reflect - after bootstrap and during runtime - the correctness of the service functionality by reflecting it over health endpoints. The health endpoint MUST return failure (red), if any internal behavior failure or misconfiguration occurs (not just the software running state). This means for instance to check continuously during runtime:

- An unreachable configured Services results in failed state
- Configured Service Endpoints need to be checked for readiness during runtime, if not reachable, it results in failure state
- Check dependent components (Database, Microservice etc.) behind it, if not reachable, it results in failed state

3.5.11 High Availability Concepts

[IDM.TRAIN.00054] Redundant Deployment

Each deployment MUST be configured for a minimum fault tolerance of 2 instances.

3.5.12 Proof of Concept

[IDM.TRAIN.00055] Architecture Changes

All Architecture Changes MUST be aligned with the contractor before implementation.

4 System Features

4.1 Trust Framework and Trust List Provision

4.1.1 Description and Priority

The feature Trust Framework and Trust List Provision is responsible

- for configuring and managing trust frameworks with its corresponding trust lists
- for hosting the trust list VC
- for anchoring the service end points to the DID Document
- transferring the data to be anchored in the DNS using the Zone Manager Handler.

Please refer to the source code on [LIGHTEST.TSPA] for integrating the Trust Framework and Trust List Provision with the Zone Manager.

4.1.2 Stimulus/Response Sequences

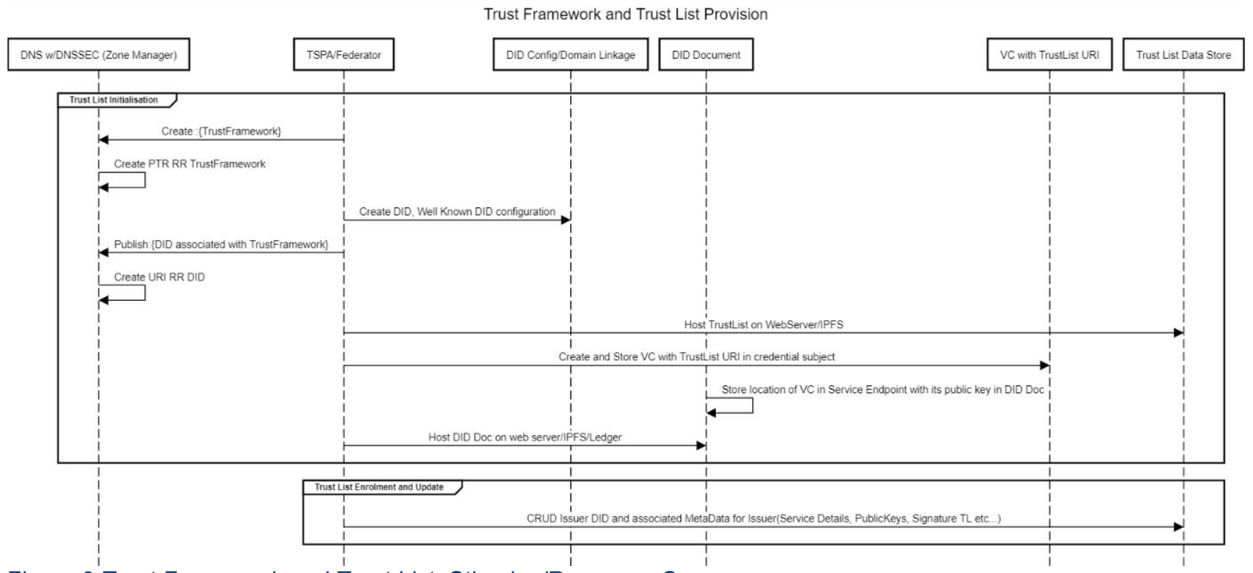

Trust Framework and Trust List Provision
DNS w/DNSSEC (Zone Manager)
TSPA/Federator
DID Config/Domain Linkage
DID Document
VC with TrustList URI
Trust List Data Store
Trust List Initialisation
Create :{TrustFramework}
Create PTR RR TrustFramework
Create DID, Well Known DID configuration
Publish:{DID associated with TrustFramework}
Create URI RR DID
Host TrustList on WebServer/IPFS
Create and Store VC with TrustList URI in credential subject
Store location of VC in Service Endpoint with its public key in DID Doc
Host DID Doc on web server/IPFS/Ledger
Trust List Enrolment and Update
CRUD Issuer DID and associated MetaData for Issuer(Service Details, PublicKeys, Signature TL etc...)

