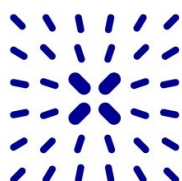




# Identity, Credential and Access Management Document - main version (7c645703)

---



Description	Gaia-X specification to build trusted decentralised digital ecosystems.
Repository	<a href="https://gitlab.com/gaia-x/technical-committee/identity-credentials-and-access-management-working-group/icam">https://gitlab.com/gaia-x/technical-committee/identity-credentials-and-access-management-working-group/icam</a>
Author(s)	Gaia-X European Association for Data and Cloud AISBL
Copyright(s)	©2024 Gaia-X European Association for Data and Cloud AISBL

## Table of Contents

---

## I About

### 1 Identity, Credential and Access Management Document

- 1.1 Publisher
- 1.2 Authors
- 1.3 Contact
- 1.4 Other format
- 1.5 Copyright notice

### 2 Introduction to ICAM

### 3 Adopted Standards and Protocols

- 3.1 Standards for Credentials and Identifiers
  - 3.1.1 JSON-LD
  - 3.1.2 Decentralized Identifiers
  - 3.1.3 JSON Web Token (JWT)/JSON Web Signature (JWS)
  - 3.1.4 JSON Web Key
- 3.2 Protocols
  - 3.2.1 OpenID for Verifiable Credentials (OID4VC)
  - 3.2.2 OpenID Connect for Verifiable Credential Issuance (OIDC4VCI)
  - 3.2.3 OpenID Connect for Verifiable Presentations (OIDC4VP)

### 4 Digital Identities

- 4.1 Overview
- 4.2 Operational Roles of Digital Identities and Key pair Usage
  - 4.2.1 Self-certified identifiers
  - 4.2.2 Self-signed certificates
  - 4.2.3 Trust Service Provider (
- 4.3 Binding Digital Identities to Claims
  - 4.3.1 Different Requirements Based on Use Cases
- 4.4 DID Resolution
  - 4.4.1 Verification Method (JSON Web Key)
  - 4.4.2 eID
  - 4.4.3 eSignature
- 4.5 Implementing Interactions between Machines and Humans
  - 4.5.1 Interactions with Human-in-the-Loop
  - 4.5.2 Interactions with Machines

### 5 Gaia-X Credentials

- 5.1 Overview
- 5.2 Core Data Model Foundations
  - 5.2.1 Namespace Bindings and Contexts
  - 5.2.2 Identifiers
  - 5.2.3 Type Property
- 5.3 Credential Format Specification
  - 5.3.1 Encoding requirements
  - 5.3.2 Credential Structure
  - 5.3.3 Credential Subject
- 5.4 Verifiable Credentials
  - 5.4.1 Standard Verifiable Credential
  - 5.4.2 Enveloped Verifiable Credential
- 5.5 Verifiable Presentations
  - 5.5.1 Standard Verifiable Presentation
  - 5.5.2 Enveloped Verifiable Presentation
- 5.6 Issuer Requirements
- 5.7 Additional Features
  - 5.7.1 Integrity of Related Resources
  - 5.7.2 Credential Lifecycle and Status

### 6 ICAM Semantic Model

- 6.1 Trust Scope Credential
  - 6.1.1 Trust Scope Credential specialisation examples
  - 6.1.2 Federation using Trust Scope Credentials
- 6.2 Party Credential
  - 6.2.1 Private Party Credential
  - 6.2.2 Public Party Credential
  - 6.2.3 Party Credential Specialisation examples
- 6.3 Signature Credential
  - 6.3.1 Multiple Signatures using SignatureCredential specializations
- 6.4 Ecosystem Onboarding and Offboarding using ICAM Semantic Model
- 6.5 Delegating Access Rights
  - 6.5.1 Types of Credentials and Issuers
  - 6.5.2 Implementation Factors

### 7 Changelog

- 7.1 2025 November Release (25.11)

- 7.2 2024 July release (24.07)

## I. About

---

## 1 Identity, Credential and Access Management Document

---

## 1.1 Publisher

---

Gaia-X European Association for Data and Cloud AISBL  
Avenue des Arts 6-9  
1210 Brussels  
[www.gaia-x.eu](http://www.gaia-x.eu)

## 1.2 Authors

---

Gaia-X European Association for Data and Cloud AISBL


## 1.3 Contact

---

<https://gaia-x.eu/contact/>

## 1.4 Other format

---

For convenience a PDF  version of this document is generated [here](#).

## 1.5 Copyright notice

---

©2025 Gaia-X European Association for Data and Cloud AISBL

This document is protected by copyright law and international treaties. This work is licensed under a [Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License](#). Third party material or references are cited in this document.



## 2 Introduction to ICAM

---

This document defines the Gaia-X Identity, Credential, and Access Management (ICAM) specifications.

Its purpose is to describe the essential components for authentication, authorisation, and access management, aiming to enable trusted identity operations and interoperability across ecosystems.

The document introduces key concepts such as decentralized identifiers, verifiable credentials and verifiable presentations, trust scope credentials and party credentials, and explains how they are applied in onboarding procedures and ecosystem transactions. It also describes how different types of digital identities can be classified and used in various contexts.

These specifications provide the credential formats, identifiers, and a semantic model that ensure interoperability within the Gaia-X ecosystem. They also contribute to the definition of Gaia-X technical compatibility, which serves as a basis for achieving interoperability across ecosystems.

Note: This document does not prescribe how to replace or operate an existing IAM system within a participant's environment. Instead, it provides specifications that enable portability, interoperability, and trust across multiple infrastructures and ecosystems.

### 3 Adopted Standards and Protocols

---

This section defines the key reference standards upon which the specifications in this document are built. It does not impose Gaia-X design decisions (e.g. format or method selections). Those decisions, profiles, and constraints will be introduced in subsequent chapters.

## 3.1 Standards for Credentials and Identifiers

### 3.1.1 JSON-LD

**JSON-LD (JavaScript Object Notation for Linked Data)** is a [W3C](#) standard for expressing Linked Data using familiar JSON syntax.

It uses a context to map JSON properties to IRIs (Internationalized Resource Identifiers), enabling unambiguous interpretation of terms, and allows [data](#) to interoperate across systems by providing semantic meaning.

JSON-LD is fully compatible with JSON, making it easier to integrate into existing JSON-based systems while supporting richer [linked-data](#) semantics.

### 3.1.2 SHACL (Shapes Constraint Language)

**SHACL (Shapes Constraint Language)** is a [W3C](#) standard for defining shapes, i.e., constraints on [RDF](#) graphs, and validating that [data](#) ([RDF](#) graphs) conform to those shapes. Shapes express constraints on property paths, cardinalities, datatype restrictions, and logical combinations, and may use extension languages like SPARQL to express more complex validation rules.

### 3.1.3 Decentralized Identifiers

**Decentralized Identifiers (DIDs)** are URIs that enable verifiable, self-sovereign digital identity without requiring a single centralised registry or [authority](#), although they may be anchored in distributed or decentralised registries. A DID resolves to a DID Document that typically contains one or more verification methods, each associated with cryptographic key material.

These verification methods commonly include key pairs consisting of a private key and a public key, mathematically linked through asymmetric cryptographic algorithms. The private key is securely generated and kept under the sole control of the DID controller, while the corresponding public key is published in the DID Document as part of the verification method.

The verification methods are used to prove control over the identifier and to enable trustworthy interactions such as the sharing and retrieval of Verifiable Credentials (VCs) and other resources (see DID specifications [here](#)).

This cryptographic binding ensures that only the DID controller holding the private key can create digital signatures that verifiers can validate using the public key from the DID Document, thus establishing the cryptographic proof of control and authenticity within the Gaia-X ecosystem.

### 3.1.4 JSON Web Token (JWT)/JSON Web Signature (JWS)

**JSON Web Token (JWT)** is an open standard (RFC 7519) for encoding a set of claims in a JSON object that can be integrity-protected or signed. **JSON Web Signature (JWS)** (RFC 7515) defines how to apply a digital signature over a payload using JSON-based structures, making it possible to verify authenticity and integrity.

### 3.1.5 JSON Web Key

**JSON Web Key (JWK)**, defined in [RFC 7517], is a JSON-based standard for representing cryptographic keys and key sets. It provides a standardised and interoperable way to publish and exchange public keys and key metadata, typically used to verify JSON Web Tokens (JWTs) and other digital signatures within identity and [credential](#) systems.

### 3.1.6 W3C Verifiable Credentials Data Model v2.0

Verifiable Credentials (VCs) as defined in the ([W3C Verifiable Credentials Data Model v2.0](#)) are used to express assertions (claims) about one or more subjects in a cryptographically verifiable, machine-readable form.

VCs include claims, [issuer](#) metadata, validity constraints, and proof mechanisms to ensure integrity and authenticity.

A VC can be shared directly or embedded in a [Verifiable Presentation \(VP\)](#), which a [holder](#) uses to present one or more credentials in a controlled manner.

A Verifiable Presentation enables a [verifier](#) to confirm that the claims in its enclosed credentials originate from the asserted [issuer\(s\)](#).

### 3.1.7 W3C VC-Bitstring Status List

The Bitstring Status List ([W3C VC-Bitstring Status List](#)) defines a privacy-preserving, space-efficient mechanism to encode the revocation or suspension status of many Verifiable Credentials in a single bitstring.

