

Software Requirements Specification

for

**Organization Credential Manager
IDM.OCM**

Published by

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

Copyright

© 2021 GAIA-X European Association for Data and Cloud AISBL

This work is licensed under the Creative Commons Attribution 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by/4.0/> or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA



Table of Contents

List of Figures	vi
List of Tables	vi
1. Introduction	1
1.1. Document Purpose	1
1.2. Product Scope	1
1.3. Definitions, Acronyms and Abbreviations	1
1.4. References	2
1.5. Document Overview	5
2. Product Overview	5
2.1. Product Perspective	5
2.2. Product Functions	7
2.3. Product Constraints	9
2.4. User Classes and Characteristics	10
2.5. Operating Environment	11
2.6. User Documentation	11
2.7. Assumptions and Dependencies	12
2.8. Apportioning of Requirements	12
3. Requirements	12
3.1. External Interfaces	12
3.1.1. User Interfaces	12
3.1.1.1. General User Interface Requirements	12
3.1.1.2. Principal Manager	13
3.1.2. Hardware Interfaces	13
3.1.3. Software Interfaces	13
3.1.3.1. General	13
3.1.4. Communications Interfaces	13
3.1.4.1. General	14
3.1.4.2. SSI Abstraction Service	14
3.1.4.3. Connection Manager	14
3.1.4.4. Attestation Manager	15
3.1.4.4.1. Issue Credential	15
3.1.4.4.2. Schemas	17
3.1.4.5. Profile Manager	17

3.1.4.6. Proof Manager	18
3.1.4.7. Principal Manager	19
3.2. Functional	19
3.2.1. General	19
3.2.2. Connection Manager	21
3.2.3. Proof Manager	25
3.2.4. Attestation Manager	28
3.2.4.1. Revocation	28
3.2.4.2. Trusted Connection	28
3.2.4.3. Schema & Credential	29
3.2.5. Principal Manager	30
3.2.6. SSI Abstraction Service	31
3.2.7. Profile Manager	32
3.2.7.1. Service-Offering	33
3.3. Other Nonfunctional Requirements	34
3.3.1. HTTP Requirements	34
3.3.2. Configuration	34
3.3.3. Logging Requirements	34
3.3.4. Monitoring Requirements	35
3.3.5. Performance Requirements	35
3.3.6. Safety Requirements	35
3.3.7. Security Requirements	36
3.3.7.1. General Security Requirements	36
3.3.7.2. Service Specific Security Requirements	36
3.3.8. Software Quality Attributes	38
3.4. Compliance	39
3.5. Design and Implementation	39
3.5.1. Distribution	39
3.5.2. Maintainability	39
3.5.3. Operability	39
3.5.4. Interoperability	40
3.5.5. Scalability	40
4. System Features	41
4.1. Integration Overview	41
4.1.2. Description	41

4.2. Principal Manager	42
4.2.1. Description and Priority	42
4.2.2. Functional Requirements	44
4.3. Connection Manager	44
4.3.1. Description and Priority	44
4.3.2. Stimulus/Response Sequences	44
4.3.3. Functional Requirements	45
4.4. Proof Manager	46
4.4.1. Description and Priority	46
4.4.2. Functional Requirements	46
4.5. Attestation Manager	47
4.5.1. Description and Priority	47
4.5.2. Functional Requirements	47
4.6. SSI Abstraction Service	48
4.6.1. Description and Priority	48
4.6.2. Functional Requirements	48
4.7. Profile Manager	48
4.7.1. Description and Priority	48
4.7.2. Functional Requirements	49
5. Other Requirements	49
6. Verification	50
Appendix A: Glossary	51
Appendix B: Overview GXFS Work Packages	51

List of Figures

Figure 1: Architecture	6
Figure 2: Product Components Overview	8
Figure 3: High Level Interaction View for Gaia-X Service-Offering and Service-Consumption Process	41
Figure 4: Issue Employee Credential	43
Figure 5: Connection Manager example flows	45

List of Tables

Table 1: References	4
Table 2: User Classes and Characteristics	10
Table 3: Apportioning of Requirements	12
Table 4: Requirements on cryptographic algorithms and key length	37
Table 5: Functional Requirements Principal Manager	44
Table 6: Functional Requirements Connection Manager	46
Table 7: Functional Requirements Proof Manager	46
Table 8: Functional Requirements Attestation Manager	48
Table 9: Functional Requirements SSI Abstraction Service	48
Table 10: Functional Requirements Profile Manager	49

1. Introduction

To get general information regarding Gaia-X and the Gaia-X Federation Services please refer to [\[TAD\]](#) and [\[PRD\]](#).

1.1. Document Purpose

The purpose of the document is to specify the requirements of the Identity Management and Trust Subcomponent “Organization Credential Manager” with the intention of a European wide public tender for implementing this software. 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 these products is to provide all necessary components for the administration of the digital identity of a participant in the Gaia-X context. The Organization Credential Manager (OCM) enables a participant to interact with the SSI-based ecosystem in a trustful and secure fashion. This comprises the utilization of the participants digital identity for different core functionalities:

- Establishment of secure and trustable connections with other parties
- Request and reception of verifiable credentials from attesting parties (e.g., Gaia-X Membership credential from a verified notary)
- Attestation of attributes to principals in the form of verifiable credentials (e.g., employees, technical assets)
- Validation of received verifiable presentations from other parties (e.g., validation of Gaia-X membership of other participants)
- Maintain verifiable Public Profile

The described functionalities allow other components in the Identity Management context to interact with the SSI-based ecosystem. The OCM interacts with the Trust-Service to allow policy enforcement by being the key point for trustful information through verifiable presentations. It can be used by different roles, such as principals and participants, to support their respective processes in terms of digital trust.

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

[Aries.RFC0036]	(Community) (2021), Aries RFC 0036: Issue Credential Protocol 1.0
	https://github.com/hyperledger/aries-rfcs/blob/master/features/0036-issue-credential/README.md (Status: 03-17-2021)
[Aries.Cloud.Agent]	(Community) (2021), Aries Cloud Agent Internals: Agent and Controller
	https://github.com/hyperledger/aries-cloudagent-python/blob/main/docs/GettingStartedAriesDev/AriesAgentArchitecture.md (Status: 03-17-2021)
[Aries RFCs]	(Community)(2021), Hyperledger Aries RFCs
	https://github.com/hyperledger/aries-rfcs (Status: 03-17-2021)
[Aries.RFC0037]	(Community) (2020), Aries RFC 0037: Present Proof Protocol 1.0 - Request Presentation
	https://github.com/hyperledger/aries-rfcs/tree/master/features/0037-present-proof#request-presentation (Status: 03-17-2021)
	Ryan West, Daniel Bluhm, Matthew Hailstone, Stephen Curran, Sam Curren (2019), Connection Protocol
	https://github.com/hyperledger/aries-rfcs/tree/9b0aaa39df7e8bd434126c4b33c097aae78d65bf/features/0160-connection-protocol (Status: 03-17-2021)
[ACA-Py]	Hyperledger Aries (2021), Hyperledger Aries Cloud Agent Python (ACA-Py)
	https://github.com/hyperledger/aries-cloudagent-python (Status 03-29-2021)
[Aries.BPA]	Hyperledger Aries (2021), Business Partner Agent
	https://github.com/hyperledger-labs/business-partner-agent (Status 03-29-2021)
[Bosch]	(Bosch) (2021), Bosch Verifier Agent
	https://boschid-agent-admin.dev.economyofthings.io/api/doc#/connection/post_connections_conn_id_accept_request (Status: 03-17-2021)
[BDD]	Specflow (n.D.), Getting Started with Behavior Driven Development
	https://specflow.org/bdd/ (Status 03-18-2021)
[CryptoLen]	Damien Giry, Prof. Jean-Jacques Quisquater (2020), Cryptographic Key Length Recommendation
	https://www.keylength.com/en (Status 03-18-2021)
[CloudEvents]	CloudEvents Authors, The Linux Foundation (2021), CloudEvents Specification

	https://cloudevents.io/ (Status: 03-27-2021)
[DID]	W3C (2021), Decentralized Identifiers (DIDs) v1.0
	https://www.w3.org/TR/2021/CR-did-core-20210318/ (Status 03-18-2021)
[ESSIF]	European Commission (n.d.), European Self-Sovereign Identity Framework (ESSIF)
	https://ec.europa.eu/cefdigital/wiki/pages/viewpage.action?pageId=262505734 (Status: 02-18-2021)
	N. Sakimura, J. Bradley, M. Jones, B.de Medeiros, C. Mortimore (2014), OpenID Connect Core 1.0 incorporating errata set 1
[EUCS]	European Union Agency for Cybersecurity (ENISA) (2020), EUCS – Cloud Services Scheme
	https://www.enisa.europa.eu/publications/eucs-cloud-service-scheme (Status: 03-29-2021)
[HyperLedger.Indy]	Ryan West, Daniel Bluhm, Matthew Hailstone, Stephen Curran, Sam Curren (2018), Hyperledger Indy HIPE - Invitation to Connect
	https://hyperledger-indy.readthedocs.io/projects/hipe/en/latest/text/0031-connection-protocol/README.html#invitation-to-connect (Status: 03-17-2021)
[Issue.Cred.Prot]	Nikita Khateev (2019), Aries RFC 0036: Issue Credential Protocol 1.0
	https://github.com/hyperledger/aries-rfcs/tree/master/features/0036-issue-credential (Status: 02-22-2021)
[IDM.AO]	Gaia-X WP1¹ (2021), Architecture Overview
	Please refer to annex “GX_IDM_AO”
[ISO25000]	ISO 25000 Portal (n.d.), ISO/IEC 25010
	https://iso25000.com/index.php/en/iso-25000-standards/iso-25010 (Status: 03-17-2021)
[NF.SPBD]	Gaia-X Federation Service Non-functional Requirements Security & Privacy by Design
	Please refer to annex “GXFS_Nonfunctional_Requirements_SPBD”
[OIDC.Core]	https://openid.net/specs/openid-connect-core-1.0.html#UserInfo (Status: 03-17-2021)
[OIDC.Conformance]	OpenID Connect Working Group, OpenID Foundation (2018), OpenID Connect Conformance Profiles v3.0
	https://openid.net/wordpress-content/uploads/2018/06/OpenID-Connect-Conformance-Profiles.pdf (Status: 03-11-2021)

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

[PresentProof]	Nikita Khateev (2019), Aries RFC 0037: Present Proof Protocol 1.0
	https://github.com/hyperledger/aries-rfcs/tree/master/features/0037-present-proof (Status: 02-22-2021)
[PRD]	Gaia-X, European Association for Data and Cloud, AISBL (2021): Gaia-X Policy Rules Document
	Please refer to annex “Gaia-X_Policy Rules_Document_2104”
[RFC3161]	C.Adams, P.Cain, D.Pinkas, R.Zuccherato (2001), Internet X.509 Public Key Infrastructure, Time-Stamp Protocol (TSP)
	https://www.ietf.org/rfc/rfc3161.txt (Status: 03-17-2021)
[SOG-IS]	SOG-IS Crypto Working Group (2020), SOG-IS Crypto Evaluation Scheme - Agreed Cryptographic Mechanisms
	https://www.sogis.eu/documents/cc/crypto/SOGIS-Agreed-Cryptographic-Mechanisms-1.2.pdf (Status 03-18-2021)
[TR02102-1]	BSI (2020), Cryptographic Mechanisms: Recommendations and Key Lengths BSI TR-02102-1
	https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TG02102/BSI-TR-02102-1.pdf?__blob=publicationFile&v=2 (Status 03-18-2021)
[TR02102-2]	BSI (2020), Cryptographic Mechanisms: Recommendations and Key Lengths: Use of Transport Layer Security (TLS) BSI TR-02102-2,
	https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TG02102/BSI-TR-02102-2.pdf?__blob=publicationFile&v=2 (Status 03-18-2021)
[TDR]	Gaia-X Federation Services Technical Development Requirements
	Please refer to annex “GXFS_Technical_Development_Requirements”
[TAD]	Gaia-X, European Association for Data and Cloud, AISBL (2021): Gaia-X Architecture Document
	Please refer to annex “Gaia-X_Architecture_Document_2103”
[VC.DataModel]	W3C (2019), Verifiable Credentials Data Model 1.0
	https://www.w3.org/TR/vc-data-model Status (02-23-2021)
[VC.DataModel]	W3C (2019), Verifiable Credentials Data Model 1.0
	https://www.w3.org/TR/vc-data-model/ (Status: 02-18-2021)

Table 1: References

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.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

The product is necessary to establish trust between the different participants within the Gaia-X ecosystem and to create a level of trust using a decentralized approach.