Figure 2 Trust Framework and Trust List: Stimulus/Response Sequences

4.1.3 Functional Requirements

Functional Requirements
Trust Framework Configuration
Trust List Management
Federation specific Framework and Trustlists
Participant specific Frameworks and Trustlists
Organization specific Frameworks and Trustlists
Zone Manager Handler

4.2 Trusted Content Resolver

4.2.1 Description and Priority

The Trusted Content Resolver feature is responsible for the Trust Discovery and Trust Validation functionalities based on the input from the Verifiable Credential / Verifiable Presentation. The trust discovery process is realized through the DNS Resolver and the Universal DID Resolver. This feature MUST also be able to validate the cryptographic signatures of the VC and the trust list. It MUST also be able to differentiate different trust lists based on the data anchored in the VC. It MUST be able to validate the institutional trust of credentials by using issuer details on the trust list. The discovery traverse mechanism MUST be mathematically improved using efficient search algorithms (e.g., Merkle Trees). The service MUST be available as connection libraries in GO, JAVA, Javascript, python languages. The libraries SHOULD be user configurable for different input (e.g., service endpoint type: gxfs-issuer-list, gxfs-self-description) and output formats (e.g., json with defined parameters (service name, digital id, etc.). The Trusted Content Resolver MUST be able to integrate with TSA to validate the trust via libraries mentioned in the above programming languages.The service should also be dockerized and easily integrable with existing systems to validate the trust.