Each [credential](#) is associated with a specific bit index: a value of 1 denotes "revoked" or "suspended", and 0 denotes "valid"; the bitstring (often compressed) is published as a Verifiable Credential so verifiers can check status without contacting each [issuer](#) individually.

## 3.2 Protocols

### 3.2.1 OpenID for Verifiable Credentials (OID4VC)

The term "OID4VC" commonly refers to the family of OpenID/OAuth-based protocols that support the issuance and presentation of Verifiable Credentials and Verifiable Presentations.

### 3.2.2 OpenID Connect for Verifiable Credential Issuance (OIDC4VCI)

(**OIDC4VCI**) is based on the [OAuth 2.0 specification](#) and allows an [issuer](#) to communicate with a [holder](#) and their [wallet](#) in order to issue Verifiable Credentials in a secure manner.

OID4VCI defines an OAuth-protected API through which an [issuer](#) can issue Verifiable Credentials to a [holder's wallet](#). The specification leverages OAuth 2.0 (rather than redefining it) to obtain authorisation for [credential](#) issuance.

 **OIDC4VCI** is published as a version 1.0 specification, intended as a stable baseline for implementers.

### 3.2.3 OpenID Connect for Verifiable Presentations (OIDC4VP)

(**OIDC4VP**) extends OAuth 2.0 / OpenID Connect to allow a [holder](#) (via a [wallet](#)) to present one or more Verifiable Credentials to a [verifier](#) as a Verifiable Presentation.

 **OIDC4VP** is published under the OpenID "version 1.0" set of drafts.

## 4 Digital Identities

---

## 4.1 Overview

Digital identity consists of the digital attributes and credentials that an entity (such as a person, organisation, or service) uses for authentication, authorisation, and trusted access to resources. A digital identity may be represented by one or more verifiable credentials issued by trusted parties, encoding claims about the identity subject. Each digital identity is anchored with a cryptographic key pair: the private key remains under the holder's control, while the corresponding public key—typically published through a verification method within a DID Document—enables others to verify signatures and authenticate issuers.

Within an ecosystem, every issued credential must link the issuer's identity to the subject's unique identity. The Ecosystem Governance Authority defines the authorised credential types, identity categories, and permitted operations within the trust framework.

The following section outlines how different types of key pairs and trust levels are applied within the ecosystem to support identity and credential operations.

## 4.2 Operational Roles of Digital Identities and Key pair Usage

Within the ecosystem, digital identities serve distinct purposes. Identities issued by the Ecosystem Governance Authority during onboarding support Identification, Authentication, and Authorisation (IAA) operations, such as secure intra-agent communication and policy enforcement. Identities used for electronic identification (eID) and electronic signatures enable trusted transactions, with assurance levels defined by the governance authority. eID services may also simplify participant verification during onboarding.

The workflow diagram below illustrates how digital identities can be used to sign credentials, how they connect to Trust Service Providers, and how verification methods are linked across the ecosystem architecture.

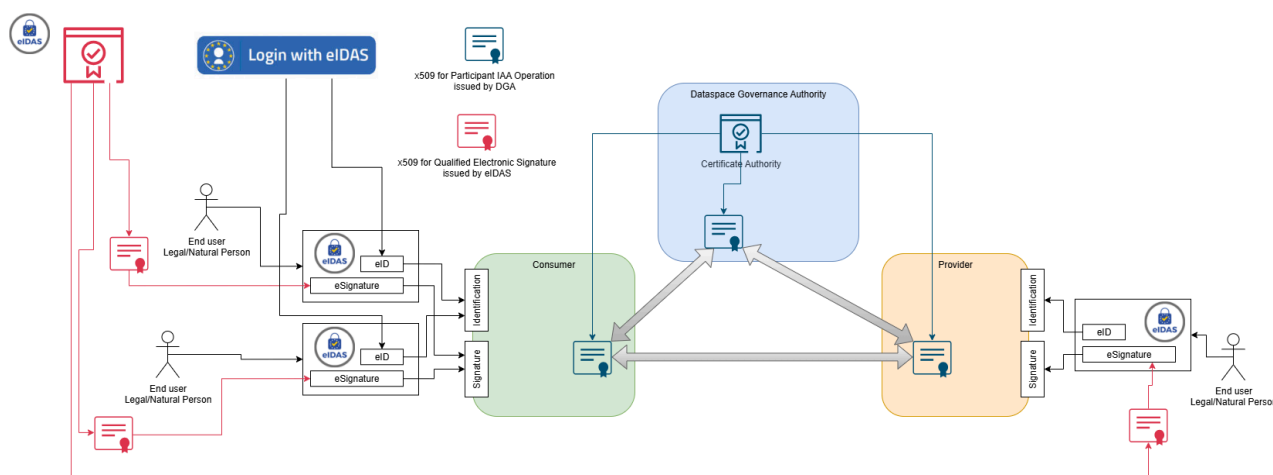


Figure 4.2 - Use of Digital Identities

Depending on the use case, key pairs may be self-issued, pseudonymous (not linked to a verified identity), or certified by a Trust Service Provider (TSP); different TSPs can be engaged according to assurance or regulatory needs.

### 4.2.1 Self-certified identifiers

Valid cryptographic key pairs may exist without being bound to a certificate or an explicit identifier of a person or entity. For example, a DID Key can serve as a verification method: it identifies the public key and algorithm, and allows the subject to present claims linked to that key when required.

### 4.2.2 Self-signed certificates

In this model, entities create and sign their own X.509 certificates rather than obtaining them from a third-party Certificate Authority. This represents a lower level of external trust assurance compared to Trust Service Provider-issued certificates. The ecosystem's governance authority defines which identity types and contexts may rely on self-signed certificates within the trust framework.

### 4.2.3 Trust Service Provider (TSP) Keypair

In this case, a qualified trust service provider issues a certificate and provides a qualified signature device (or hardware-protected private key) which ensures non-direct access to the private key. Because the TSP conducts know-your-customer / know-your-business (KYC/KYB) checks and maintains revocation mechanisms, the certificate is considered authentic and valid. The TSP or issuing certificate authority publishes endpoints (such as certificate revocation lists or OCSP) that verifiers may query to validate status.

## 4.3 Binding Digital Identities to Claims

When an issuer provides a claim, the issuer's digital identity MUST be verified to ensure that the claim originates from a trusted and authorized source. Issuers using verified digital identities SHOULD comply with the eligibility and assurance requirements defined by the ecosystem's governance framework. When X.509 certificates are legally recognized and bound to verified legal identities, signatures on issued claims elevate assurance and legal value for relying parties, especially when the issuer is recognised in the ecosystem Registry and exposes CRL/OCSP for status.

For example, when a digital contract is represented as a verifiable credential, signing it with a qualified digital identity (e.g., a legally recognised electronic signature) provides it with legal enforceability.

Verified claims are then evaluated by access control engines (e.g., policy decision points) to authorise participant actions and resource access.

### 4.3.1 Different Requirements Based on Use Cases

Requirements for digital identities vary depending on the use case and ecosystem context.

Example: University Scenario: In a university ecosystem, participants such as students, professors, and administrators require different levels of identity assurance and access. While a simple key-pair–based identity may suffice for internal access control, more sensitive contexts, such as a medical university where student doctors access patient information, require higher assurance and legally qualified identities, such as certificates issued under an eIDAS-compliant trust service.

## 4.4 DID Resolution

When a verifiable credential is submitted for verification (for example, to a Digital Clearing House), the DID resolution process retrieves the associated DID document of the credential's issuer. This process exposes the issuer's verification methods (such as public keys) and service endpoints required to validate the credential's signature and to confirm the issuer's trust relationship within the ecosystem.

If the credential relies on an X.509 certificate chain, the verifier MUST validate that the chain terminates at a root certificate authority recognised by the ecosystem's Registry. In business or licensing scenarios, a verifiable credential may include claims such as the organisation's name, legal address, or contact information. These attributes are typically verified by Trust Service Providers (TSPs) acting on behalf of legal representatives.

Some processes may require additional authorisation or assurance steps, for which the verifier uses the cryptographic material referenced in the DID document to confirm identity control and integrity.

Each verifiable credential MUST reference a single controlling digital identity, ensuring that when the DID is resolved, the corresponding controller manages the associated cryptographic keys. The resolution process reveals not only the cryptographic material but also metadata about where and when the DID was resolved, including the DID document's last update time, controller, and verification methods. This provides a full chain of resolvable identities that can be used by any participant who wants to assign or revoke roles and permissions to another participant, enabling comprehensive audit trails and trust validation across the ecosystem.

### 4.4.1 Verification Method (JSON Web Key)

In the Gaia-X Ecosystem, the designated verification method for digital identities is the JSON Web Key (JWK) format, as defined in RFC 7517.

A subject's identity record (such as a DID document or equivalent reference) MUST publish a publicKeyJwk parameter or reference a jwks\_uri that resolves to a JSON Web Key Set (JWKS). During verification, the verifier retrieves the JWK or JWKS endpoint, validates the key identifier (kid), and uses the public key material to verify the signature on the credential. This ensures that the key is controlled by the specified identity controller, aligns with the ecosystem's trust registry, and is suitable for cryptographic proof of the credential's authenticity and integrity.

Implementations SHOULD provide a jwks\_uri and status endpoints so verifiers can fetch current keys, confirm key rotation, and check revocation before authorising access.