The OCM fulfills parts of the functionality that an identity provider provides for centralized or federated identity approaches but in a decentralized fashion using concepts of decentralized identity, verifiable credentials, and verifiable presentations.

To achieve this goal, components are required that on the one hand allow the management of a participant identity for the creation of signatures for various properties, attributes and documents, and on the other hand enable the verification of external documents. This includes the creation of verifiable credentials with a corresponding digital signature based on an identity, the issuing of verifiable presentations based on existing and already received verifiable credentials, the requesting of verifiable credentials from third parties for the attestation of own attributes, for example, as well as the validation of incoming connection requests and proof requests. The format used for communication is based on the RFCs² described in the Hyperledger Indy context and the standards of W3C^{3 4} to guarantee a uniform process flow and exchange formats.

The OCM must support the following Gaia-X processes:

- Participant onboarding [IDM.AO, Section “Participant Onboarding”]
- Principal onboarding [IDM.AO, Section “Principal Onboarding”]
- Offboarding [IDM.AO, Section “Offboarding”]
- Authentication [IDM.AO, Section “Authentication”]
- Trust Establishment [IDM.AO, Section “Trust Establishment”]

Compliance-specific processes and business flows are not part of the specification or out of scope.

² <https://github.com/hyperledger/aries-rfcs>

³ <https://www.w3.org/TR/did-core/>

⁴ <https://www.w3.org/TR/vc-data-model/>

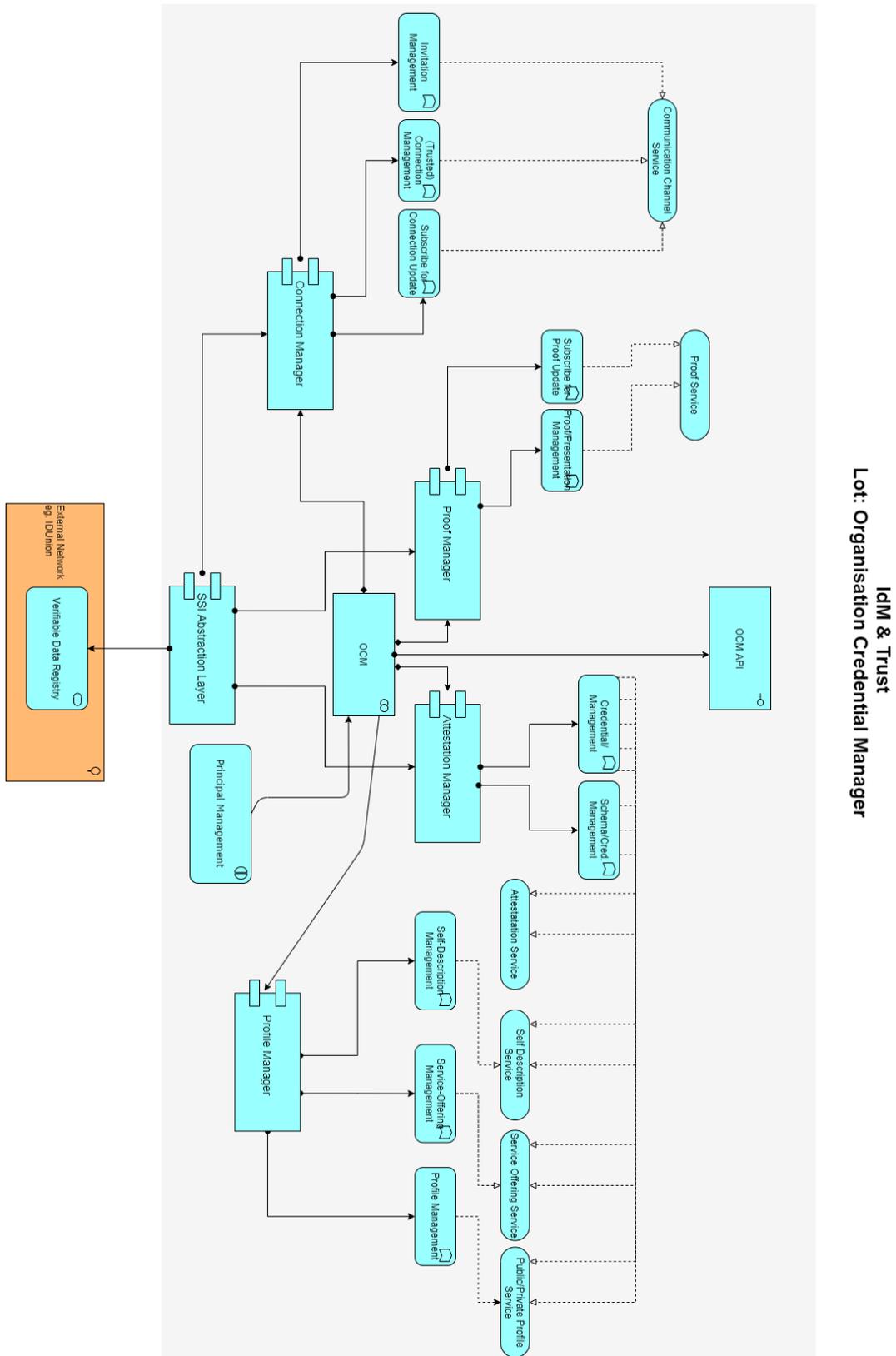


Figure 1: Architecture

The Hyperledger Indy Identity ecosystem has done a great job to build frameworks and components to support specific project implementations. The most relevant components for the goal of the OCM are: Hyperledger Aries Cloud Agent [[ACA-Py](#)] and the Hyperledger Labs' Business Partner Agent [[Aries.BPA](#)].

Both are great sources for inspiration for the following specification and can be used as a reference in case details are missing on a technical level.

2.2. Product Functions

The functions of the Organization Credential Manager (OCM) component are provided as a runtime component and MUST expose endpoints as REST services and made accessible over the network using encrypted connections (e.g., HTTPS). The scalability of these services MUST be taken into consideration using well-known and tested concepts like a microservice based architecture and load balancing. Since this component is the very core of trust relationships between participants in the Gaia-X ecosystem, security measures MUST be in place accordingly. This includes the protection of exposed service endpoints, data storage protection and access control. The storage for cryptographic material MUST be particularly secured, e.g., by integrating Hardware Security Modules. The overall functionality of the OCM component and exposed services MUST be auditable (in compliance with GDPR).

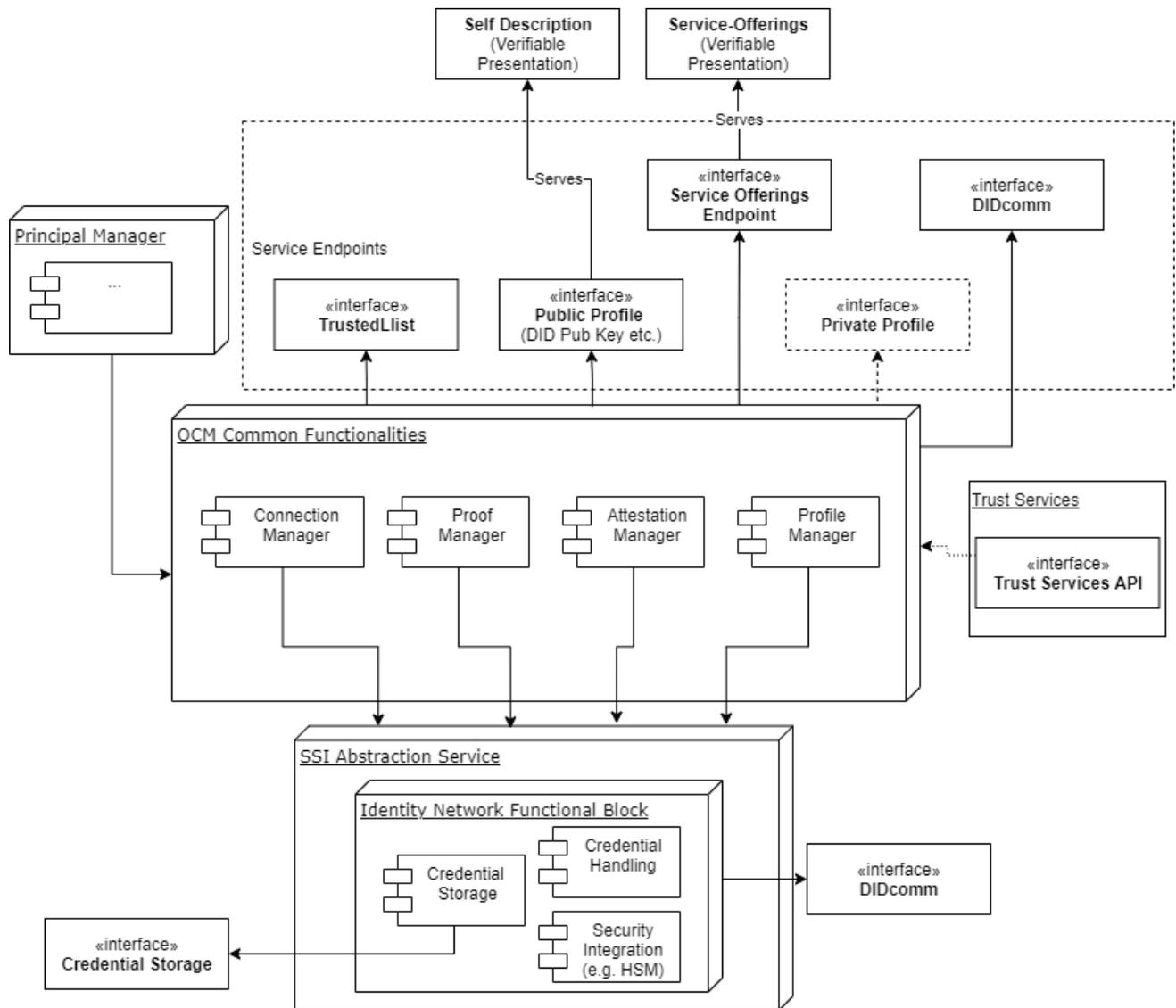


Figure 2: Product Components Overview

The core functions of the OCM are:

- Establishing trusted connections between entities (Connections in this context are private, secured, and persistent channels between entities)
 - Creation of Connection invitations
 - Handling of incoming connection invitations
 - Managing of existing connections
 - Association of a connection and proven attributes, allowing for a connection that is trusted according to the scope of Gaia-X
- Handling of verifiable credentials exchanges
 - Issuing of credentials to participants, principals, assets etc.
 - Handling of incoming Credential proposals
- Handling of verifiable proof exchanges

- Requesting and verifying proofs from other entities (e.g., participants) in the ecosystem
- Handling of incoming proof requests
- Secure storage of credentials
- Checking the validity of proof presentations (iteration 1: expiration time, iteration 2: revocation mechanism)
- Providing publicly visible and verifiable service endpoints
 - Public Profile: Company information that is made publicly available (e.g., imprint). One aspect that has to be served is the self-description according to Gaia-X, other aspects can be served according to the trust policies of the Participant
 - Private Profile: Company information that is only visible after proving certain aspects (e.g., Gaia-X membership)
 - Service-Offering: A list of services that the Participant wants to offer in Gaia-X
 - TrustedList: A list of participants that are trusted by this instance of OCM. Only relevant for AISBL in the first iteration (list of trusted notarization DIDs)

The SSI Abstraction Service provides the required SSI functionality to the other components but is not aware of the Gaia-X context. The context-specific aspects and interaction with Trust Services are implemented in the specific components:

- Connection Manager
- Proof Manager
- Attestation Manager

2.3. Product Constraints



IDM.OCM.00001 **The document IDM.AO is the common basis for this functional specification**

The architecture document "IDM.AO" [\[IDM.AO\]](#) is an essential part of this specification and a prerequisite for understanding the context. The specifications and requirements from the Architecture Document MUST be considered during implementation.



IDM.OCM.00002 **Micro Service Architecture**

For a better scale out and decentralization, the product architecture MUST be a micro service architecture. The modules MUST NOT be tightly integrated into the IAM solution, as Plugin or Extensions, rather should interact with the said system through standard APIs and Protocols.