4.2.2 Stimulus/Response Sequences

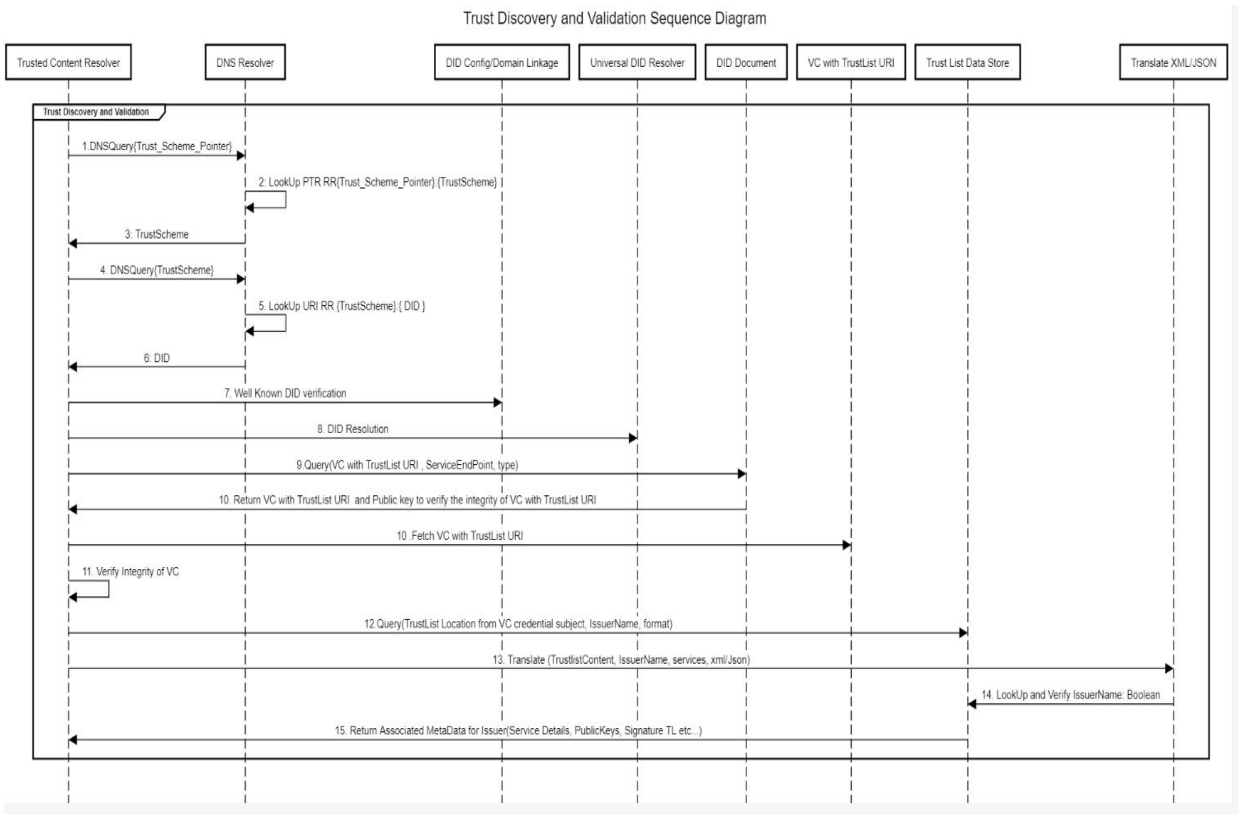

Trust Discovery and Validation Sequence Diagram
Trusted Content Resolver
DNS Resolver
DID Config/Domain Linkage
Universal DID Resolver
DID Document
VC with TrustList URI
Trust List Data Store
Translate XML/JSON
Trust Discovery and Validation
1.DNSQuery{Trust_Scheme_Pointer}
2: LookUp PTR RR{Trust_Scheme_Pointer}:{TrustScheme}
3: TrustScheme
4. DNSQuery{TrustScheme}
5. LookUp URI RR {TrustScheme}:{ DID }
6: DID
7. Well Known DID verification
8. DID Resolution
9.Query(VC with TrustList URI , ServiceEndPoint, type)
10. Return VC with TrustList URI and Public key to verify the integrity of VC with TrustList URI
10 .Fetch VC with TrustList URI
11. Verify Integrity of VC
12.Query(TrustList Location from VC credential subject, IssuerName, format)
13. Translate (TrustlistContent, IssuerName, services, xml/Json)
14. LookUp and Verify IssuerName: Boolean
15. Return Associated MetaData for Issuer(Service Details, PublicKeys, Signature TL etc...)

Figure 3 Trusted Content Resolver: Stimulus/Response Sequences

4.2.3 Functional Requirements

Functional Requirements
Trust Discovery
Validation
Translate XML
Translate JSON

4.3 DNS Zone Management

4.3.1 Description and Priority

The DNS Zone Management feature is responsible for managing the DNS zone file and used for anchoring the trust framework DID information into the zone file. This feature is also responsible for resigning the zone file based on DNSSEC for every new update in the zone file. The implementation approach MUST be based on https://github.com/H2020LIGHTest/ZoneManager with the latest python version. The DNS server used for the backend MUST be NSD and KnotDNS. The service MUST be dockerized.