#### 4.4.1.1 Assurance and Trust Mechanisms

**KYC/KYB** processes: Used by Trust Service Providers to verify the identity of natural persons and legal entities.

**Certificate chain validation**: Verifies that any included X.509 certificate has a trust path to a root certificate authority recognised by the ecosystem registry.

**Revocation checking**: Uses mechanisms such as Certificate Revocation Lists (CRL) or the Online Certificate Status Protocol (OCSP) to confirm that certificates or keys have not been revoked.

**DID resolution**: Retrieves the DID document containing the verification method (JWK), public keys, and service endpoints required for cryptographic verification.

##### 4.4.1.1.1 SELF-SOVEREIGN IDENTITY (SSI) IMPLEMENTATION

In SSI-enabled ecosystems, participants can use party credential specialization to delegate a subset of claims received from the governance authority. A single keypair can serve dual functions for both authentication and signing operations. Participants retrieve access tokens to access relying parties, using publicly verifiable claims for identification. When implementing SSI with verifiable credentials, all claims MUST be bound to a single keypair rather than multiple keypairs to maintain cryptographic integrity and clear accountability.

## 4.5 eIDAS Integration

In Gaia-X, a digital identity may be used (i) to verify claims as a Verifiable Credential and (ii) to sign artefacts or transactions, with the ecosystem specifying the required level of assurance for each operation.

The eIDAS Regulation establishes a framework for electronic identification and trust services across the European Union. Within an ecosystem, eIDAS-compliant mechanisms can be leveraged for both identity and signature purposes, ensuring legal recognition and cross-border interoperability.

An entity possessing an eIDAS-qualified certificate may use it to sign a Verifiable Credential (VC). When the certificate is legally binding under EU law, the resulting signature grants the credential legal value. Any Trust Service Provider (TSP) supporting eIDAS MUST be qualified and listed under the EU Trusted List (EUTL), and must protect private keys within certified qualified signature creation devices (QSCD) or secure hardware modules.

eIDAS defines two principal service domains:

Trust Services used for creating and validating electronic signatures, seals, timestamps, and related services.

Electronic Identification Services (eID) used for authenticating and identifying natural or legal persons across EU Member States.

For example, an individual in one EU country may use their state-issued eID to authenticate and access official data (such as pension or university records) in another Member State.

These services can be integrated into ecosystem frameworks by applying the reusable building blocks provided under the eIDAS and Digital Identity Toolbox.

### 4.5.1 eID

An eID (electronic identification) is used solely for authentication and identification, not for electronic signing. For example, a citizen uses their eID to log in to a public-administration portal and legally identify themselves.

Each Trust Service Provider offering eID services MUST be officially recognised under eIDAS and listed in the EU Trusted List, enabling mutual recognition of digital identities across Member States.

This allows individuals to use their national eID to access online services in other EU countries. (For additional details, refer to the eID and eID FAQ.)

### 4.5.2 eSignature

An electronic signature (eSignature) represents a person's intent to approve or agree to the content of a document or dataset. Like its handwritten equivalent, an eSignature captures the signatory's intent to be legally bound by the signed content. Within eIDAS, eSignatures may be used to create and verify qualified electronic signatures on Verifiable Credentials or other digital attestations.

To enable legally valid signing, a participant must use [eIDAS](#)-qualified certificates and qualified [trust](#) providers that issue or manage Qualified Electronic Signatures (QES). While [eIDAS](#) is an EU framework, qualified [trust](#) providers in other jurisdictions can also support equivalent mechanisms for cross-border use cases.

Levels of electronic signatures under [eIDAS](#):

Simple Electronic Signature (SES): any electronic [data](#) used to indicate agreement (e.g., typing a name in an email).

Advanced Electronic Signature (AdES): uniquely linked to and capable of identifying the signatory; tampering with the signed [data](#) is detectable.

Qualified Electronic Signature (QES): an AdES created with a qualified signature creation device (QSCD) and based on a qualified [certificate](#).

Remote Qualified Signature: a QES managed remotely by a qualified Trust Service Provider through a certified QSCD service.

eSignatures can also be combined with Verifiable Credentials, enabling legally binding attestations within decentralised identity ecosystems. (For further information, see the [eSignature FAQ](#).)

## 4.6 Implementing Interactions between Machines and Humans

---

In the Gaia-X digital identity context, interactions may involve either a human in the loop or fully automated machine-to-machine processes. The specifications must support both modes while ensuring legal compliance and [trust](#) through strong digital identity proofs. Below, we outline how to handle each mode:

### 4.6.1 Interactions with Human-in-the-Loop

---

Certain identity transactions or [Data Usage Agreement \(DUA\)](#) workflows require explicit human approval before completion. In these scenarios, automation must pause to obtain human consent or a signature.

#### 4.6.1.1 Legal Requirements for Human Sign-off

Under the EU [eIDAS](#) framework, some approvals must be provided by a natural person. For example, a Qualified Electronic Signature (QES), equivalent to a handwritten signature, can only be created by a verified human signer with a qualified [certificate](#). When a legally binding signature is needed (e.g., signing a [contract](#) or consent), the workflow MUST pause and hand off to the user to sign with their personal credentials.

#### 4.6.1.2 Cryptographically Verifiable Human Binding

The system SHOULD use credentials that link an individual's identity to their signature. A qualified [certificate](#) is issued to each signer, ensuring high assurance. The [certificate](#)'s private key is under the user's sole control (e.g. on a smartcard or hardware token), and any signature they generate can be verified against that [certificate](#). This linkage provides non-repudiation, proving that the specific individual approved the transaction.

#### 4.6.1.3 Qualified Providers and Personal Wallets

Implementations SHOULD integrate qualified [trust](#) service providers and user-bound identity wallets. Gaia-X recognises [eIDAS](#)-qualified [certificate](#) issuers as [trust](#) anchors in its framework. Each user keeps personal credentials and keys in an identity [wallet](#) (such as the EU Digital Identity Wallet). When a sensitive operation (like signing a DUA) requires human action, the system prompts the user via their [wallet](#) to review details and apply a qualified signature. The workflow then resumes only after a valid human signature or consent, ensuring compliance with legal requirements.

### 4.6.2 Interactions with Machines

---

Many Gaia-X scenarios rely on fully automated exchanges between machines or services without human involvement. These use decentralised identity credentials, digital signatures, and [trust](#) frameworks so that machines can authenticate each other.

#### 4.6.2.1 Automated Credential Exchange

Gaia-X uses Verifiable Credentials in machine-to-machine ([M2M](#)) processes instead of human checks. A service (the [holder](#)) sends a signed [credential](#) to another (the [verifier](#)), whose software immediately validates the signature and checks that the [issuer](#) is trusted. This happens through cryptographic verification, so one service can instantly confirm another's credentials (like compliance or attributes) without manual steps. For more information see the "Gaia-X Credentials" chapter.

#### 4.6.2.2 Establishing Trust in [M2M](#) Interactions

Without human oversight, machines rely on [trust](#) frameworks and cryptography to decide if a [credential](#) is valid. The [verifier](#) first checks that the [credential](#)'s [issuer](#) is recognised (e.g. listed in a Gaia-X registry of trusted issuers). Gaia-X rules require each [issuer](#) to be a registered [trust](#) anchor or linked to one (for example via an [eIDAS](#)-qualified CA or a known DID). The machine accepts the [credential](#) only if this [issuer trust](#) is established and other checks (signature validity, expiration) pass. This automated enforcement ensures that machine exchanges maintain the same [trust](#) level as human-reviewed processes.

## 5 Gaia-X Credentials

---

## 5.1 Overview

---

This chapter defines how Verifiable Credentials (VCs) are modelled, structured, and used within the Gaia-X [trust](#) framework, following the [W3C VC Data Model 2.0](#). It details the encoding, required properties, and processing rules for issuing and verifying Gaia-X compliant credentials. The goal is to ensure semantic consistency and technical interoperability across participants in the Gaia-X ecosystem.

## 5.2 Core Data Model Foundations

---

### 5.2.1 Namespace Bindings and Contexts

---

Gaia-X credentials use JSON-LD contexts and namespace bindings as part of their [credential](#)/presentation representation (e.g. via `@context`).

On the level of the Verifiable Presentation and the Verifiable Credentials contained in the Verifiable Presentation, a Gaia-X Credential MUST adhere to the vocabulary of the Verifiable Credentials Data Model, i.e., use terms from the <https://www.w3.org/2018/credentials#> namespace.

To enable human authors of Gaia-X Credentials to write down these terms conveniently, they MAY, by using the `@context` keyword at the level of the Verifiable Presentation, e.g.:

- reference the **JSON-LD context** provided by the Verifiable Credentials Data Model (<https://www.w3.org/ns/credentials/v2>) as in the initial example listing, or
- define their own context, which:
- defines the above namespace as the default vocabulary using the `@vocab` keyword, or
- maps the above namespace to a designated prefix, e.g. `"cred"`.

Similarly, the claims about any [credential](#) subject MUST adhere to the vocabulary of the Gaia-X Credential Schemas published in the [Gaia-X Ontology](#).

### 5.2.2 Identifiers

---

The `@id` MUST be present and unique for a given [issuer](#).

The `@id` keyword can be aliased to `id` (consequently we may also use this alias).