IDM.OCM.00003 **Internal IAM**

An internal IAM for user accounts/login management is REQUIRED and MUST provide attributes related to the user in a technical and automatable way. But all interfaces to the internal IAM interfaces SHOULD support already existing IAMs. Therefore, the internal IAM interfaces SHOULD be set up on standard OIDC implementations.

2.4. User Classes and Characteristics

<i>User Class</i>	<i>Description</i>	<i>Frequency</i>	<i>Expertise</i>	<i>Privilege Level</i>	<i>Product Usage</i>
Administrator	Setup, organize and monitor the system. Integration into the company systems and networks wherever necessary	Low	High	High	Maintenance
Principal	Principals receive credentials to proof affiliation to the organization	Low	Low	Low	Trusted Connections and Information Exchange
Trust Services	Controls the usage of most OCM functions via policies	High	High	High	Administration of trusted connections and information exchange
Organization Internal Systems	Internal systems provide and consume data to/from the OCM	High	High	Low	Information / Trust Sources
External Participants / Principals	Provides data or requests Proofs	High	High	Low	Trusted Connections and Information Exchange

Table 2: User Classes and Characteristics

2.5. Operating Environment

Please refer to [\[TDR\]](#) for further binding requirements regarding the operating environment.



IDM.OCM.00004 **TLS Protected Endpoints**

To protect the product endpoint(s), it's necessary to support a network infrastructure e.g., load balancers/proxies which **MUST** support TLS encryption. The encryption **MUST** meet the requirements listed in the chapter for security requirements.



IDM.OCM.00005 **NTP Server**

The product **MUST** be operated within an environment connected to a trusted stratum level 2/3 NTP Server.



IDM.OCM.00006 **Secure Timestamps**

All timestamps **MUST** be issued according to RFC 3161⁵.

2.6. User Documentation

Please refer to [\[TDR\]](#) for further requirements regarding documentation.



IDM.OCM.00007 **Participant Administration Documentation**

The documentation **MUST** contain:

- Installation Manuals
- Cryptographic Initialization (if applicable)
- Description of Deployment
- Description of the Automatic Tests / Verification
- How to build the products from source code



IDM.OCM.00008 **Participant Documentation**

The documentation **MUST** contain:

- Short Software Description/Usage
- Usage Guide
- GDPR Design Decisions
- Security Concept
- Operations Concept
- FAQ
- Keyword Directory

⁵ <https://tools.ietf.org/html/rfc3161>

2.7. Assumptions and Dependencies

An understanding of the overall Gaia-X architecture and philosophy is necessary. Please refer to [\[TAD\]](#) and [\[PRD\]](#).

2.8. Apportioning of Requirements

<i>Feature</i>	<i>Priority</i>
<i>Connection Manager</i>	<i>1</i>
<i>Proof Manager</i>	<i>1</i>
<i>Attestation Manager</i>	<i>1</i>
• <i>Revocation Mechanism</i>	<i>2</i>
<i>Principal Manager</i>	<i>1</i>
<i>Profile Manager</i>	<i>1</i>
<i>TrustedList</i>	<i>1</i>
<i>SSI Abstraction Service</i>	<i>1</i>
<i>eIDAS Compliant Credentials</i>	<i>3</i>
<i>Private Profile</i>	<i>2</i>
<i>Service Instance Attestations</i>	<i>3</i>

Table 3: Apportioning of Requirements

3. Requirements

Further binding requirements can be found in [\[TDR\]](#).

3.1. External Interfaces

3.1.1. User Interfaces

3.1.1.1. General User Interface Requirements



IDM.OCM.00009 **Language and Web-Accessibility Support**

User Interfaces must support the main user agents (browsers), Chrome, Firefox, Safari, Edge.

User Interfaces must support multiple languages.

Iteration 1: English (EN-en), German (DE-de), French (FR-fr)

The frontend must support to manually switch between languages.

The frontend and backend may automatically detect the language from the user agent (browser).
Default: English if any not supported language was detected.

User Interfaces must support “web accessibility” / legal conformity. ☐

3.1.1.2. Principal Manager

☐ IDM.OCM.00010 **Principal Manager Frontend**

Requires ☐ [IDM.OCM.00009 Language and Web-Accessibility Support](#)

User Interface Iteration 1:

- Dialog / Action: Page to start the Authentication Flow to login with the internal IdM and show the QR Code with the invitation_url to the to the User (Principal)

User Interface Iteration 2:

- Theming: User Interface is customizable with a major template language (e.g., mustache)
- User Interface allows to request additional Verifiable Credentials
 - Issuing based on policy result with input from internal IdM (e.g., IdM roles) ☐

3.1.2. Hardware Interfaces

☐ IDM.OCM.00011 **Hardware Encryption**

An option to securely create, store and access cryptographic material MUST be provided (e.g., HSM, Vault, Keymanager). ☐

3.1.3. Software Interfaces

3.1.3.1. General

☐ IDM.OCM.00012 **General Operation Requirements**

Every component must be able to run as a container. For scalable deployment e.g., a helm chart SHOULD be provided.

If database connections are used, it must provide options to run the container “stand-alone”, e.g., in-memory and with an external, configurable database. ☐

☐ IDM.OCM.00013 **External Schemas**

All schemas MUST be defined in the predefined identity network. External networks with separate schemas are out of scope for the first implementation release. Future releases MAY support a schema sync or credential issue process based on external schemas. ☐

3.1.4. Communications Interfaces

3.1.4.1. General



IDM.OCM.00014 **Event Handling**

A lot of services within the OCM are publishing and receiving events, mostly events in the scope of SSI for the different protocol flows. The way to publish and subscribe these events **MUST** be consistent throughout the functionality of the OCM.

In general, this functionality **MUST** consider the cloudevents specification for this event system [\[CloudEvents\]](#).

3.1.4.2. SSI Abstraction Service



IDM.OCM.00015 **DIDComm Endpoint (externally exposed)**

The SSI Abstraction Service **MUST** cover all didcom protocol interactions for: basicmessage, connection, credential-definition, credentials, issue-credential, ledger, out-of-band, present-proof, revocation, schema, wallet. Reference implementation is done within hyperledger with aries-cloudagent-python which offers for all needed interfaces REST endpoints that provide didcom functionality for other components. This also includes the possibility to check all available connections, credentials, schemas, credential definitions and proofs.⁶



IDM.OCM.00016 **SSI Event Handling**

The SSI Abstraction Service **MUST** provide a way for all components to forward specific types of events to the corresponding service. Connection events **MUST** be forwarded to the Connection Manager, events in the process of issuing credentials **MUST** be forwarded to the Attestation Manager and events for proof presentation **MUST** be forwarded to the Proof Manager. These services **MUST** provide corresponding endpoints to receive events. This routing of events **SHOULD** be provided via REST services, but message bus systems can also be considered.



IDM.OCM.00017 **SSI Admin Interface**

The SSI Abstraction Service **MUST** provide an API interface to control all necessary SSI functionality and provide this to the Connection Manager, Attestation Manager and Proof Manager. This endpoint **MUST** be secured.

3.1.4.3. Connection Manager



IDM.OCM.00018 **Create Invitation Endpoint**

The Connection Manager **MUST** provide a Create Invitation endpoint. The response body is composed of JSON content. The response includes the created invitation object from the SSI Abstraction Service

The endpoint supports two kind of arguments

- Alias (STRING): attribute provides a suggested label for the connection. This allows the user to tell multiple connection offers apart. This is not a trusted attribute.

⁶<https://github.com/hyperledger/aries-cloudagent-python/blob/main/docs/GettingStartedAriesDev/AriesAgentArchitecture.md>

- public (BOOLEAN): Create invitation from public DID ☐

☐ **IDM.OCM.00019 Connection Information Endpoint**

The Connection Manager MUST provide a connection information endpoint where other components can request if a connection for a specific DID already exists. The endpoint supports as input a ConnectionID or a DID as well as a flag for “all-information” which also covers all received presentations and issued credentials and responds with existing information (connection status, ConnectionID, if requested additional information) or nothing if it does not exist. ☐

☐ **IDM.OCM.00020 Connection Status Subscribe Endpoint**

The Connection Manager MUST provide an endpoint that allows a component to subscribe/unsubscribe for connection events to receive updates when the status changes. A list of subscribers MUST be in place and maintained based on the respective implementation. (subscribe, unsubscribe MUST be possible). This functionality of publishing events SHOULD be provided via a REST service, but a message bus system can also be considered. ☐

3.1.4.4. Attestation Manager

☐ **IDM.OCM.00021 Create DID**

The Attestation Manager MUST provide an Endpoint that allows the creation of DID for a participant. In a future implementation release DIDs MUST also be generated for Assets. The Attestation Manager MUST call the SSI Abstraction Service to generate a DID with required information (e.g., verkey for Hyperledger Indy network). The DID is used for the onboarding process at the AISBL.

Interface:

[SSI Abstraction Service] DID Create ☐

3.1.4.4.1. Issue Credential

☐ **IDM.OCM.00022 Credential Issue Endpoint**

The Attestation Manager MUST provide an Endpoint to allow the request for issuing a credential for a specific subject DID from another component. The endpoint expects the following attributes as payload

- Subject DID
- Schema
- Claims for Credential
- Additional Information and mandatory fields are defined in [IDM.AO Chap-4.7] (Expiration date of credential, ...)

Constraints:

A connection for the provided subject DID MUST be established before.

Issuance of credentials to external subject DIDs require a “trusted” state of the connection Credential-Definition existing

Supported Actions: POST with payload (JSON)

Response:

- If the credential was issued: An appropriate response to the HTTP action like 200
- If the credential was not issued: An appropriate response to the HTTP with the respective error ☐

☐ **IDM.OCM.00023 Credential-Proposal Endpoint**

The Attestation Manager **MUST** provide an endpoint to create a credential proposal that will be sent to an issuing party in the Gaia-X ecosystem. An existing connection **MUST** be in place and all necessary information **MUST** be provided.

Constraint:

Existing connection

Supported Actions: POST (Credential Proposal) ☐

☐ **IDM.OCM.00024 Issue Credential Status Endpoint**

The Attestation Manager **MUST** provide an endpoint to receive updates for an ongoing issue credential process. The endpoint supports a JSON based body.

Constraints:

A credential issue process has been already started ☐

Supported Actions: POST(CredentialExchange Attributes as JSON payload (connection_id, credential_exchange_id, created_at, credential_id, state)) ☐

☐ **IDM.OCM.00025 Credential Status Subscribe Endpoint**

The Attestation Manager **MUST** provide a Credential Status Endpoint that allows a component to subscribe/unsubscribe for Credential event updates when a credential status has changed (valid options are request-credential, propose-credential, receive credential ack). A list of subscribers for a specific ConnectionID **MUST** be in place and maintained based on the respective implementation. (subscribe, unsubscribe **MUST** be possible). This functionality of publishing events **SHOULD** be provided via a REST service, but a message bus system can also be considered. ☐

☐ **IDM.OCM.00026 Get Credentials**

The Attestation Manager **MUST** provide an endpoint to request existing credentials inside the wallet of the OCM. This request **MUST** support a simple query language or offer query/path parameters to query for credentials based on schemas, issuer DIDs, attribute names, attribute values, or an internal CredentialID. ☐

3.1.4.4.2. Schemas

- ▶▶ IDM.OCM.00027 **Create Schema Endpoint**
The Attestation Manager MUST provide a create schema endpoint that expects a JSON based object with one or more schema-definitions containing for each entry attributes, schema_name and schema_version. ◀◀

- ▶▶ IDM.OCM.00028 **Update Schema Endpoint**
The Attestation Manager MUST provide an update schema endpoint that expects a JSON based object with one or more schema-definitions containing for each entry attributes, schema_name and schema_version. ◀◀

- ▶▶ IDM.OCM.00029 **Get DIDs for Schema Endpoint**
The Attestation Manager MUST provide an endpoint that allows to request a list of DIDs that are trusted members. The request expects a schema as parameter and delivers a list of DIDs that have received credentials based on the provided schema. One example for that endpoint is the request for AISBL to get a list of all “trusted” members by passing the Participant Credential schema as parameter. ◀◀

3.1.4.5. Profile Manager

- ▶▶ IDM.OCM.00030 **Create Public Profile Endpoint**
Creates and publishes the Public Profile Endpoint in the DID Document via SSI Abstraction Service ◀◀