4.3.2 Stimulus/Response Sequences

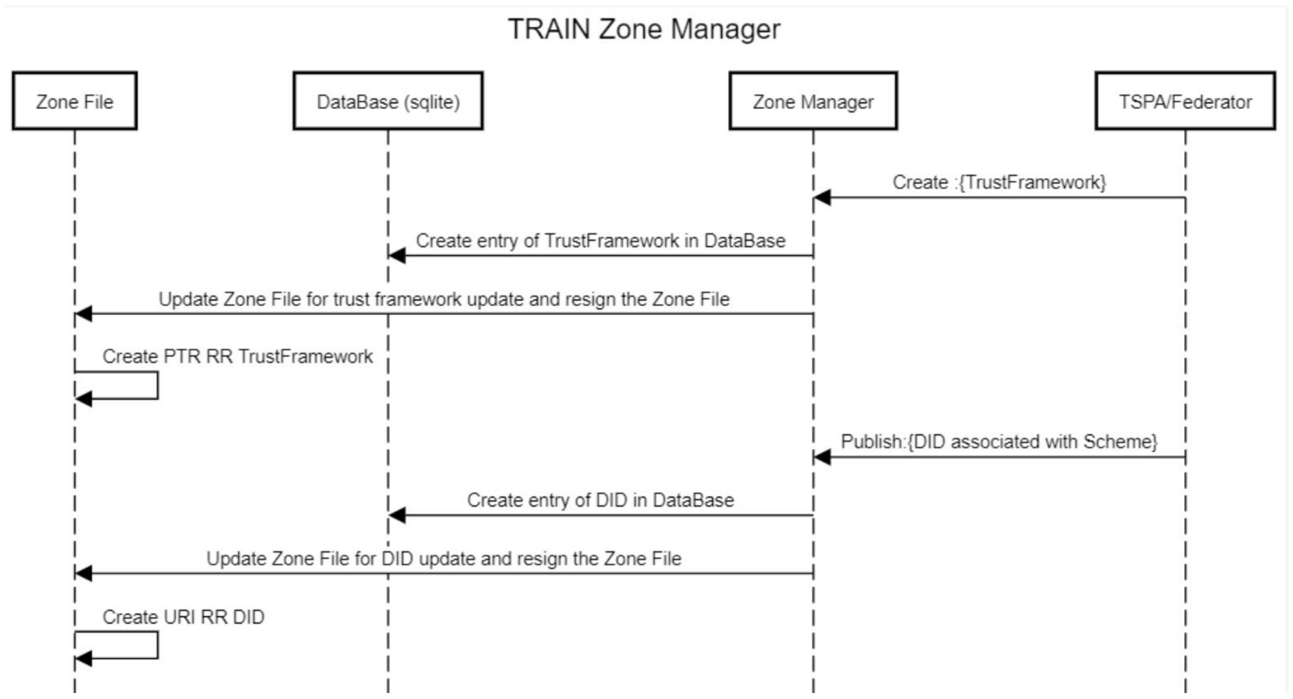

TRAIN Zone Manager
Zone File
DataBase (sqlite)
Zone Manager
TSPA/Federator
Create :{TrustFramework}
Create entry of TrustFramework in DataBase
Update Zone File for trust framework update and resign the Zone File
Create PTR RR TrustFramework
Publish:{DID associated with Scheme}
Create entry of DID in DataBase
Update Zone File for DID update and resign the Zone File
Create URI RR DID

Figure 4 DNS Zone Management: Stimulus/Response Sequences

4.3.3 Functional Requirements

Functional Requirement
Zone Manager

5 Verification

5.1 Core Verification Requirements

All listed verification items/criterias MUST be fulfilled by a demonstration of the implementation within the provided Kubernetes environment.

⏩ **[IDM.TRAIN.00056] Kubernetes Deployment**

If the verification is related to software components, it MUST be deployed in a k8s test cluster and the components must be deployable in a Kubernetes cluster with automated package manager deployment (e.g., Helm).

Docker Compose and other local systems can be used for local development and testing, but it's NOT allowed for a final acceptance demonstration.

5.2 Support for Kubernetes

[IDM.TRAIN.00057] Eventing

All eventings MUST be demonstrated based on cloud events specifications together with the kNative[25] broker in a Kubernetes environment.

[IDM.TRAIN.00058] Config Map Support

Each service MUST be demonstrated up and running in Kubernetes, configured by config maps.

[IDM.TRAIN.00059] Helm Installation

The service installation MUST be demonstrated during HELM install.

[IDM.TRAIN.00060] ArgoCD Integration

The helm chart MUST be able to install inside of argoCD. This includes the usage of the postgres hooks[26] and the provisioning of usable values.yaml(s) for all developed services.

[IDM.TRAIN.00061] SCS Environment

All HELM installations MUST run on SCS (Sovereign Cloud Stack). The final acceptance demonstration cannot be realized on azure, google cloud etc.

5.3 Functionality Acceptance Criteria

[IDM.TRAIN.00062] Zone Manager/TSPA Configuration

A domain was registered, and a DNSSec Server is linked to it. The product Zone Manager was installed on a machine, and it MUST be possible to link the Zone Manager with the DNSSec Server. The Zone Manager Handler MUST be configured to allow access to the zone manager (internal API). After this, using TSPA Trust Frameworks and DIDs can be published over the Zone Manager Handler into the zone file. After the rollout, using the endpoints of the TSPA with Dig commands and queries, the Trust Framework and DID (PTR and URI RR) of the Zone File are resolvable.

[25] https://knative.dev/docs/eventing/
[26] https://gitlab.com/gaia-x/data-infrastructure-federation-services/gxfs-integration/-/tree/main/helm/charts/postgresql-hook

[IDM.TRAIN.00063] Trust Zone Demonstration

Trusted Content publishing and resolution MUST be demonstrated over a public domain with DNSsec functionality. Various entries MUST be added to the zone and be resolved by the Trusted Content Resolver.

[IDM.TRAIN.00064] Trust Framework cross-linking Demonstration

Cross-linking of Trust Frameworks MUST be demonstrated over two public domains with DNSsec functionality. A cross-link from one trust framework on domain 1 (using the PTR Record) MUST reference another Trust Framework on domain 2. Various entries MUST be added to the Trust Frameworks on domain 1 and domain 2 and be correctly resolved by the Trusted Content Resolver.