It is up to the [issuer](#) to decide if the `@id` is a resolvable [URL](#) or not.

Each of the following MUST have a different identifier:

- a Verifiable Presentation
- a Verifiable Credential inside a Verifiable Presentation
- the subject of a Verifiable Credential, i.e. the Conceptual Model entity about which claims are made.

Gaia-X Credentials MAY reference other Gaia-X Credentials. Consider, for example, a *ServiceOffering* that:

- is provided by a *Provider*,
- is a composition of other *ServiceOfferings*, or
- has an aggregation of *Resources*.

### 5.2.3 Type Property

---

The `@type` property MUST be present in Verifiable Presentation and Verifiable Credentials. The expected values for the first `@type` property are:

- `"VerifiablePresentation"` for a Verifiable Presentation
- `"EnvelopedVerifiablePresentation"` for an Enveloped Verifiable Presentation encoded as a [VC-JWT](#)
- `"VerifiableCredential"` for a Verifiable Credential
- `"EnvelopedVerifiableCredential"` for an Enveloped Verifiable Credential encoded as a [VC-JWT](#)

This `@type` can be followed with one or more [credential](#) related types ( i.e. `@type: [ 'Verifiable Credential', 'gx:LegalPerson' ]`).

The `@type` keyword can be aliased to `type` (consequently, we may also use this alias).

The expected values for the `@type` property of a [credential](#) subject are given by the taxonomy of classes defined in the [Gaia-X Ontology](#), having the superclasses `Participant`, `ServiceOffering` and `Resource`.

An ecosystem MAY define additional subclasses of these by defining further shapes and hosting them in its Registry.

In the future, Gaia-X and other ecosystems may also define additional, more specific [credential](#) types.

The shapes of Gaia-X Credentials, to be used as the vocabulary of the claims about [credential](#) subjects, MUST be available in the form of [SHACL](#) shapes (cf. the [W3C Shapes Constraint Language SHACL](#)) in the Gaia-X Registry or in the Catalogue of a Federation.

At any point when Gaia-X Credentials are created or received, a certain set of [SHACL](#) shapes is known, which forms a *shapes graph*. A Gaia-X Credential forms a *data graph*. For compliance with Gaia-X and/or a different ecosystem, this *data graph* MUST be validated against the given *shapes graph* according to the [SHACL](#) specification.

## 5.3 Credential Format Specification

---

Gaia-X Credentials conform to the [W3C Verifiable Credential Data Model 2.0](#), and use the Gaia-X Ontology which is available via the Gaia-X Registry.

### 5.3.1 Encoding requirements

---

Gaia-X mandates [credential](#) encoding via [VC-JWT](#), with the cryptographic proof constructed using JSON Web Signature (JWS) so that the payload, header and signature can be verified for authenticity and integrity.

Below is an example showing a JSON document in its native form (e.g. document.json), and how it is encoded as a [VC-JWT](#) via JOSE-based signing (and optionally encryption). This demonstrates how a plain JSON Verifiable Credential can be transformed into a cryptographically verifiable form under the [VC-JWT/JOSE/COSE](#) framework.

```

document.json
1  {
2    "@context": [
3      "https://www.w3.org/ns/credentials/v2",
4      "https://w3id.org/gaia-x/development#"
5    ],
6    "@type": [
7      "VerifiableCredential",
8      "LegalPerson"
9    ],
10   "@id": "https://example.org/legal-participant/68a5bbea9518e7e2ac1cc75bcc8819a7edd5c4711e073ffa4bb260034dc6423c/data.json",
11   "issuer": "did:web:example.org",
12   "validFrom": "2024-01-01T12:26:22.601516+00:00",
13   "validUntil": "2024-04-01T12:26:22.601516+00:00",
14   "credentialSubject": {
15     "id": "https://example.org/legal-participant-json/68a5bbea9518e7e2ac1cc75bcc8819a7edd5c4711e073ffa4bb260034dc6423c/data.json",
16     "type": "gx:LegalPerson",
17     "gx:legalName": "Example Org",
18     "gx:legalRegistrationNumber": {
19       "id": "https://example.org/gaia-x-legal-registration-number/68a5bbea9518e7e2ac1cc75bcc8819a7edd5c4711e073ffa4bb260034dc6423c/data.json"
20     },
21     "gx:headquarterAddress": {
22       "gx:countrySubdivisionCode": "FR-75"
23     },
24     "gx:legalAddress": {
25       "gx:countrySubdivisionCode": "FR-75"
26     }
27   }
28 }

```

Once it has been encoded using the [VC-JWT specification](https://www.w3.org/TR/vc-jose-cose/#securing-json-ld-verifiable-credentials-with-jose), it will become the following Verifiable Credential:

```

eyJhbGciOiJIUzI1NiIsInR5cCI6Ii99vzhZG61vsGBlL0OgFhLWwprLmW7YklnKD4QoTv-RxBX3JCakUCE_ukSoeUoERUFJKFEfbyAAMjBnRZsbeH7xt5MLrs482TYx2HhSdNkxVZU4UHkOhGSAuoGfZrHV5e7XT4N2q4XIRN3iihYbw4-27sSDgNwOkuY34IwWRZSQSP3PoBneJcHOKDvEPgKvOt8V9ZM78wbyH9Nlae8qAEKwVNF61cs3XQx6-0bql6hOn9I4C93ShXrqmjgTA

```

This VC-JWT can be analysed and verified with tools such as [JWT.io's debugger](https://jwt.io/#debugger-io?)

```

eyJhbGciOiJIUzI1NiIsInR5cCI6Ii99vzhZG61vsGBlL0OgFhLWwprLmW7YklnKD4QoTv-RxBX3JCakUCE_ukSoeUoERUFJKFEfbyAAMjBnRZsbeH7xt5MLrs482TYx2HhSdNkxVZU4UHkOhGSAuoGfZrHV5e7XT4N2q4XIRN3iihYbw4-27sSDgNwOkuY34IwWRZSQSP3PoBneJcHOKDvEPgKvOt8V9ZM78wbyH9Nlae8qAEKwVNF61cs3XQx6-0bql6hOn9I4C93ShXrqmjgTA). The following private and public key have been used to sign this VC-JWT example and **all the examples in the upcoming chapters**. This key pair will have the following ID : 'did:web:example.org#JWK-RSA'. You will notice it in the 'kid' header claim of the example JWT below.

```

```

-----BEGIN PRIVATE KEY-----
MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSIAgEAAQBAQC7VJTU9Us8ckj
MzEYyjiWAA4R4/M2bS1GB417NXp98C3SC6dWmVvDicitGeur78JbnvJZHCsuYEvu
Nm0Sfm76odFvAp8G0iZ5xjZmSnXyCdFevGhLaOvZMaQ8s+CL0yS56YyCFGeJZ
qgtzJ6GR3eqoYSW9b9UMvKbPZODStWSN9g3P7JRFDO5VoTQCAWbFnQjDH5UIj
p2PKSQnSjP3AJLQNFNe7br1XbrhV//e0+151mlpGSDCv3E0DDFCWDT9cXDTTIR
ZVEIR2BwpZ00KE/ZO/BvnhZYL71oZV34bKwJQIT6V/isSMahdsAASAcP4ZTGTwi
VuNd9tybAgMBAEECggEBAKtjmaS6tkK8BIPXCITQ2vz/N6uxDeS35mXpqasqskV
IaIdgg/sWqpXDbXr93otIMLlWmM+XOCqMDgSXKejLS2x4GDJ1ZTXg++OAMJ8
sJ74PwZVDOfmCEQ/7wXs3+cbnXhKIO8Z036q92Qc1+N87S138nkGaoABH9CN83H
mQqt4B7UdHuzlRe/me2Pghlq5ZBzj6h3ppPoGzEP+x3I9YmK8t/1cN0pq1dQwY
dglfjackLu/2qH80MCF7lyQaseZUOjYrCLtSDj/lix/hzDEUFP0CjFDgTpz3cw
ta8+oE4wHCo1i1/4TIPkwmX4qSxtmw4aQpZ7IDQvEcGyEABKNTHC02gsC2I9PQ
DM/8Cw0O983WCDY+oi+7JPINAw5DYBqEZB1QYd06YD16XIC/HAZMsMku1na2T
NdrwiwnQWzoev3g2S7gRDoS/FCJSI3j+kgta7Qmzlgk1TxODN+G1H91HW7t
017VnL27IwYo2qRRK3zpxqUIPUCgYEAxOoQs2reBQGMVZnApD1jeq7n4MvNlCPv
t8b/eU9iUy6Y4MjSo/AUBiYzXm8ubbgAlwz2VSVunD2tOplHyMurtCtObARVdu
AhCndKa9gAgfb3xw1IKbuQ1u4f1FJ13VtumfQn//LIH1B3rXhcdyo3/vttEk
48RakUKCI8CgYEaZv7W3CO0IDdQd935DdtKBFRApRAlspQUnzMi5eSHMD/ISL
DY5iIQhI83D4bwXq07qQoSBSNP7Dv3HYuqMhfoDaegrIBuJlFVq9qPVRnK
xt1I2HgXObvhhOt+9in1Bz+Y9J9UzC8500Qo6A+CmtHEy4aZ2kjHjECgYEA
mNS44A8Fks8Js1RieK2LniBxMgmYml3pVlKGNzmg7H2+cvPlhPlzluwytXywh
2bzbsYefYx3EoEVGMEpPhoarQnyYkJO4gWE2o5Te6T5mJSZGIQJq4ZB2DF
et6INsK0oG8XGSpQyQh3RUyEKYqkBBFopqWpbiEscGyanM3DQJ3fJoSnaXmhr
Vblovci5OxkFkEhskAJfTe0V8Fsz1C2aSeRkSQG0Q0tmJzBES1R6kqHnicd
TQrKhArgLXX43cdjFTRkFWDBE/CkVKNoref1nhaGQCpRj2Kukj1FhI9Cnc
dn/RsYEONbwQSIjMPkvxf+8HQ==
-----END PRIVATE KEY-----

-----BEGIN PUBLIC KEY-----
MIIBjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEaU1SU1LVLPHCoZMxH2Mo
4lgOEePzNm0tRgeLezV6fAtOgunVTLw7onLrnq0/izW7yWR7QkrmBL7JTKEn5u
+qKhhwbkFBstls+bmY2Zkp18gnTxKLoS21FczGkPLgPzskuemMghRniWaoLcyeh
kd3qqGENW/VDL5AaWTg0nLVkR9z+4ORQzUvAE8AkAFmzZow3x+VJYKdjykkJ
0IT9wCSODRTXu269V264Vf/3vrredZIKRkgwL9NkNwXfG0x/FwO05UWVRlkdg
cKWtjBP2dFwVZ4WwC+9aGvY+Gyn1o0CLef4rEjGoXbAAEgAgeGUXrcIbXbc
mwiDAQAB
-----END PUBLIC KEY-----