- ▶▶ IDM.OCM.00031 **Update Public Profile Endpoint**
Updates the Public Profile Endpoint in the DID Document via SSI Abstraction Service ◀◀

- ▶▶ IDM.OCM.00032 **Delete Public Profile Endpoint**
Deletes the Public Profile Endpoint from the DID document via SSI Abstraction Service ◀◀

- ▶▶ IDM.OCM.00033 **Update Self-Description**
Triggers ▶▶ [IDM.OCM.00096 Self Description Output](#) and serves via Public Profile ◀◀

- ▶▶ IDM.OCM.00034 **Serve Public Profile**
Serves the Self Description from ▶▶ [IDM.OCM.00096 Self Description Output](#) via the configured and published (▶▶ [IDM.OCM.00030 Create Public Profile Endpoint](#)) endpoint. ◀◀

- ▶▶ IDM.OCM.00035 **Unserve Public Profile**
Unserve / delete the Public Profile Content (Self-Description) from the Public Profile endpoint ◀◀

- ▶▶ IDM.OCM.00036 **CRUD Service-Offering Endpoints**

- Provide CRUD endpoints for [IDM.OCM.00097 Service-Offering CRUD](#)
- ▶▶ IDM.OCM.00037 **Publish Service-Offering Endpoint**
 Uses [IDM.OCM.00098 Service Offering Export](#) to generate and publish the Service Offering under the Service Offering Endpoint (from DID document endpoints) http resource. ◀◀
- ▶▶ IDM.OCM.00038 **Un-Publish Service-Offering Endpoint**
 Deletes the http resource generated in [IDM.OCM.00037 Publish Service-Offering Endpoint](#) and removes the Service-Offering endpoint from the DID document ◀◀

3.1.4.6. Proof Manager

- ▶▶ IDM.OCM.00039 **Presentation Request**
 The Proof Manager MUST provide an endpoint to request a presentation over an existing connection. If the holder DID for the request is a Participant and no connection exists, a trusted connection MUST be established beforehand. The relevant payload or attributes for the request as well as the connection or a DID for the request MUST be provided.

Interface

[SSI Abstraction Service] Presentation Request endpoint ◀◀

- ▶▶ IDM.OCM.00040 **Presentation Status Subscribe Endpoint**
 The Proof Manager MUST provide a Presentation Status Endpoint that allows a component to subscribe/unsubscribe for Presentation event updates when new Presentations are received or the status changes. A list of subscribers for a specific ConnectionID MUST be in place and maintained based on the respective implementation (subscribe, unsubscribe MUST be possible). This functionality of publishing events SHOULD be provided via a REST service, but a message bus system can also be considered. ◀◀

- ▶▶ IDM.OCM.00041 **Verify JSON-LD Presentation Endpoint**
 The endpoint input is a JSON-LD Verifiable Presentation. It provides 2 options:

First, to validate the JSON-LD structure, including the used context information and the overall “proof”, which is the signature of the Holder.

Second, to additionally verify all the individual Verifiable Credential contained in the Verifiable Presentation, including [IDM.OCM.00042 Verify Indy CredentialDefinition ProofType Endpoint](#) ◀◀

- ▶▶ IDM.OCM.00042 **Verify Indy Credential Definition Proof Type Endpoint**
 Input: A JSON-LD Verifiable Credential with an Indy specific Proof Type.
 Uses [IDM.OCM.00094 CredDefProofType](#) to verify the claims and returns

true / false and a reference ID to the exchanged Proof. ◀◀

3.1.4.7. Principal Manager



IDM.OCM.00043 **Principal Manager Trust Service Interaction**

In addition to the interaction with the internal IdM, the component needs to interact with the Trust Service component, more specifically the Policy Decision Engine (PDE).

The PDE provides a REST style interface. It provides APIs to query whether a specific user is allowed to receive the *PrincipalCredential* or not.

Request: User information data and Policy name *PrincipalCredentialRequest*

Response: It returns a true/false response.

With a positive response (true) from the PDE, the Attestation Manager is used to issue the Verifiable Credential. ◀◀



IDM.OCM.00044 **Principal Manager Connection Manager Interaction**

The component uses the Connection Manager to connect to the Principal identity (Personal Credential Manager). ◀◀

The component requests an invitation link from the Connection Manager API [▶](#) [IDM.OCM.00063 Create Invitation](#). The Frontend MUST show this to the Principal (user).

The Connection Manager provides a REST style API with JSON.

Request: -

Response: JSON "invitation_url" ◀◀



IDM.OCM.00045 **Principal Manager Attestation Manager Interaction**

The component uses the Attestation Manager to issue a Verifiable Credential to the Principal requesting it [▶](#) [IDM.OCM.00022 Credential Issue Endpoint](#)

The Attestation Manager provides a REST style API with JSON.

Issuing:

Request: as described in [▶](#) [IDM.OCM.00022 Credential Issue Endpoint](#)

Response: as described in [▶](#) [IDM.OCM.00022 Credential Issue Endpoint](#) ◀◀

3.2. Functional

3.2.1. General



IDM.OCM.00046 **Service Instance Credentials**

Issuing of verifiable credentials for Service Instances are not in scope for the first implementation phase.

In future iteration attestations for Service Instances SHOULD be considered (e.g., bare metal attestations). Conform to the technical architecture concept a trust and validation is also planned in future for Service Instances. In the architecture decision process this should be considered. ☐

☐ **IDM.OCM.00047 Connection Protocol**

A trusted communication between two parties MAY be established. If needed it MUST follow the connection protocol.⁷ All defined functions in the connection protocol MUST be integrated in the SSI Service Abstraction Service and are REQUIRED by Connection Manager, Attestation Manager and Request Manager. Those components MUST be able to use the connection protocol for interaction and extend those functions with specific subtasks. Extension example: Post the created ConnectionID to the Trust Service. ☐

☐ **IDM.OCM.00048 Issue Credential Protocol**

The Attestation Manager MUST follow the Issue Credential specification for messages, file-types, and payloads. The protocol defines all necessary roles, states, and the process for issuing a credential.⁸ All necessary endpoints to an external party are implemented and integrated in the SSI Abstraction Service which provides endpoints for Attestation Manager to receive updates for the credential issue protocol. This MUST include the decision for issuing a credential to a specific DID. The necessary request for decision MUST be done by the Attestation Manager to the Trust-Service endpoint. The response defines the respective call for Attestation Manager on the SSI Abstraction Service for further steps in the issue credential protocol. ☐

☐ **IDM.OCM.00049 Secure Timestamp**

The time related evaluations and creations a secure timestamp MUST be used. RFC 3161 from IETF MUST be used.⁹ ☐

☐ **IDM.OCM.00050 SSI Abstraction Service Subscription Context**

Subscribes to all updates from the SSI Abstraction Service. Relevant especially for Connections, Proofs, Issue Protocols. ☐

☐ **IDM.OCM.00051 SSI Abstraction Service Calls**

All other services are actively using the functionality provided by the SSI Abstraction Service. This means they all MUST implement calls to the functions that are necessary within their scope. ☐

☐ **IDM.OCM.00052 Subscription Context**

⁷<https://github.com/hyperledger/aries-rfcs/tree/9b0aaa39df7e8bd434126c4b33c097aae78d65bf/features/0160-connection-protocol>

⁸<https://github.com/hyperledger/aries-rfcs/blob/master/features/0036-issue-credential/README.md>

⁹<https://www.ietf.org/rfc/rfc3161.txt>

Attestation, Connection and Proof Manager are offering subscription endpoints. As additional parameters key: DID (if available), namespace: identity and scope:credential;definition;schema (if available) are provided to the emitted event to give the subscriber context about the subscription. ◻◀

▶◻ IDM.OCM.00053 **Trust Service Interaction Context**

All requests to Trust Service are including context specific information from the component and the current protocol or meta information regarding the request. ◻◀

▶◻ IDM.OCM.00054 **Queuing, Caching**

All services MUST be able to cache and queue current interactions with other services. E.g., a connection has not been established directly or a Trust Service call processes longer.

Constraints:

Caching component ◻◀

3.2.2. Connection Manager

▶◻ IDM.OCM.00055 **Connection States**

An established Connection based on the connection protocol can have two different states. “Active” in the case that the connection is just established but no further proofs are requested yet or “trusted” after specific attributes based on verifiable credentials have been requested and verified (official trusted Gaia-X member by AISBL credentials). The Connection state MUST be stored by the connection information. Trust state updates from Proof Manager are forwarded to SSI Abstraction Service to store/update the latest state by the connection. ◻◀

▶◻ IDM.OCM.00056 **Connection Information Request**

When the Connection Manager receives a connection information request the component MUST call the SSI Abstraction Service to get all issued credentials and present-proof items for the specific connection by transmitting the ConnectionID or DID as parameter. The received information MUST be composed and sent to the requesting component.

Interface:

[SSI Abstraction Service] Issue Credential Records (ConnectionID)

[SSI Abstraction Service] Present Proof Records (ConnectionID) ◻◀

▶◻ IDM.OCM.00057 **Auto-Accept Connections**

The Connection Manager SHOULD auto-accept (default policy) all incoming connection invitations from other parties. After a successful establishment of the connection the state MUST be stored by the connection. The initial state after establishment is “active”. This is needed as a base layer for communication. In future releases a policy SHOULD decide whether a connection MUST be accepted or not. ◻◀

▶▶ IDM.OCM.00058 **Connection Use**
 All invitations created by the Connection Manager MUST be single use in the first place. In future development multi-use SHOULD be considered. ◀◀

▶▶ IDM.OCM.00059 **Initial Connection State**
 The Connection Manager MUST check if a connection is fully established or not. If a connection is after a specific timespan not established and marked as “active” the connection MUST be removed. A specific timespan SHOULD be provided by Trust Service.

Constraints:

A connection-protocol flow has been started for a connection

The current state of the connection protocol for the specific connection is known

Interfaces:

[SSI Abstraction Service] - Delete connection (ConnectionID) ◀◀

▶▶ IDM.OCM.00060 **Maintain Not Completed Connection**
 The Connection Manager MUST be able to request all connections which are currently in the connection-protocol flow inclusive the current state. For all connections that have received no update since a predefined time the connection manager MUST delete the connection.

Interface:

[SSI Abstraction Service] - connections ◀◀

▶▶ IDM.OCM.00061 **Get Trusted Connection State Policy**
 If a new connection is established the general state is set to “active”. For further communication over this connection the state SHOULD be “trusted”. To get this the “trusted state. The Connection Manager MUST request the Trust Services with the Policy *GetTrustedConnectionState*. The Trust Service response contains a full proof request with all necessary attributes and restrictions following the Request Presentation Aries RFC definition.¹⁰ The response will be used to call the presentation request on the Proof Manager for establishment of a “trusted” connection. ▶▶ [IDM.OCM.00039 Presentation Request](#)

Constraint:

Established connection in state “active”

Interface:

[Trust Service] *GetTrustedConnectionState*

[Proof Manager] Presentation Request

¹⁰<https://github.com/hyperledger/aries-rfcs/tree/master/features/0037-present-proof#request-presentation>

Response: Presentation Request with all attributes and restrictions (Participant Credential & Organizational Credential) ◀◀

▶▶ IDM.OCM.00062 **Request Trusted Connection State**

After the Connection Manager has received the Presentation Request payload from *Get Trusted Connection State Policy* request for a specific connection which is in state “active” the Connection Manager MUST call the Proof Manager with the Presentation Request to update the current state of the connection.

Constraint:

Established connection

Get Trust Connection State Policy Payload (Presentation Request)

Interface:

[Proof Manager] Presentation Request (*Get Trusted Connection State Policy* Payload) ◀◀

▶▶ IDM.OCM.00063 **Create Invitation**

The Connection Manager MUST be able to create an invitation for a secure and permanent private channel. A request from another component is received on the Connection Manager and fulfilled via SSI Abstraction Service.