[IDM.TRAIN.00065] Trust Framework and DID Enrollment

Trust Frameworks and their corresponding DIDs MUST be enrolled using API endpoints offered by the TSPA. The enrolled details MUST be reflected in the database and the DNS Zone file of the Name server.

[IDM.TRAIN.00066] Trusted List Management using API endpoints

The Trust List content MUST be managed using API endpoints. Separate API endpoints MUST be available for performing CRUD Operations.

[IDM.TRAIN.00067] Trusted Content Resolver resolves Trust Zone Pointers

The trusted content resolver is called by a Trust Framework Pointer (URL/DNS Name). The resolver recognizes that a DNS query must be made and it's starting to resolve the records about other Trust Framework Pointers (if applicable). After resolving the entire Zone, the DIDs of the RR/URI Records can be resolved as usual. Output is a List of Pointers which the resolver returns.

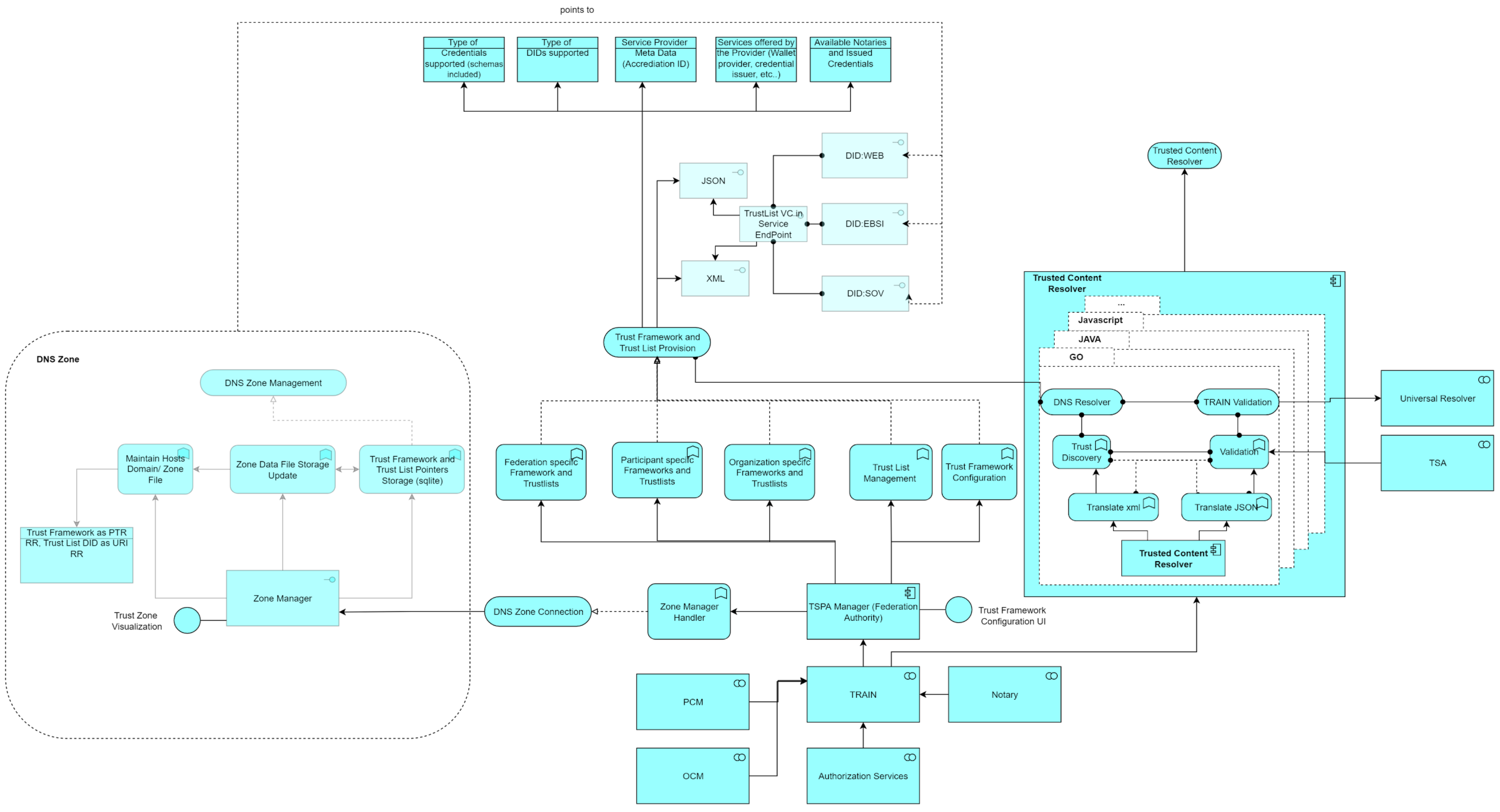
[IDM.TRAIN.00068] Trusted Content Resolver resolves Trusted Content

The trusted content resolver is called by a DID, and a list of requested content types. The resolver resolves the DID, iterates through the service endpoint references, and returns a list of trust lists (no DID documents), regarding the given DID. The DID documents for the resolved references can be found in a second list.