```

### 5.3.2 Credential Structure

#### 5.3.2.1 Header

The VC-JWT header MUST contain the following fields:

- **alg**, the signature algorithm (ie. `PS256`)
- **typ**, the media type of the JWT which MUST be set to `vc+jwt`
- **cty**, the content type of the payload which MUST be set to `vc`
- **kid**, the verification method using a `did:web` or URL reference to the **verification method in a DID document**
- **iss**, the **issuer's** DID address

Additional headers that aren't described in the [VC-JWT](#), [JWT](#) or [JWS](#) specifications SHOULD be ignored.

### 5.3.2.2 VC-JWT Payload

The payload of the [VC-JWT](#) is a standard verifiable [credential](#) with claims as described in the [Verifiable Credential Data Model v2.0](#) specification.

Some payload claims from the [JWT specification](#) MUST be replaced by the described verifiable [credential](#) fields such as:

- `jti` will be replaced by the verifiable [credential](#)'s `id` or `@id`
- `sub` will be replaced by the verifiable [credential](#)'s `credentialSubject.id` or `credentialSubject.@id`

The `vc` and `vp` payload claims MUST NOT be present.

**i** The `iat` and `exp` payload claims represent the JWT's signature validity period whereas the `validFrom` and the `validUntil` verifiable [credential](#) payload claims represent the verifiable [credential](#)'s [data validity period](#). Therefore these claims can cohabit in the payload.

If the `@type` is "VerifiableCredential", the property `credentialSubject` MUST be defined. The value of `credentialSubject` can be a [Credential](#) or an array of [Credentials](#). A Verifiable Credential MUST have :

- an `@id`,
- an `issuer` matching the `iss` JWT header,
- a `@type`, and
- a `credentialSubject` object or a `credentialSubject` array.

**NB:** The `@id` and `@type` keywords are aliased to `id` and `type` respectively. Consequently, we may also use these aliases.

### 5.3.2.3 Signature

The last element of a [VC-JWT](#) is the signature which is cryptographically secured to ensure integrity hence making the Verifiable Credential tamper-proof.

A [JWS](#) is signed using the [issuer](#)'s private key and can be verified by using the [issuer](#)'s public key which is obtainable through the [issuer](#)'s DID document (referenced in the `kid` [JWS](#) header).

[VC-JWT](#) signatures are created following the [JSON Web Signature \(JWS\)](#) specification. **Many libraries are available online** to manage [JWS](#) creation.

## 5.3.3 Credential Subject

The `credentialSubject` can be an object or array of objects, containing claims.

The claims about one [Gaia-X](#) entity may be spread over multiple [Credentials](#) and their subjects.

Each [credential](#) subject MUST have an `@id`.

A [credential](#) subject MAY be described *by value*, i.e., by stating one or more claims about it in place. In this case, it MUST have a `@type` as specified below.

Alternatively, a [credential](#) subject MAY be described *by reference*. In this case, the `@id` MUST be resolvable to an [RDF](#) resource that has the same `@id`, a `@type`, and one or more claims. See [Identifiers](#) section for more details.

The value of the `@type` property dictates the vocabulary available in the [Gaia-X Ontology](#) for the definition of claims about the [credential](#) subject. E.g., `LegalPerson`, `ServiceOffering`, `DataResource`, ...

#### Example of credentialSubject

```
{
  "@id": "https://example.com/legalPersonABC?vcid=c93b5075b3988eda4a529afce7e7c127f607b55dc08bb12e8c9adc9e33fe814",
  "@type": "gx:legalPerson",
  "gx:legalName": "Legal Person ABC",
  "gx:legalRegistrationNumber": {
    "@id": "https://gaia-x.eu/legalRegistrationNumber_VC.json"
  },
  "gx:headquarterAddress": {
    "gx:countrySubdivisionCode": "FR-IDF"
  },
  "gx:legalAddress": {
    "gx:countrySubdivisionCode": "FR-IDF"
  }
}
```

## 5.4 Verifiable Credentials

### 5.4.1 Standard Verifiable Credential

Verifiable Credentials are encoded as [Json Web Tokens](#) as described in the [VC-JWT specification](#). This type of proof is an [enveloping proof](#).

A [JWT](#) consists of a header, a payload and a signature, each element being separated by a dot (`.`).



### Example Verifiable Presentation

Below is an example of a Verifiable Presentation containing the example from the [Enveloped Verifiable Credential](#enveloped-verifiable-credential) chapter.

```

{
  "@context": [
    "https://www.w3.org/ns/credentials/v2"
  ],
  "id": "https://gaia-x.eu/verifiablePresentation/1",
  "type": [
    "VerifiablePresentation"
  ],
  "verifiableCredential": [
    {
      "@context": "https://www.w3.org/ns/credentials/v2",
      "id":
      "data:application/vc+jwt;eyJhbGciOiJIUzI1NiIsInR5cCI6ImlzLWZkZXkK2pb24rand0liwiY3R5IjoiaWMrbGQranNvbiliImtpZC16ImRzDp3ZWl6ZXhhbXBsZS55cmc6bGlnYXVxQWQ6d2V0mV4YV1wbGUuY28xUChPearyZzxiBV_3J5hAe8XvfnFo9Y__Lcbu0jNMsU2kKhl9otw9Ll4C8lZ9Qsqd52QFCvkbvtcvX_3lJpzyxSS7Tx0XAPPwYbYv_u7tgygPRvwmQG99Q651y62tQGA_B6Eag",
      "type": "EnvelopedVerifiableCredential"
    }
  ]
}

```

## 5.5.2 Enveloped Verifiable Presentation

Just like an [Enveloped Verifiable Credential](#), an Enveloped Verifiable Presentation is a representation of a [Verifiable Presentation](#) in the form of a basic JSON object containing an `application/vp+jwt` [data URL](#).

This `data URL` expresses a JWS secured Verifiable Presentation. The same headers as [Verifiable Credentials](#) are used in a Verifiable Presentation `VC-JWT` except:

- the `typ` header is set to `vp+jwt`
- the `cty` header is set to `vp`

### Example Enveloped Verifiable Presentation

Below is a representation of the [Verifiable Presentation example](#verifiable-presentation) as an Enveloped Verifiable Presentation.

```

{
  "@context": "https://www.w3.org/ns/credentials/v2",
  "id":
  "data:application/vp+jwt;eyJhbGciOiJIUzI1NiIsInR5cCI6ImlzLWZkZXkK2pb24rand0liwiY3R5IjoiaWMrbGQranNvbiliImtpZC16ImRzDp3ZWl6ZXhhbXBsZS55cmc6bGlnYXVxQWQ6d2V0mV4YV1wbGUuY28xUChPearyZzxiBV_3J5hAe8XvfnFo9Y__Lcbu0jNMsU2kKhl9otw9Ll4C8lZ9Qsqd52QFCvkbvtcvX_3lJpzyxSS7Tx0XAPPwYbYv_u7tgygPRvwmQG99Q651y62tQGA_B6Eag",
  "type": "EnvelopedVerifiablePresentation"
}

```

## 5.6 Issuer Requirements

The `issuer` property MUST be present in a Verifiable Credential and a Verifiable Presentation. The value of the `issuer` property MUST be a resolvable [URI](#).

The supported schemes for `issuer`'s [URI](#) are:

- `https`
- `did`. The supported [DID methods](#) are:
  - `web`

The DID Methods supported by the different Gaia-X software releases are listed in the [Architecture Document](#).

### Note

#### Temporal Validity:

`validFrom` Property - The `validFrom` property is mandatory in a Verifiable Credential and a Verifiable Presentation.

`validUntil` Property - The `validUntil` property is recommended in a Verifiable Credential and a Verifiable Presentation.