Interface:

[SSI Abstraction Service] - Create Invitation

Request:

Request on [SSI Abstraction Service] Create Invitation (POST(Alias, public))

Necessary attributes are defined based on role of the invitee and reference in the connection protocol

Response:

An appropriate response to the HTTP action from SSI Abstraction Service with a JSON object the invitation (can be formatted as URL with a Base64URLEncoding)

Acceptance Criteria:

The Connection Manager receives a valid invitation object ◀◀

▶▶ IDM.OCM.00064 **Connection Exist**

The Connection Manager MUST be able to call the SSI Abstraction Service to request the ConnectionID for a connection with a provided DID

Interface:

[SSI Abstraction Service]: connection (DID)

Response:

An appropriate http response with the ConnectionID as payload or empty

Acceptance Criteria:

1. Response to Requester send (another component) with ConnectionID + DID
2. Appropriate 200 HTTP response received 

 **IDM.OCM.00065 Serve Invitation**

The Connection Manager **MUST** be able to respond to an invitation request with a created invitation based on the request input data (POST). The created invitation **MUST** be provided by the SSI Abstraction Service on the Connection Manager - Connection Protocol Endpoint

Constraints:

Create Invitation Request received

Invitation json object (possibly a Base64URLEncoded json object) received

Interface:

[Requester Scope]: OCM Internal Component

Invitation json object (possibly a Base64URLEncoded json object) from SSI Abstraction Service
Request Details to the original requester

Response:

An appropriate http response

Acceptance Criteria:

3. Response to Requester send
4. Appropriate 200 HTTP response received 

 **IDM.OCM.00066 Connection Invitation Update Received**

The Connection Manager **SHOULD** be able to call an endpoint on the Trust Service component to get a decision regarding a new invitation request or an update for an already sent out invitation. This includes also a possible rejected response for a connection from an external party. This call is triggered by an incoming connection update from SSI Abstraction Service on the Connection Manager. The respective response from Trust Service **MUST** resolve in a call “accept-invitation”, “remove-invitation”.

Constraints:

Invitation json object received

Interfaces:

[Requester Scope]: OCM Internal Component

[Trust Services] Connection

Request to [Trust Services]

POST: Invitation as JSON Object (Base64URLEncoding MAY be supported), ConnectionID

Request to [SSI Abstraction Service]:

GET: accept-invitation (ConnectionID) or remove connection (ConnectionID)

Response from [Trust Services]

An appropriate http response

Response from [SSI Abstraction Service]

An appropriate http response

Acceptance Criteria:

5. Response to Requester send
6. Appropriate 200 HTTP response ◀◀

▶ IDM.OCM.00067 **Update Connection Subscriber**

[Requester Scope]: OCM Internal Component

The Connection Manager **MUST** be able to forward new connection updates to all subscribers.

Constraints:

Existing connection

Not empty list of subscribers for a specific ConnectionID ◀◀

3.2.3. Proof Manager

▶ IDM.OCM.00068 **Cryptographic Hardening of threadIDs**

It **MUST** be cryptographically ensured that the threadID (which functions as a correlation ID) which is used for the state polling is not guessable or fakeable. The ThreadID can be generated by a 3rd party system (e.g., an existing IdM / authorization service) to be used in the protocol flow. ◀◀

▶ IDM.OCM.00069 **Presentation Request to establish a Trusted Connection**

The Proof Manager receives a presentation request for an “active” connection and **MUST** call the Trust Service for the *TrustedConnectionCredentials Policy* with the received payload (Presentation request). As a response the Trust Service provides an answer if the request has to be answered or

not. If the answer is “yes” the Proof Manager MUST call the SSI Abstraction Service to answer the presentation Request with available Credentials.

Interface:

[Requester Scope]: External Component (via SSI Abstraction Service)

[SSI Abstraction Service] Present Proof (POST with credential_ids) ◀◀

▶ IDM.OCM.00070 **Presentation Received for Trusted Connection**

The goal of this function is to change the connection state from “active” to “trusted”. The Proof Manager receives a presentation for an earlier established connection over SSI Abstraction Service which is currently in “active” state. The Proof Manager MUST call the Trust Service with the received presentation as Payload for the *TrustedConnectionUpdate Policy*. The Trust Service will update the Connection State over the Connection Manager based on the provided presentation.

Constraint:

Existing connection in “active” state

Interface:

[Requester Scope]: External Component (via SSI Abstraction Service)

[Trusted Services] TrustedConnectionUpdate Policy ◀◀

▶ IDM.OCM.00071 **Presentation Request Received**

The Proof Manager receives a presentation request and MUST check if necessary credentials are available to respond to the request. Therefore, the Proof Manager requests the SSI Abstraction Service to receive the needed credentials. If all needed credentials are present the Proof Manager MUST call the Trust Service for *ProofRequestResponse Policy* including the received Presentation Request to get a decision if the request shall be answered or not. Iteration 1 scope does not include further Presentation Requests from Trust Service to the requesting party to get more information. The response from Trust Service is either respond with existing credentials (Iteration 1 scope does not include a mapping of best fitting credentials if some attributes can be answered with multiple credentials that are present) over the SSI Abstraction Service or do nothing.

Interface:

[Requester Scope]: External Component (via SSI Abstraction Service)

[SSI Abstraction Service] Get Credentials

[SSI Abstraction Service] Present Proof ◀◀

▶ IDM.OCM.00072 **Check Received Presentations**

The Proof Manager MUST periodically check if received presentations are still valid (time validity). Therefore, the Trust Service MUST provide a list to match which presentations MUST be observed

and when a new Presentation Request has to be sent out to the respective party. The Proof Manager MUST call the *PresentationFreshnessState Policy* from Trust Service to receive the list. The list MAY be cached for a specific time. If a presentation for an organization or participant is invalid the Proof Manager MUST request a new presentation and replace the old one.

Interface:

[Trust Service] PresentationFreshnessState Policy 

 IDM.OCM.00073 **Check Received Presentations for Revocation State**

The Proof Manager MUST periodically check if received presentations are still not revoked. Therefore, the Trust Service MUST provide a list to match which presentations MUST be observed and when a proof of non-revocation MUST be sent out. The Proof Manager MUST call the *PresentationRevokationState Policy* from Trust Service to receive the list. The list MAY be cached for a specific time.

Interface:

[Trust Service] PresentationRevokationState Policy 

 IDM.OCM.00074 **Proof of non-Revocation Received**

When the Proof Manager received a Proof of non-Revocation request it MUST call the Trust Service for *ProofOfNonRevocation Policy* (proof has already been sent out earlier) with the provided request to get a decision if the request shall be answered or not. The Trust Service responds with either an “answer” where the Proof Manager calls the SSI Abstraction Service to answer the non-Revocation request or “no answer” where the Proof Manager cancels the answer process.

Interface:

[Requester Scope]: external component (via SSI Abstraction Service)

[Trust Service] PresentationRevokationState Policy 

 IDM.OCM.00075 **Update Proof Subscriber**

The Proof Manager MUST be able to forward new proof events from SSI Abstraction Service to all subscribers.

Constraints

Not empty list of subscribers

Interface:

[Requester Scope]: OCM Internal Component 

3.2.4. Attestation Manager

3.2.4.1. Revocation



IDM.OCM.00076 **Revocation Support**

The Attestation Manager MUST support revocation of issued credentials in a future release and is for Iteration 1 handled by expiration date in verifiable credentials. Revocation MUST be fast in focus on time and communication. Especially the revocation testing MUST preserve privacy without correlation to existing other credentials or subjects. As reference for an existing implementation the credential revocation concept of hyperledger indy MAY be used but it needs to be decided individually for a specific context. Respective endpoints MUST be defined by implementation time to fulfil the revocation process flow requirements.¹¹

3.2.4.2. Trusted Connection



IDM.OCM.00077 **Establish Secure Connection**

All Attestation Manager interactions with other participants or entities MUST use a “trusted” connection.



IDM.OCM.00078 **Credential-Issue Request Received**

When a new credential-issue request is received on the endpoint, the Trust Service for *CredentialIssueRequest Policy* is called. The response from Trust Service is either issue credential with the provided structure and information for the issuing process or do not issue. In case that the credential MUST be issued the Attestation Manager creates a new credential definition first, if it doesn't exist yet. The Attestation Manager calls the SSI Abstraction Service with the provided information to issue the credential to the requester. This includes the possibility for self-attestation.

Interfaces:

[Requester Scope]: OCM Internal Component

[SSI Abstraction Service] Issue Credential



IDM.OCM.00079 **Create Credential-Proposal Request**

When a new credential-proposal request is received on the endpoint, the Attestation Manager MUST call the SSI Abstraction Service to send the Credential-Proposal over an existing “Trusted” connection. Initially, it MUST be checked if a “trusted” connection is available for the issuing party. If no “trusted” connection is present the process for Establish secure connection MUST be followed before the credential-proposal is sent out to the issuing party over the newly existing “trusted” connection. This includes ConnectionID, credential-definition-id and all to be attested attributes.

Interface:

[Requester Scope]: OCM Internal Component, External Component (via SSI Abstraction Service)

¹¹ <https://hyperledger-indy.readthedocs.io/projects/hipe/en/latest/text/0011-cred-revocation/README.html>

[SSI Abstraction Service] issue-credential send proposal (POST with credential-proposal) 

3.2.4.3. Schema & Credential



IDM.OCM.00080 **Create Schema**

When the Attestation Manager receives a new create schema request on its endpoint, it MUST request the SSI Abstraction Service to query all schemas that are present. The Attestation Manager MUST check if the response already contains a schema with the provided attributes excluding the version. If the schema does not exist a new schema MUST be created within SSI Abstraction Service with the provided information.

Interface:

[Requester Scope]: OCM Internal Component

[SSI Abstraction Service] schemas created (GET)

[SSI Abstraction Service] schemas (POST with schema) 



IDM.OCM.00081 **Update Schema**

When the Attestation Manager receives a new update schema request on its endpoint it MUST request the SSI Abstraction Service to query all schemas that are present. If there is no schema existing the Attestation Manager has to call the create schema to create an initial version of the schema. If a schema exists, the version of the received schema on the endpoint needs to be higher. If so, the Attestation Manager calls the create schema to update the schema.

Interface:

[Requester Scope]: OCM Internal Component

[SSI Abstraction Service] schemas (POST with schema)

[SSI Abstraction Service] credential-definitions (POST with schema information) 



IDM.OCM.00082 **Create Credential-Definition**

The Attestation Manager MUST be able to create a new credential definition for a specific schema. The Attestation Manager checks if a respective credential definition is in place by calling the SSI Abstraction Service to request all existing credential definitions. If one matches the proposed new one, nothing is to do. If no credential definition exists a new definition MUST be created by passing the credential definition information to the SSI Abstraction Service containing schema_id, revocation specific information, tags.

Interface:

[Requester Scope]: OCM Internal Component

[SSI Abstraction Service] Credential-Definition created (GET)

[SSI Abstraction Service] Credential-Definition (POST with credential definition information) 



IDM.OCM.00083 **Verifiable Credential Definition**

General requirements regarding verifiable credentials are defined in [\[IDM.AO Chap-4.7\]](#) including the initial set of verifiable credentials that are used in the Gaia-X context. ☐

☐ IDM.OCM.00084 **Credential Validity**

All issued credentials from the Attestation Manager MUST include an expiration date. ☐

☐ IDM.OCM.00085 **Check Credential State**

The Attestation Manager MUST periodically check if issued credentials are still not revoked and the expiration date is not reached. If a credential is not valid any more the Attestation Manager MUST call Trust Service for *CredentialFreshnessState Policy* to get a decision if a specific credential needs to be re-issued or not. The response is either “issue new” where the Attestation Manager MUST issue the specific credential again or “no” where the Attestation Manager MUST remove the credential within the SSI Abstraction Service. Iteration 1 does not include a notification service if a credential has been revoked.

Interface:

[Trust Service] CredentialFreshnessState Policy

[SSI Abstraction Service] Remove Credential Record

[SSI Abstraction Service] Issue Credential ☐

☐ IDM.OCM.00086 **Update Attestation Subscriber**

The Attestation Manager MUST be able to forward new attestation/credential events from SSI Abstraction Service to all subscribers.