[IDM.TRAIN.00069] Trusted Content Resolver validates institutional trust of a Verifiable Credential

The issuer details of a credential MUST be validated with the resources found from the Trust List. This MUST be demonstrated with Trust Frameworks that are cross-linked to other Trust Frameworks (hosted on different domains) as well as Trust Frameworks that do not include a cross-linking. This MUST be available as integratable libraries for programming languages GO, Javascript, python and Java.

Annex A: TRAIN Overview


points to
Type of Credentials supported (schemas included)
Type of DIDs supported
Service Provider Meta Data (Accrediation ID)
Services offered by the Provider (Wallet provider, credential issuer, etc..)
Available Notaries and Issued Credentials
DID:WEB
JSON
TrustList VC in Service EndPoint
DID:EBSI
XML
DID:SOV
Trusted Content Resolver
Trusted Content Resolver
...
Javascript
JAVA
GO
DNS Resolver
TRAIN Validation
Trust Discovery
Validation
Translate xml
Translate JSON
Trusted Content Resolver
Universal Resolver
TSA
Trust Framework and Trust List Provision
DNS Zone
DNS Zone Management
Maintain Hosts Domain/ Zone File
Zone Data File Storage Update
Trust Framework and Trust List Pointers Storage (sqlite)
Trust Framework as PTR RR, Trust List DID as URI RR
Zone Manager
Trust Zone Visualization
Federation specific Framework and Trustlists
Participant specific Frameworks and Trustlists
Organization specific Frameworks and Trustlists
Trust List Management
Trust Framework Configuration
DNS Zone Connection
Zone Manager Handler
TSPA Manager (Federation Authority)
Trust Framework Configuration UI
PCM
TRAIN
Notary
OCM
Authorization Services

Annex B: Overview GXFS Work Packages

The project “Gaia-X Federation Services” (GXFS) is an initiative funded by the German Federal Ministry of Economic Affairs and Energy (BMWi) to develop the first set of Gaia-X Federation Services, which form the technical basis for the operational implementation of Gaia-X.

The project is structured in five Working Groups, focusing on different functional areas as follows:

Work Package 1 (WP1): Identity & Trust
Identity &Trust covers authentication and authorization, credential management, decentral Identity management as well as the verification of analogue credentials.

Work Package 2 (WP2): Federated Catalogue
The Federated Catalogue constitutes the central repository for Gaia-X Self-Descriptions to enable the discovery and selection of Providers and their Service Offerings. The Self-Description as expression of properties and Claims of Participants and Assets represents a key element for transparency and trust in Gaia-X.

Work Package 3 (WP3): Sovereign Data Exchange
Data Sovereignty Services enable the sovereign data exchange of Participants by providing a Data Agreement Service and a Data Logging Service to enable the enforcement of Policies. Further, usage constraints for data exchange can be expressed by Provider Policies as part of the Self-Description

Work Package 4 (WP4): Compliance
Compliance includes mechanisms to ensure a Participant’s adherence to the Policy Rules in areas such as security, privacy transparency and interoperability during onboarding and service delivery.

Work Package 5 (WP5): Portal & Integration
Gaia-X Portals and API will support onboarding and Accreditation of Participants, demonstrate service discovery, orchestration, and provisioning of sample services.

Identity & Trust	Federated Catalogue	Sovereign Data Exchange	Compliance	Integration & Portal
• Authentication and Authorization • Personal Credential Manager • Organizational Credential Manager • Trust Services	• Core Catalogue Services • User Management and Authentication • Inter-Catalogue Synchronisation	• Data Contract Service • Data Exchange Logging Service	• Continuous Automated Monitoring • Onboarding & Accreditation Workflows • Notarization	• Portal • Orchestration • Workflow Engine / Business Management • API Management • Compliance Documentation Service

Further general information on the Federation Services can be found in [TAD].