## 5.7 Additional Features

### 5.7.1 Integrity of Related Resources

In order to enable reference to objects - Verifiable Credentials or [credential](#) subject - which are not under control of the same issuers, it is recommended to specify an `@sri` [Subresource Integrity](#) attribute to enable the verification of the integrity of the referenced object.

The `sri` attribute is computed by taking the hash of the referenced normalised JSON object.

The JSON object is normalised following the JSON Canonicalization Scheme (JCS) defined in the [RFC 8785](#).

### Example of "sri" attribute

#### SRI attribute

```

{
  "id": "https://example.com/ABC",
  "sri": "sha256-b9a822666c3569a8ae80c897a1984f68bbdf1f18141caadb3f168b1c0b9aa36"
}

```

### 5.7.2 Credential Lifecycle and Status

Verifiable Credentials are a fundamental component of secure [data](#) and identity systems, enabling the issuance and presentation of trustworthy and tamper-proof credentials. However, in dynamic and evolving environments, it is crucial to establish mechanisms for the timely revocation or suspension of these credentials in case of compromised or outdated information.

A Verifiable Credential can have one of the following statuses:

- expired if the `validUntil` attribute is older than the current datetime or the `certificate` containing the key used to sign the `claim` has expired.
- revoked
- if the keypair used to sign the array is revoked.
- if the `credentialStatus` has the `statusPurpose` property set to "revocation" and the value of `status` at position `credentialIndex` is true
- suspended if the `credentialStatus` has the `statusPurpose` property set to "suspension" and the value of `status` at position `credentialIndex` is true
- deprecated if another verifiable `credential` with the same identifier and the same signature `issuer` has a newer issuance datetime.
- active only if none of the above.

The use of Credential Status Lists (CSL), specifically the [W3C Verifiable Credentials Bitstring Status List](#), addresses this need by providing a standardised approach to manage and communicate the revocation status of Verifiable Credentials.

When a Verifiable Credential is issued, the `issuer` has the option to embed a reference to the Credential Status List (CSL) entry associated with the `credential`. This reference, often in the form of a Uniform Resource Identifier (URI), enables relying parties (commonly verifiers) to promptly determine the current status of the `credential`'s validity.

To validate the Verifiable Credential, the relying `party` retrieves the referenced Credential Status List entry using the provided URI. This entry contains information about the status of the `credential`, allowing to check if the `credential` is still valid, has been revoked/suspended, or has any other relevant status.

Relying parties can periodically update their local copy of the Credential Status List from trusted sources to ensure they possess the most current revocation status information. This practice prevents reliance on outdated or incorrect information, enhancing the overall security of the ecosystem.

```

party_credential_revocation.json
1  {
2    "@context": [
3      "https://www.w3.org/ns/credentials/v2",
4      "https://w3id.org/gaia-x/development#"
5    ],
6    "id": "https://did.actor/alice/credentials/status/3",
7    "type": ["VerifiableCredential", "BitstringStatusListCredential"],
8    "issuer": "did:web:did.actor:alice",
9    "issued": "2021-04-05T14:27:40Z",
10   "credentialSubject": {
11     "id": "https://example.com/status/3#list",
12     "type": "BitstringStatusList",
13     "statusPurpose": "revocation",
14     "encodedList": "H4slAAAAAAAA-3BMQEAAADCoPVPbQwfoAAAAAAAAAAAAAAAAAAIC3AYbSVksAQAAA"
15   }
16 }

```

## 6 ICAM Semantic Model

---

## 6.1 Trust Scope Credential

The Trust Scope Credential is based on the [Verifiable Credentials Data Model v2.0](#) and has the purpose of providing a machine-readable representation of the accreditation of a Trust Service Provider for a specific scope. This credential enables the usage of Party Credentials, described later in this chapter, by defining their accredited issuers. Furthermore, the Trust Scope Credential supports the cooperation and interoperability between organisations/ecosystems/[data](#) spaces, by easing the use of external Trust Service Providers.

The TrustScopeCredential mainly defines:

- The Scope within which the Trust Service Provider is accredited (within which the issued credentials are considered valid).
- The TrustedIssuers entitled to issue Credentials in the above Scope
- The Vocabularies (expressed in [SHACL](#)) that semantically define the Credentials which can be issued by the TrustedIssuer in the defined Scope.
- The Trusted List used to check DIDs issued by the Trust Service Provider.

The `Trust Scope Credential` is defined by the following attributes:

Attribute	Type.Value/Voc	Mandatory	Comment
<code>gx:scopeDescription</code>	String	No	A description of the scope of the Trust Service Provider
<code>gx:trustIssuers</code>	DID[]	Yes	A list of resolvable link(s) to the <a href="#">issuer verificationMethod</a> to be used to uniquely identify ONE and ONLY key pair
<code>gx:vocabularies</code>	URI[]	No	A list of URIs pointing to the vocabularies/schemas ( <a href="#">SHACL</a> ) semantically describing the Trust Service Provider's scope
<code>gx:trustedListKind</code>	KindOfTrustedList	No	Which kind of implementation of the trusted list is used to check DID issued by Trust Anchors
<code>gx:trustedListEndpoint</code>	URI	No	The address of the above trusted list

The KindOfTrustedList type defines the list of the identified implementations of Trusted Lists:

- Gaia-X Trusted List Generic REST API Specification
- Gaia-X IPFS (with ETSI TS 119 612 format)
- TRAIN
- EBSI
- (other possible implementations)

Important Note The *Gaia-X Trusted List Generic REST API Specification* is meant to be a simple [API contract](#) that for a given DID can return "true" if it is still valid. This will be very useful to enable integration of any custom trusted list not included in the defined list.

### 6.1.1 Trust Scope Credential specialisation examples

Several specialisations of the TrustScopeCredential can be easily defined, for example:

#### 6.1.1.1 Organization Trust Scope

This specialisation is a self-issued [credential](#) that entitles the issuing organization to define its *Roles/Identity Attributes* to be used by identify and authorise its parties:

- scope - *Organization Credential Management* (OCM).
- trusted issuers - the organisation itself.
- vocabularies - defines the semantics of *Roles/Identity Attributes* and Domain Specific Credentials that are valid in the defined scope.
- trusted list - to assign and revoke *Roles/Identity Attributes* and Domain Specific Credentials to its parties (users, natural persons, endpoint services, etc) by issuing PartyCredentials.

#### 6.1.1.2 Gaia-X Compliance Trust Scope

This [credential](#) is issued by Gaia-X and entitles an organisation to issue attestations about specific claims in the context of Gaia-X Compliance.

- scope - issue attestations used to attest Gaia-X Compliance.
- trusted issuers - the organisation accredited by Gaia-X to issue attestations in the defined scope.
- vocabularies - defines the semantic applicable to issue attestations in the defined scope.
- trusted list - to assign and revoke attestations issued to Gaia-X Providers. Trust Anchor Credential specialisation examples

#### 6.1.1.3 Ecosystem Trust Scope

This specialisation is a self-issued [credential](#) that entitles an Ecosystem operator to define:

- scope - manage an *Ecosystem* (onboarding/offboarding/role assignment etc.).
- trusted issuers - the Ecosystem itself.
- vocabularies - defines the semantic of *Roles/Identity Attributes* and Domain Specific Credentials valid in the defined scope.
- trusted list - to assign and revoke *Roles/Identity Attributes* and Domain Specific Credentials to its members (other Participants) issuing MembershipPartyCredentials.

## 6.1.2 Federation using Trust Scope Credentials

The TrustScopeCredential is designed with interoperability in mind and easily enables implementation of selective, bidirectional and monodirectional [trust](#), and a federation between two or more trusted scopes is the perfect use case. Below are some examples:

### 6.1.2.1 Monodirectional selective federation between 2 Ecosystems

Ecosystem A can enable federation with Ecosystem B by simply trusting the trusted issuers of the EcosystemTrustScope issued by Ecosystem B and selecting which subset of its Domain Specific Credentials to [trust](#).

### 6.1.2.2 Bidirectional full federation between 2 Ecosystems

Ecosystem A enables federation with Ecosystem B by fully trusting the EcosystemTrustScope issued by Ecosystem B, and vice versa.

### 6.1.2.3 Federation managed by an operator that selectively trusts multiple Ecosystems

An Ecosystem operator maintains the list of the trusted issuers and relative Domain Specific Credentials of the trusted EcosystemTrustScope issued by Ecosystems.

Important Note In all the above examples, the *trusting* relationship can be concretely implemented by issuing a [verifiable credential](#) whose [credentialSubject](#) is the ID of the EcosystemTrustScope. Alternatively, a more fine-grained TrustRelationCredential could be designed for that purpose.