Constraints

Not empty list of subscribers ☐

☐ IDM.OCM.00087 **Get Issue-List (DIDs) for Schema**

When a request on the endpoint is received, the Attestation Manager MUST be able to call the SSI Abstraction Service to get all issued credentials for the provided schema. A configuration option MAY be provided e.g., if revoked credentials or expired are not removed from the result set. The response list MUST be filtered according to the selected configuration options. The remaining subject DIDs are extracted and provided to the requesting entity. This allows e.g., the AISBL to get a list of all “trusted” participants in the Gaia-X ecosystem. ☐

3.2.5. Principal Manager

☐ IDM.OCM.00088 **Principal Manager Authentication Flow**

The Authentication (Basic) Flow MUST follow OpenID Connect Specification. It MUST be able to fetch information about the authenticated user from the UserInfo Endpoint¹²

The necessary parameters MUST be configurable. This includes, but is not limited to:

- client_id, client_secret
- endpoints
- redirect urls
- basic profile

The system MUST support the Basic Relying Party Profile [\[OIDC.Conformance\]](#)

to further issue a credential over the connection [▶ IDM.OCM.00022 Credential-Issue Endpoint](#)

Iteration 2: The Authentication flow MUST be customizable to support other authentication mechanisms than OpenID Connect to support further internal, sometimes proprietary IdM solutions.



▶ IDM.OCM.00089 **Principal Manager Credential Issuing**

After a connection to the Principal is established and authenticated according to [▶ IDM.OCM.00088 Principal Manager Authentication Flow](#), a PrincipalCredential is issued to the Principal through the Attestation Manager [▶ IDM.OCM.00022 Credential-Issue Endpoint](#)

[Requester Scope]: OCM Internal Component

▶ IDM.OCM.00090 **Principal Manager Backend**

The backend is connected to the frontend. It MAY serve the frontend directly or optionally leave this to a 3rd party http(s) server. Configuration MUST be possible. Logging MUST be configurable.

▶ IDM.OCM.00091 **Principal Manager Schema Mapping**

The component MUST be able to provide configuration options to map any incoming *UserInfo* field values to the *PrincipalCredential* schema.

3.2.6. SSI Abstraction Service

▶ IDM.OCM.00092 **Core SSI Functionality**

The SSI Abstraction Service MUST provide access to all the core cryptographic functionality required to implement the Aries Interop Profile 1.0 (AIP 1.0).¹³ This SHOULD be achieved by wrapping existing

¹² https://openid.net/specs/openid-connect-core-1_0.html#UserInfo

¹³ <https://github.com/hyperledger/aries-rfcs/blob/8526600048b88115b5aaa6d9b4b0120bc4e69bae/concepts/0302-aries-interop-profile/README.md>

frameworks like the open-source implementations currently ongoing in the Hyperledger umbrella project.

The SSI Abstraction service MUST provide the functionality for other components to subscribe to events and if necessary, store a list of subscribers for internal routing.

The SSI Abstraction service MUST be able to create and maintain a DID Document on the verifiable data registry. This functionality MUST be made available to other components and in an administration context. ◀◀

3.2.7. Profile Manager

▶◻ IDM.OCM.00093 **Credential Transformation**

Indy Credentials (Anonymous Credentials) and W3C JSON-LD Credentials are not fully compatible with each other yet.

The transformation transforms credentials from Indy Credentials into W3C JSON-LD Credentials.

Since the original proof gets lost on this transformation, a new ProofType needs to be introduced ▶◻ [IDM.OCM.00094 CredDefProofType](#) ◀◀

▶◻ IDM.OCM.00094 **CredDefProofType**

This function implements a new ProofType based on JSON-LD Credential / Presentation input and a separate Presentation Request over a DIDComm connection to prove the claims from the JSON-LD document.

Input example:

```
"proof": {
  "type": "IndyCredDefProofType",
  "proofPurpose": "assertionMethod",
  "verificationMethod": "M6Mbe3qx7vB4wpZF4sBRjt:3:CL:571:bank_account_no_revoc"
}
```

The verification Method contains the “Credential Definition ID”. Together with the subject’s identifier (and from there, a resolvable endpoint), all necessary information is given to request a ProofRequest with the given attributes and constraints to receive a true/false (valid/not valid) result.

Returns true/false result and a reference ID to the exchanged Proof.

Since this is a relatively new method, alternatives MUST be considered in case this is not a community accepted method. ◀◀

▶◻ IDM.OCM.00095 **Self-Description Content**

To decide whether a Credential is part of the Public Self-Description, it MUST be tagged accordingly from the Trust Service.

The Trust Service *PublicProfileCredentials Policy* returns which credentials are part of the Self-Description. The response contains a list of “Issuer” / “Schema” combinations. This MUST be matched against the OCM stored Verifiable Credentials.

Interface:

[Requester Scope]: OCM Internal Component

[Trust Service] PublicProfileCredentials Policy 

IDM.OCM.00096 **Self-Description Output**

The transformed Credentials are exported into the “verifiableCredential” property of the Verifiable Presentation structure [VC.DataModel]

The Transformed Credentials  [IDM.OCM.00093 Credential Transformation](#) do not contain a 3rd party (issuer) proof itself but are “self-signed” via the entire Verifiable Presentation proof.

The component must remember a mapping from the original VC to the one as part of the Self-Description, represented by a unique VC ID. Once the original VC has changed (e.g., through auto-renewal), the corresponding VC as part of the Self-Description receives a new VC ID. Thus, reading components can easily detect updates.

In case of an update of the Self-Description the component MAY inform all active connections about the new Self-Description content with an intent. 

3.2.7.1. Service-Offering

IDM.OCM.00097 **Service-Offering CRUD**

The component contains CRUD features for Service Offerings maintained by the OCM.

The Service Offering is exported in the form of individual Verifiable Credentials. First, with the schema content “service type” and “description”.

More detailed schema MUST be implemented as soon as available.

Individual Service Offerings do not have their own Identity / DID in the first iteration. 

IDM.OCM.00098 **Service Offering Export**

Available Service Offerings are exported to a JSON-LD Verifiable Presentation structure, including a signature (“proof”) under the configured Service-Offering Endpoint.

A detailed description can be found in [IDM.AO chapter 8.9. Self-Description Anatomy].

In case of an update the component MAY inform all active connections about the new content with an intent. ☐

3.3. Other Nonfunctional Requirements

3.3.1. HTTP Requirements



IDM.OCM.00099 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.OCM.00100 HTTP Protocol Definitions

All HTTP Endpoints MUST follow RFC 7231¹⁴ and RFC 5789¹⁵, but it MAY be chosen what of the protocols is necessary to realize the functionality. For problem reports the RFC7807¹⁶ MUST be used in combination with Standard HTTP Error Codes. ☐

3.3.2. Configuration



IDM.OCM.00101 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. ☐

3.3.3. Logging Requirements



IDM.OCM.00102 Data Minimization

From GDPR perspective the product MUST NOT log data which is related to personal information. (e.g., User Names, Birth Dates etc.) The product MUST only log data, which is relevant to technical operations, except for the purpose that, in the event of an incident, enable reconstruction of the sequence of the message exchange for establishing the place and the nature of the incident. 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

¹⁴ <https://tools.ietf.org/html/rfc7231>

¹⁵ <https://tools.ietf.org/html/rfc5789>

¹⁶ <https://tools.ietf.org/html/rfc7807>

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



▶▶ IDM.OCM.00103 **Logging Frameworks**

The product MUST support the OpenTelemetry Standard and 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 potentially the most common open-source logging solutions. 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. ◀◀

3.3.4. Monitoring Requirements

▶▶ IDM.OCM.00104 **Monitoring Frameworks**

The product MUST support monitoring frameworks e.g., grafana to support the analysis of incoming data by the enterprise infrastructures. The supported framework MAY be chosen for the first version, but it MUST support potentially the most common monitoring solutions. (e.g., Zabbix) 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. ◀◀

▶▶ IDM.OCM.00105 **Alerting Frameworks**

Additional to the Monitoring Frameworks an Alerting framework (e.g., Prometheus or Cloud Based) MUST/MAY be in place at least in the System nodes to promptly communicate to e.g., System Administrators or owners the occurrence of an event in form of a security incident or application/system malfunction or anomaly. ◀◀

3.3.5. Performance Requirements

▶▶ IDM.OCM.00106 **Performance Scalability**

The performance of the product MUST be scalable. This MUST be demonstrated in a load demonstration example. The optimal scalability SHOULD be in the best case a linear behavior of minimum 50% more performance by each additional instance. ◀◀

▶▶ IDM.OCM.00107 **Performance by Design**

The product SHOULD be designed and implemented in a way, that the implementation is non-blocking and performance oriented. It SHOULD be a microservice architecture, but it MAY follow other concepts. The decision MUST be documented. ◀◀

3.3.6. Safety Requirements

▶▶ IDM.OCM.00108 **Recovery Point Objective (RPO)**

The RPO for the product MUST be 0 for a single and multiple instance(s). It MAY be higher by configuration or deployment, decided by the user. ◀◀

- ▶ IDM.OCM.00109 **Recovery Time Objective (RTO)**
The RTO for the product MUST be one Minute for a single instance. For multiple instances the RTO MUST be 0. ◀

- ▶ IDM.OCM.00110 **Mitigation of Single Point of Failure threats**
Critical components in the GAIA-X Ecosystem MUST be identified and strategies to warranty their availability and scalability MUST be implemented. ◀

3.3.7. Security Requirements

3.3.7.1. General Security Requirements

Each Gaia-X Federation Service MUST meet the requirements stated in the document “Specification of non-functional Requirements Security and Privacy by Design” [\[NF.SPBD\]](#). Federation Services specific requirements will be documented in the next chapter.

3.3.7.2. Service Specific Security Requirements

This chapter will describe the service specific requirements, which will extend the requirements defined in the chapter above.

- ▶ IDM.OCM.00111 **Cryptographic Algorithms and Cipher Suites**
Cryptographic algorithms and TLS cipher suites SHALL be chosen based on the recommendation from the German Federal Office for Information Security (BSI) or SOG-IS. These recommendations and the recommendations of other institutions and standardization organization are quite similar¹⁷ [\[Cryptolen\]](#). The recommendations can be found in the technical guidelines¹⁸ TR 02102-1 [\[TR02102-1\]](#) and TR 02102-2 [\[TR02102-2\]](#) or SOG-IS Agreed Cryptographic Mechanisms¹⁹ [\[SOG-IS\]](#). ◀

- ▶ IDM.OCM.00112 **Digital Certificates**
For digital certificates and cryptographic signatures in the context, the major requirements on cryptographic algorithms and key length MUST meet the definitions in the following table (as of 2020):

Signature Algorithm	Key size	Hash function
EC-DSA	Min. 250 Bit	SHA-2 with an output length \geq 256 Bit or better
RSA-PSS (recommended) RSA-PKCS#1 v1.5 (legacy)	Min. 3000 Bit RSA Modulus (n) with a public exponent $e > 2^{16}$	SHA-2 with an output length \geq 256 Bit or better

¹⁷ See <https://www.keylength.com/en> for a comparison

¹⁸ See https://www.bsi.bund.de/EN/Service-Navi/Publications/TechnicalGuidelines/tr02102/tr02102_node.html

¹⁹ See <https://www.sogis.eu/documents/cc/crypto/SOGIS-Agreed-Cryptographic-Mechanisms-1.2.pdf>

DSA	Min. 3000 Bit prime p 250 Bit key q	SHA-2 with an output length \geq 256 Bit or better
-----	--	---

Table 4: Requirements on cryptographic algorithms and key length

Named curves SHALL be used for EC-DSA (e.g., NIST-p-256). ☐

☐ **IDM.OCM.00113 TLS Certificate Validity Periods**

In general, the recommended validity period for a certificate used in the system should be one year or less. Under some circumstances (for example RootCA) the certificate validity can be extended. Certificate owners MUST ensure that valid certificates are renewed and replaced before their expiration to prevent service outages. ☐

☐ **IDM.OCM.00114 Security by Design**

The software security MUST be from the beginning a design principle. Means separation of concerns, different administrative roles, especially for private key material and separate access to the data MUST be covered from the first second. It MUST be described in the security concept, what are the different security risks of the product and how they are mitigated (e.g., by Threat Modeling Protocols) ☐

☐ **IDM.OCM.00115 Installation of Critical Security Updates**

Node operators SHALL deploy security critical updates without undue delay. ☐

☐ **IDM.OCM.00116 Avoid HTTP Request Smuggling**

To avoid Request Smuggling attacks, the product MUST implement a standard which handles this kind of attack by design, because the attack vector results in an insufficient implementation of the header handling. The chosen way to handle it MUST be shared to the other implementers of all other subcomponents within IDM & Trust and MUST be described in the security concept. ☐

☐ **IDM.OCM.00117 HTTP Pentesting**

All HTTP parts of the product has to be pen tested, for the following criteria:

- 1) Unauthorized Access to the System MUST be tested
- 2) Unauthorized Actions MUST be triggered without a user action
- 3) Endpoints MUST be tested for HTTP smuggling attack vectors
- 4) If a datastore is present over HTTP, illegal data access MUST be tested

It's RECOMMENDED to test more attack vectors and document it for the purpose to mitigate it in later versions. ☐

☐ **IDM.OCM.00118 Storage of Secrets**

The storage of secret information such as private keys MUST take place in state-of-the-art secure environments to protect secret data confidentiality and integrity. Examples of this are Secure

Enclaves, TPMs, HSM or Secure Vaults. In case (Personal) Agents are not equipped with a secure storage it MAY also be possible to store the secrets in a third party (e.g., Cloud) provider (e.g., Secure Wallet) that MUST provide overall the same level of security as the aforementioned methods. ☐

☐ IDM.OCM.00119 **Secret Distribution and Usage**

The product MUST ensure interoperability of cryptographic primitives and components by public standards and MUST use secure state of the art methods to create and import secrets into the secure storage, as well as performing cryptographic operations (e.g., encryption or digital signatures). For Key distribution, state of the art DKMS methods MUST be implemented. ☐

☐ IDM.OCM.00120 **Support for Potential Requirements for Secret Storages**

Devices that hold cryptographic information and perform cryptographic functions MUST be compliant with the standard PKCS #11. Moreover, the products MUST be potentially eligible for a FIPS-140-2 or ETSI/Common Criteria certification with the minimum-security level necessary to operate securely in the Gaia-X ecosystem. Security Levels in FIPS-140-2 range from 1 to 4. Current HSM Cloud Service offerings (aws, azure, gcp) are Level 3 (Source: https://en.wikipedia.org/wiki/FIPS_140-2). ☐

☐ IDM.OCM.00121 **Special Availability and Scalability Requirements for Secret Storage Components**

Secret Storage components play a central role in storage, encryption, and digital signing in the Gaia-X ecosystem, thus they can become a single point of failure for a Gaia-X participant, for example an organization. Therefore, methods and procedures to ensure the availability and scalability of the Secret Storage functionality MUST be implemented. ☐

3.3.8. Software Quality Attributes

☐ IDM.OCM.00122 **Quality Aspects**

The software MUST meet the following requirements:

- The quality standards MUST meet ISO 25010 [ISO25000]
<https://iso25000.com/index.php/en/iso-25000-standards/iso-25010>
- Robustness / Reliability
- Performance
- Availability must be 24/7
- Interoperability with the other work packages
- Security
- Adaptability / expandability
- Maintainability and Code Quality
- Scalability

Major security concerns regarding design and implementation MUST be documented and highlighted to the steering board. Minor security concerns SHALL be documented and mitigated. ☐☐

3.4. Compliance

- ☐▶ IDM.OCM.00123 **GDPR Audit Logging**
All GDPR relevant access to personal relevant data MUST be logged for a later audit. ☐☐
- ☐▶ IDM.OCM.00124 **GDPR Data Processing**
If it is 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 person relevant data MUST be deleted after the processing, if applicable. ☐☐

3.5. Design and Implementation

Please also refer to [\[TDR\]](#) for further requirements.

3.5.1. Distribution

- ☐▶ IDM.OCM.00125 **Config Data Distribution**
The product SHOULD support a global data distribution of config data to synchronize configurations between multiple regions in the world. Built-in synchronization technology (asynchronous and synchronous) MAY be used. ☐☐

3.5.2. Maintainability

- ☐▶ IDM.OCM.00126 **Micro Service Architecture**
For a better scaleout, maintainability and decentralization, the product architecture MUST have a micro service architecture. Each microservice MUST NOT be limited on the lines of code or number of days to implement it. The service “size” SHOULD be oriented on the fine granular business capabilities. (e.g., Order, ListMenu, Payment) ☐☐
- ☐▶ IDM.OCM.00127 **Domain Driven Design**
To support the micro service architecture within the maintainability, it MUST be declared a domain model 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. ☐☐

3.5.3. Operability

- ☐▶ IDM.OCM.00128 **FTE Estimation**

The product MUST be designed so that over scripts and tools one FTE within a Month SHOULD host and operate the product without any third-party help. It MUST be sketched in the operations concept how this can be achieved. If this target is not reachable it MAY be explained and described why the effort is higher and appropriate. ◀

3.5.4. Interoperability



IDM.OCM.00129 Interoperability of IT security features and algorithms

The following interoperability requirements of the respective IT security features and algorithms MUST be ensured across the system components:

- Interoperability of crypto algorithms and protocols (including the novel peer-reviewed ones through the established bodies and communities)
- Interoperability of secure secret transfer protocols (such as the holistic usage of PKCS#11 for HSM communication, etc.)
- Format interoperability of crypto material (such as the holistic usage of PKCS#12 for relevant cases) ◀

3.5.5. Scalability



IDM.OCM.00130 Key Infrastructure Scalability

Specific components require a secure and trustful key infrastructure. The used key infrastructure MUST be scalable in a way that multiple deployments of the components are able reuse the existing key infrastructure in an atomic and non-blocking way. The implementation of such an infrastructure and load balancing is out of scope. ◀

4. System Features

4.1. Integration Overview

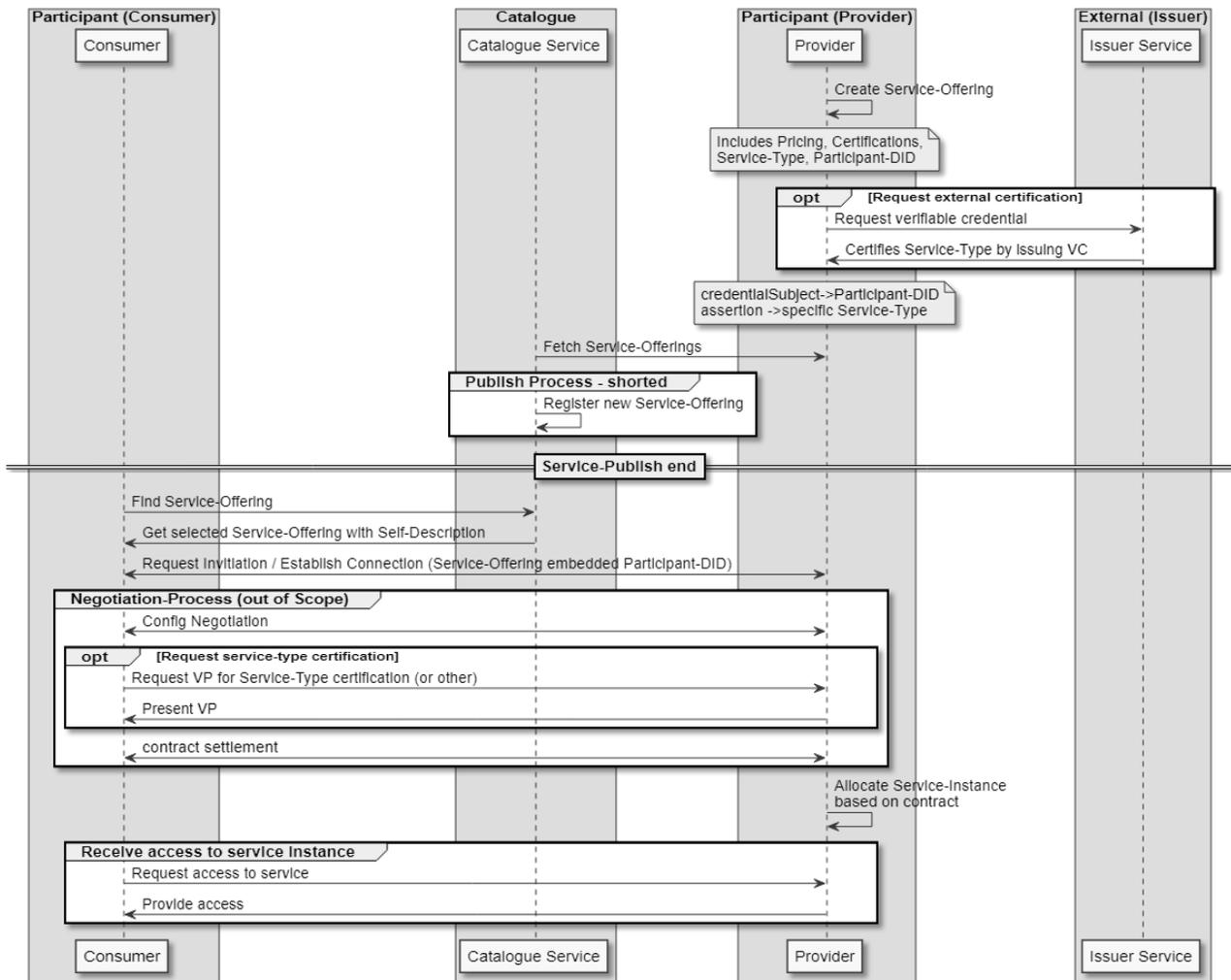


Figure 3: High Level Interaction View for Gaia-X Service-Offering and Service-Consumption Process

4.1.2. Description

The product is reused by a wide variety of other services and requires corresponding interfaces for comprehensive communication. The sequence shown describes a coherent process that includes the functionalities of the product. The service offering process in particular provides an overview of the essential functions for the participant identity in the overarching role as provider and consumer.

The service management process shows the creation and offering of a service offering, the processing of attestations regarding a service type as well as the interface to the Federated Catalogue. The service provider creates a service offering through internal systems in the form of a document that contains metadata such as price lists, service type and other information about the service. After creation, certification for this service type can be requested from third parties. This uses the functionality of the Request Verifiable Credentials of

the OCM, which allows the request and attestation of certain information from third parties. After an optional attestation has taken place, this entry is crawled by the Federated Catalogue and included in the list of services.

Another functionality of the OCM becomes necessary during the booking of a service by a participant in the role of consumer. After selecting a service in the Federated Catalogue, there is a need for verification of attestations of a certain service type by the consumer. The OCM provides the possibility to send a proof request to the participant in the role of provider to request attestations of the service type. The OCM can send this request to the participant and forward the answer to the trust service for validation.

Via the Federated Catalogue, a service consumer can find the service offering and establish a trustworthy connection to the service provider through the embedded information. This connection can be used to negotiate a contract and to request certifications or attestations. After concluding a contract for a certain configuration of the service, the service provider creates a corresponding service instance and allows the service consumer to access it.

4.2. Principal Manager

Principal Manager is a component to bridge between internal IdM and SSI. Its core is to authenticate a Principal (user) with an existing internal IdM and based on that, issues a Verifiable Credential to the Principal (user) who holds this credential in their Personal Credential Manager (PCM).

The Principal Manager uses features from the OCM Common Functionalities, mainly the Attestation Manager, and interacts with the Trust Services, more specifically the Policy Decision Engine.

4.2.1. Description and Priority

The Principal Manager handles the internal identities of all principals within an organization that can act on behalf of the organization. Examples of such principals are employees in the first iteration. Future releases include technical systems automating workflows. The organization offers credentials to these principals, allowing them to act within the Gaia-X ecosystem in the name of the organization within certain scopes.