## 6.2 Party Credential

The Party Credential is based upon the [Verifiable Credentials Data Model v2.0](#) and is the basis for all IAA Parties such as Natural Persons, Services, Legal Persons, etc. The general purpose [party credential](#) is intended to be extended into specialized Credentials (see Party Credential Specializations)

The `PartyCredential` is defined by the following attributes:

Attribute	Type.Value/Voc	Mandatory	Comment
<code>gx:holder</code>	DID	Yes	A resolvable link to the <code>holder</code> <a href="#">verificationMethod</a> to be used to uniquely identify ONE and ONLY key pair
<code>odrl:hasPolicy</code>	<a href="#">policy[]</a> in <a href="#">ODRL</a>	No	A list of <code>policy</code> expressed using <a href="#">ODRL</a>
<code>gx:identityAttributes</code>	String[]	No	A list of literals representing Identity Attributes to be used in a ABAC context
<code>gx:identityRoles</code>	String[]	No	A list of literals representing Identity Roles to be used in a RBAC context
<code>gx:parentPartyCredential</code>	<a href="#">URI</a>	Yes if delegated by another existing Party Credential	A resolvable link to the parent Party Credential where the <code>gx:holder</code> MUST be equal to the <code>signer(issuer verificationMethod)</code> of this <a href="#">credential</a>

### VERY IMPORTANT NOTE

The `credentialSubject.id` MUST identify the DID Document that contains the `verificationMethod` (keypair) referenced by `gx:holder` property that *belong to/is controlled by* the Credential Holder. With this solution, the PartyCredential [VC](#) can be either publicly published in case it contains no sensitive [data](#) or kept private in case it contains PII - Personal Identifiable Information.

### 6.2.1 Private Party Credential

The Party Credentials containing PII are not considered to be published and reachable via their id to everybody, instead, they are intended to be stored in secure storage such as a [wallet](#), secure storage device, secure vault storage, etc. An example of this type of [credential](#) is the `NaturalPersonCredential`, issued by a Legal Participant to one of his users/employees, with the purpose to entitle a Natural Person to interact with Relying Parties(RP) in a certain context This [credential](#) contains Name, Surname, `identityAttributes`, Roles, etc. and MUST NOT be published as Public Party Credentials (see later in this section) are. "Selective disclosure" during interaction with RP can be considered.

#### 6.2.1.1 Private Party Credential Example

The following `NaturalPersonPartyCredential` example represents the scenario where the Participant `did:web:did.actor:alice` is issuing a [credential](#) that asserts several claims (`givenName`, `surname`, `idRoles` etc). This Credential is not published (the id is not public) and MUST be stored in the [wallet](#) of the [holder](#).

Note that the `gx:holder` property is a `did:key` referencing a keypair owned by the Holder that identifies the public key of the target [wallet](#).



This credential is issued by a LegalParticipant that runs an ecosystem to another Participant in order to attest his Membership status.

## 6.3 Signature Credential

The Signature Credential is a Verifiable Credential introduced to express digital signatures in a machine-readable and interoperable form enabling several scenarios ranging from data transaction permissions to agreement signing. It allows participants to cryptographically sign digital resources and bind the signature to the identity of the signer with verifiable integrity.

By using a verification method associated with a certificate issued under a recognized legal trust framework (e.g. eIDAS), the Signature Credential may achieve legal equivalence with handwritten signatures, depending on the assurance level (e.g., Qualified Electronic Signature).

A specialization of this credential, Signed Agreement Credential, introduces revocability to model agreements whose validity may be rescinded over time. These credentials can be issued independently by each participant and later aggregated into a Verifiable Presentation, thus enabling decentralized signature workflows for contracts such as data usage agreements.

Important note: the issuance of a signature credential from a digital identity bound to a human-controlled device/wallet, can be used to realise the "human-in-the-loop" interaction.

This approach promotes semantic clarity, cryptographic assurance, and interoperability across use cases involving legally binding actions, whether initiated by humans or automated agents. It also enables trust anchors and policies defined in the Gaia-X Trust Framework to validate each Signature Credential independently.

### 6.3.1 Multiple Signatures using SignatureCredential specializations

The SignatureCredential in the trust model defined by the ICAM Document, represents a general purpose and machine readable *signature* that Participants are asked to provide in several contexts, one of the most important of these contexts is represented by the **Data Transaction** section of the Data Exchange Document(version...) where this credential will be used to sign **Data Usage Agreement** and Data Product Usage Contract.

The concept behind this credential is the same as the credentials issued by the compliance service, to issue Participant Credentials and Service Offering Credentials (using the "credentialSubject" claim), where the credentialSubject consists in:

Attribute	Type.Value/Voc	Mandatory	Comment
type	string	Yes	Indicating the <i>type</i> of the subject to be Signed
id	<u>URI</u>	Yes	A resolvable single <u>URI</u> as defined in the <u>VC</u> Specification identifying the subject to be Signed
digestSR I	String	Yes	Integrity of related Resources as defined in <u>W3C VC Data Model v2.0</u> to ensure that referenced resource is not changed/modified/tampered(still identical in time)

Another important factor is represented by the fact that, issuing and signing this credential with a verificationMethod bound to a certificate that has legal relevance (e.g. eIDAS) gives it the same level of trust, enabling the possibility to check the `gx:legalValidity` property of the Signature Check Type (see `gx:signers` property).

#### 6.3.1.1 SignedAgreementCredential

The SignedAgreementCredential is a specialization of SignatureCredential that represents a *Revocable Signed Agreement* given by a participant to a specific subject (e.g. Data Usage Agreement, GDPR Agreement, etc.) and the only difference is that it can be revoked.

#### 6.3.1.2 Data Usage Agreement Example

In a Data Transaction the Data Provider(did:web:provider.com) and a Data Consumer(did:web:consumer.com) must agree to a Data Usage Agreement (id "https://provider.com/data-usage-contract.654321") and this is done treating "Signatures" as a Verifiable Credential that each Participant can issue and sign. In this way, when all Participants have issued their SignatureCredential referencing the contract it will be possible to create Verifiable Presentations that contain all the Signatures.

Here is an example of how CredentialSignature can be used:

```
{
  "@context": [
    "https://www.w3.org/ns/credentials/v2",
    "https://w3id.org/gaia-x/development#"
  ],
  "type": [
    "VerifiableCredential"
  ],
  "id": "https://consumer.com/data-usage-contract-signatures.123456",
  "issuer": "did:web:consumer.com",
  "validFrom": "2023-07-28T12:31:49.074Z",
  "validUntil": "2023-10-26T12:31:49.074Z",
  "credentialSubject": {
    "type": "gx:dataUsageAgreement",
    "id": "https://provider.com/data-usage-contract.654321",
    "digestSRI": "sha384-IHKDhH0mSc6pRx8PhDOMkNtSI8bOfsp4gINbUrw71nXXL13nTqNJorP3Nx+ArVK"
  },
  "credentialStatus": {
    "id": "https://consumer.com/status/1#127",
    "type": "BitstringStatusListEntry",
    "statusPurpose": "revocation",
    "statusListIndex": "127",
    "statusListCredential": "https://consumer.com/credentials/status/1"
  }
}
```

```

{
  "@context": [
    "https://www.w3.org/ns/credentials/v2",
    "https://w3id.org/gala-x/development#"
  ],
  "type": [
    "VerifiableCredential"
  ],
  "id": "https://provider.com/data-usage-contract-signatures.987",
  "issuer": "did:web:provider.com",
  "validFrom": "2023-07-25T10:31:49.074Z",
  "validUntil": "2023-10-25T10:31:49.074Z",
  "credentialSubject": {
    "type": "gx:dataUsageAgreement",
    "id": "https://provider.com/data-usage-contract.654321",
    "digestSRI": "sha384-IHKDHH0mSc6pRx8PhDOMkNTS18b0fsp4gINbUrw71nXXL13nTqNJoRp3Nx+ArVK"
  },
  "credentialStatus": {
    "id": "https://provider.com/status/3#125221",
    "type": "BitstringStatusListEntry",
    "statusPurpose": "revocation",
    "statusListIndex": "125221",
    "statusListCredential": "https://provider.com/credentials/status/3"
  }
}

```

## 6.4 Ecosystem Onboarding and Offboarding using ICAM Semantic Model

The ICAM Semantic Model enables onboarding and offboarding of participants in an ecosystem, ensuring consistent trust and interoperability across federations.

Defining a Trust Scope establishes the boundaries of mutual recognition within an ecosystem by specifying the types of credentials, policies, and trusted issuers relevant for participation.

A Trusted Issuer is a recognized entity authorized to issue verifiable credentials that conform to the defined Trust Scope, such as Party Credentials and Onboarding Credentials. During onboarding, an entity receives an Onboarding Credential, a specialization of the Party Credential, which attests that the entity has successfully met all policy, identity, and compliance requirements defined by the ecosystem. This credential enables the entity to operate as a trusted participant, interoperable with other data space actors under the same trust framework.

Conversely, offboarding is achieved through the revocation of the Onboarding Credential, thereby terminating the entity's ability to participate in trust-based interactions. The revocation event is registered within the trust infrastructure, ensuring that all dependent systems can verify the updated status through verifiable credential mechanisms.

## 6.5 Delegating Access Rights

The ecosystem uses decentralized identity and access management based on Self-Sovereign Identity (SSI) to enhance security, privacy, and flexibility. An organization can provide secure access to their employees for various clouds (e.g., email, document storage, collaboration tools) without relying on a central identity provider. The HR department or respective manager is able to revoke or change roles for an employee if an employee is no longer working for the organization or changes designations.

### 6.5.1 Types of Credentials and Issuers

Employee Credential (specialised PartyCredential)

The company HR department issues an Employee Credential to each employee. The employee credential contains the employee's unique identifier, role, department and unique company ID (unique identifier used in the onboarding phase of the company to the ecosystem). The Employee Credential is used to verify the employee's identity and affiliation with the company. The Company ID MUST be registered in the membership trustlist of the ecosystem and in parallel used in the separate issued membership credential. The HR department MUST revoke the employee's credential in case he/she leaves the company, and the employee SHOULD NOT be able to access any cloud services on behalf of the company.

#### Note

The Employee Credential type and the attributes MUST be standardised. The `EmployeeCredential` is a prerequisite for receiving `AccessEntitlementCredentials` and is used to establish the employee's identity within the organization.

- Type: `EmployeeCredential` (Specialised Party-Credential)
- Issuer: HR Department
- Subject: Employee
- Company ID: Global unique identifier for the company or organization (eg. GLEIF ID, VAT.), same across all employees. The Company ID should be used in the membership trustlist and membership credential used to verify the authenticity of the credential.
- Emp Attributes: Like, Employee ID, Name, Role, Department

Access Entitlement Credentials

The Manager issues `AccessEntitlementCredential` to grant access to specific cloud services based on the employee's role and responsibilities. The manager can revoke these credentials to dynamically manage access rights. The `EmployeeCredential` is a prerequisite for receiving `AccessEntitlementCredentials`. The manager MUST be entitled to issue such a credential; therefore, the manager MUST adhere to the following conditions:

- MUST have a valid `ManagerCredential` issued by the HR department
- MUST be a managing member of the ecosystem
- Company ID MUST be part of the credential to ensure the manager is a valid member of the company.

Therefore, the manager is registered in the membership trustlist of the specific ecosystem services and in parallel he/she holds a valid issued membership credential.

- Type: `AccessEntitlementCredential` (for each access right individual credentials MUST be issued)
- Issuer: Employee Manager
- Attributes: Employee ID, Access Rights (specific cloud services the employee can access)
- Purpose: Specifies the services the employee is entitled to access.
- Scope: Issued by the respective manager to grant access to specific services based on the employee's role and responsibilities.
- Revocation: Manager who can revoke the credential to dynamically manage access rights. <!--could they differ from the Issuer/Employee manager?-->

- Employee ID: The employee's `EmployeeCredential` is a prerequisite for receiving `AccessEntitlementCredentials`.

#### Authentication Process

1. Employee Presents Credentials: When accessing a service, the employee presents their `EmployeeCredential` and `AccessEntitlementCredential`.
2. Verification: The service verifies:
3. The authenticity of the credentials (ensuring they are issued by the HR department and the respective manager).
4. The validity of the credentials (not revoked).
5. The attributes match the access control policies (e.g., the employee's role and entitlements align with the service being accessed).

#### Revocation Process

Revocation List or Registry: The HR department and managers can revoke credentials by adding them to a revocation list or registry that the services check during the authentication process.

- Issuer: HR Department for `EmployeeCredential`, Manager for `AccessEntitlementCredential`.
- Mechanism: Could use a decentralized identifier (DID) revocation mechanism or a centralised revocation service, depending on the architecture.

### 6.5.2 Implementation Factors

---

- Decentralized Identifiers (DIDs): Use DIDs for employees, HR, and managers to ensure a decentralized and self-sovereign identity system.
- Verifiable Data Registry: Implement a verifiable data registry to store DIDs for validation.
- Privacy: Ensure that the system only reveals the minimum necessary information for authentication and access control to protect employee privacy.
- Interoperability: Design the credential schema to be interoperable with W3C Verifiable Credentials standards and compatible with the cloud services' access control mechanisms.
- Trust An ecosystem may have many trustlists (Gaia-X , XFCs Ecosystem Member, Service Manager).

## 7 Changelog

---

## 7.1 2025 November Release (25.11)

---

The ICAM document is restructured with additional topics. Below are the ToCs from previous version and the newly refactored ICAM document to give a clarity on what has been removed or added/updated.

Previous version ToC	Current New ToC
1. Identity, Credential and Access Management Document	1. Identity, Credential and Access Management Document
1.1 Publisher	1.1 Publisher
1.2 Authors	1.2 Authors
1.3 Contact	1.3 Contact
1.4 Other Format	1.4 Other Format
1.5 Copyright Notice	1.5 Copyright Notice
2. Introduction and Scope of the Document	2. Introduction to ICAM
3. Credential Format	3. Adopted Standards and Protocols
3.1 Gaia-X Credential Format	3.1 Standards for Credentials and Identifiers
3.1.1 Gaia-X Credential Example	3.1.1 JSON-LD
3.2 Digital Signature Standard	3.1.2 <u>SHACL</u> (Shapes Constraint Language)
3.3 Decentralized Identifiers	3.1.3 Decentralized Identifiers
3.3.1 Verification Methods	3.1.4 JSON Web Token (JWT)/JSON Web Signature (JWS)
3.4 Use of Identifiers in Gaia-X Credentials	3.1.5 JSON Web Key
3.5 Verifiable Credential and Verifiable Presentation	3.1.6 <u>W3C</u> Verifiable Credentials Data Model v2.0
3.5.1 namespace Bindings and Contexts	3.1.7 <u>W3C</u> <u>VC</u> -Bitstring Status List
3.5.2 Identifiers	3.2 Protocols
3.5.3 Integrity of Related Resources	3.2.1 OpenID for Verifiable Credentials (OID4VC)
3.5.4 Types	3.2.2 OpenID Connect for Verifiable Credential Issuance (OIDC4VCI)
3.5.5 Issuers	3.2.3 OpenID Connect for Verifiable Presentations (OIDC4VCP)
3.5.6 validFrom	4. Digital Identities
3.5.7 validUntil	4.1 Overview
3.5.8 Verifiable Credential	4.2 Operational Roles of Digital Identities and Keypair Usage
3.5.9 Enveloped Verifiable Credential	4.2.1 Keypair not bound to a Certificate
3.5.10 Verifiable Presentation	4.2.2 Self-Issued Keypair
3.5.11 Enveloped Verifiable Presentation	4.2.3 Trust Service Provider ( <u>TSP</u> ) Keypair
3.6 Gaia-X Compliance input/output	4.3 Binding Digital Identities to Claims
3.6.1 Input	4.3.1 Different Requirement based on Use cases
3.6.2 Output	4.4 DID Resolution
4.2.2 Policy description	4.4.1 Verification Method
4. TrustAnchor Credential	4.5 <u>eIDAS</u> Integration
5. Party Credential	4.5.1 eID
5.1 Private Party Credential	4.5.2 eSignature
5.1.1 Private Party Credential Example	4.6 Implementing Interactions between Machines and Humans
5.2 Public Party Credential	4.6.1 Interactions with Human-in-the-Loop

Previous version ToC	Current New ToC
6. Party Credential Lifecycle	4.6.2 Interactions with Machines
7. Party Credential Status	5. Gaia-X Credentials
8. OpenID Connect for Verifiable Credentials	5.1 Overview
8.1 OpenID Connect for Verifiable Issuance	5.2 Core Data Model Foundations
8.2 OpenID Connect for Verifiable Presentations	5.2.1 Namespace Bindings and Contexts
8.3 Usage	5.2.2 Usage
8.4.Cloud/Enterprise Wallet	5.2.3 Type Property
9. Signature Credential	5.3 Credential Format Specification
9.1 Multiple Signatures using SignatureCredential specializations	5.3.1 Encoding Requirements
9.1.1 SignatureAgreementCredential	5.3.2 Credential Structure
9.1.2 Data Usage Agreement Example	5.3.3 Credential Subject
9.2 Multiple Signatures using Proof Set and Proof Chain	5.4 Verifiable Credentials
10. Trustframework Implementation	5.4.1 Standard Verifiable Credential
10.1 Trust Framework Implementation	5.4.2 Enveloped Verifiable Credential
10.2 Trust Anchor Credential specialization examples	5.5 Verifiable Presentations
10.3 Party Credential Specialization examples	5.5.1 Standard Verifiable Presentation
10.3.1 Natural Person Party Credential	5.5.2 Enveloped Verifiable Presentation
10.3.2 Legal Person Party Credential	5.6 Issuer Requirements
10.3.3 Service Part Credential	5.7 Additional Features
10.3.4 Membership Party Credential	5.7.1 Integrity of Related Resources
10.4 Access rights delegation example - Employee Authentication	5.7.1 Credential Lifecycle Status
10.4.1 Problem Statement	6. ICAM Semantic Model
10.4.2 Types of Credentials and Issuers involved	6.1 Trust Scope Credential
10.4.3 Access Entitlement Credentials	6.1.1 Trust Scope Credential specialisation examples
10.4.4 Authentication Process	6.1.2 Federation using Trust Scope Credentials
10.4.5 Revocation	6.2 Party Credential
10.4.6 Implementation Considerations	6.2.1 Private Party Credential
11. Changelog	6.2.2 Public Party Credential
	6.2.3 Party Credential Specialisation examples
	6.3 Signature Credential
	6.4 Ecosystem Onboarding and Offboarding using ICAM Semantic Model
	6.5 Delegating Access Rights
	6.5.1 Types of Credentials and Issuers
	6.5.2 Implementation Factors
	7. Changelog

## 7.2 2024 July release (24.07)

---

- Updated chapter "Credential Format"
- New chapter "Trust Anchor Credential and Party Credential"
- New chapter "OpenID Connect for Verifiable Credentials"
- New chapter "Signature Credential"
- New chapter "Trust Framework implementation", containing Trust Anchor Credential specialisation examples and Party Credential specialisation examples and an access rights delegation example.