Stimulus/Response Sequences

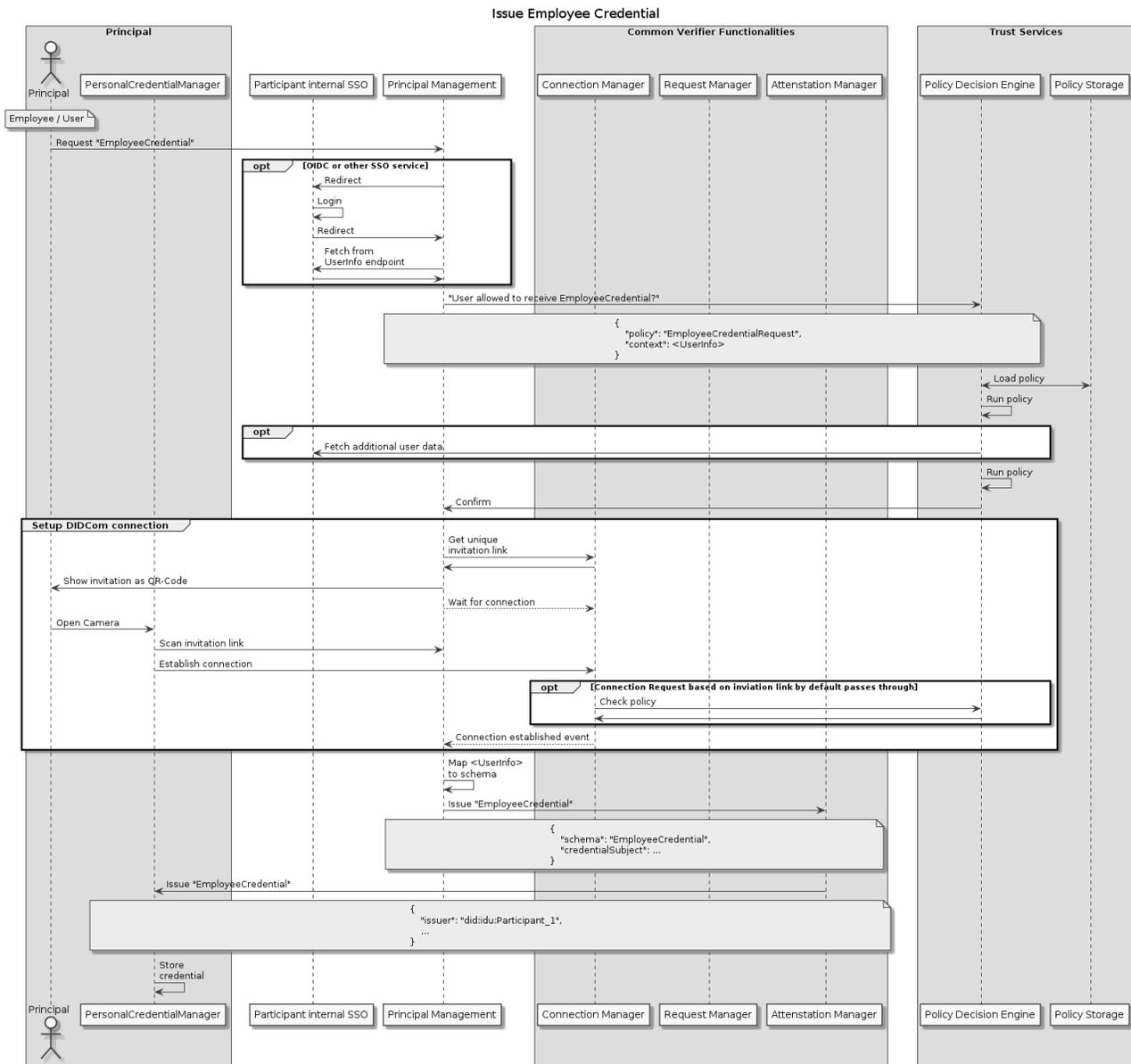


Figure 4: Issue Employee Credential

The Principal Manager landing page is the starting point for the issuing process. It redirects the user to the Participant’s internal login component. Iteration 1 supports OIDC flow.

After a redirect back to the Principal Manager, user information is fetched from the UserInfo endpoint. The UserInfo is attached to the context object for the given request.

The context, together with the policy execution request for *IssueEmployeeCredential* is sent to the Policy Decision Engine.

Option A: The Policy Decision Engine returns true or false.

true: The Principal Manager issues *SimpleEmployeeCredential* with values filled with UserInfo values.

Option B: The Policy Decision Engine MAY fetch additional user data. It further returns a Verifiable Credential structure filled with content.

The Principal Manager signs and issues the VC to the Principal.

4.2.2. Functional Requirements

<i>User Interface</i>
 IDM.OCM.00010 Principal Manager Frontend
<i>Endpoints</i>
 IDM.OCM.00043 Principal Manager Trust Service Interaction
 IDM.OCM.00044 Principal Manager Connection Manager Interaction
 IDM.OCM.00045 Principal Manager Attestation Manager Interaction
<i>Functions</i>
 IDM.OCM.00088 Principal Manager Authentication Flow
 IDM.OCM.00089 Principal Manager Credential Issuing
 IDM.OCM.00090 Principal Manager Backend
 IDM.OCM.00091 Principal Manager Schema Mapping

Table 5: Functional Requirements Principal Manager

4.3. Connection Manager

4.3.1. Description and Priority

The Connection Manager allows you to establish a secure connection to other parties based on the Connection Protocol. This includes the processing of incoming connection requests as well as the issuing of so-called invitations, which can be sent as connection requests to third parties or displayed for them. The individual requests are forwarded to the trust service component for validation and assessment of whether corresponding connections may be established and processed. The other components of the OCM can retrieve information regarding existing connection requests and inform themselves about the current status of the connections.

4.3.2. Stimulus/Response Sequences

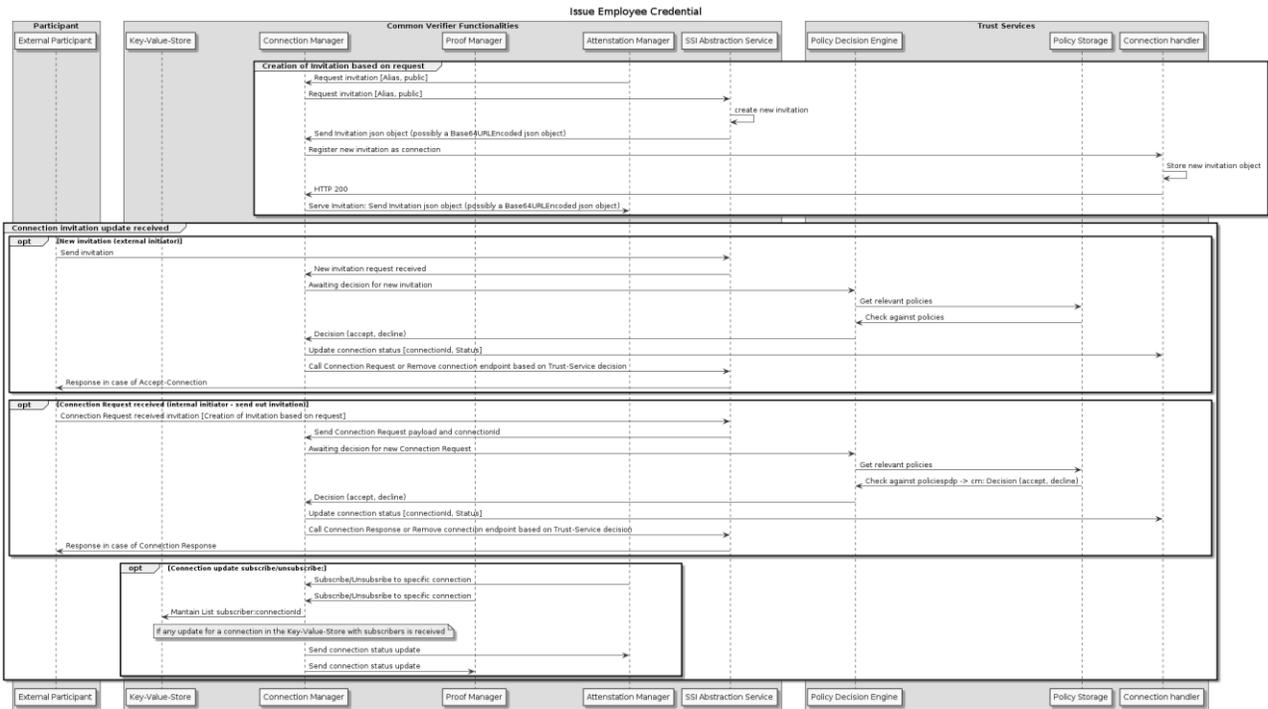


Figure 5: Connection Manager example flows

4.3.3. Functional Requirements

IDM.OCM.00018 Create Invitation Endpoint
IDM.OCM.00019 Connection Information Endpoint
IDM.OCM.00020 Connection Status Subscribe Endpoint
Functions
IDM.OCM.00055 Connection States
IDM.OCM.00056 Connection Information Request
IDM.OCM.00057 Auto-Accept Connections
IDM.OCM.00058 Connection Use
IDM.OCM.00059 Initial Connection State
IDM.OCM.00060 Maintain Not Completed Connection
IDM.OCM.00061 Get Trusted Connection State Policy
IDM.OCM.00062 Request Trusted Connection State
IDM.OCM.00063 Create Invitation

◀◀ IDM.OCM.00064 Connection Exist
◀◀ IDM.OCM.00065 Serve Invitation
◀◀ IDM.OCM.00066 Connection Invitation Update Received
◀◀ IDM.OCM.00067 Update Connection Subscriber

Table 6: Functional Requirements Connection Manager

4.4. Proof Manager

4.4.1. Description and Priority

The product provides necessary functionality to maintain inbound and outbound proof requests. This includes the interaction with Trust Service to verify necessary attributes to be requested for outbound proof requests and for inbound proof requests the policy to construct the respective response. The product is also responsible for expiration and revocation checks for already received proofs from external parties.

4.4.2. Functional Requirements

Endpoints
▶▶ IDM.OCM.00039 Presentation Request
▶▶ IDM.OCM.00040 Presentation Status Subscribe Endpoint
▶▶ IDM.OCM.00041 Verify JSON-LD Presentation Endpoint
▶▶ IDM.OCM.00042 Verify Indy CredentialDefinition ProofType Endpoint
Functions
▶▶ IDM.OCM.00068 Cryptographic Hardening of threadIDs
▶▶ IDM.OCM.00069 Presentation Request to establish a Trusted Connection
▶▶ IDM.OCM.00070 Presentation Received for Trusted Connection
▶▶ IDM.OCM.00071 Presentation Request Received
▶▶ IDM.OCM.00072 Check Received Presentations
▶▶ IDM.OCM.00073 Check Received Presentations for Revocation State
▶▶ IDM.OCM.00074 Proof of non-Revocation Received
▶▶ IDM.OCM.00075 Update Proof Subscriber

Table 7: Functional Requirements Proof Manager

4.5. Attestation Manager

4.5.1. Description and Priority

The product must provide the functionality to attest information for entities in the Gaia-X ecosystem with verifiable credentials. This comprises attestations for assets, principals, participants and also covers the signature process for self-descriptions and service-offerings.

4.5.2. Functional Requirements

<i>Endpoints</i>
 IDM.OCM.00021 Create DID
 IDM.OCM.00022 Credential Issue Endpoint
 IDM.OCM.00023 Credential-Proposal Endpoint
 IDM.OCM.00024 Issue Credential Status Endpoint
 IDM.OCM.00025 Credential Status Subscribe Endpoint
 IDM.OCM.00026 Get Credentials
 IDM.OCM.00027 Create Schema Endpoint
 IDM.OCM.00028 Update Schema Endpoint
 IDM.OCM.00029 Get DIDs for Schema Endpoint
<i>Functions</i>
 IDM.OCM.00076 Revocation Support
 IDM.OCM.00077 Establish Secure Connection
 IDM.OCM.00078 Credential-Issue Request Received
 IDM.OCM.00079 Create Credential-Proposal Request
 IDM.OCM.00080 Create Schema
 IDM.OCM.00081 Update Schema
 IDM.OCM.00082 Create Credential-Definition
 IDM.OCM.00083 Verifiable Credential Definition
 IDM.OCM.00084 Credential Validity

 IDM.OCM.00085 Check Credential State
 IDM.OCM.00086 Update Attestation Subscriber
 IDM.OCM.00087 Get Issue-List (DIDs) for Schema Schema

Table 8: Functional Requirements Attestation Manager

4.6. SSI Abstraction Service

4.6.1. Description and Priority

The SSI Abstraction Service is a service that provides all of the SSI functionality required by the other components. This includes providing endpoints for creating/accepting connection invitations, issuing/accepting verifiable credentials, creating/verifying verifiable presentations (proofs). Another aspect is the handling of events for ongoing processes and forwarding to the corresponding service, e.g., forwarding incoming events regarding connections to the connection manager.

For iteration 1, this component SHOULD be closely coupled to the Hyperledger aries framework and MUST implement the Aries Interop Profile 1.0 (AIP 1.0)²⁰ for interoperability between different aries based clients. The functionality for later iteration SHOULD follow the ongoing standardization of the DIDComm protocol and interoperability profiles.

4.6.2 Functional Requirements

Endpoints
 IDM.OCM.00015 DIDcomm Endpoint (externally exposed)
 IDM.OCM.00016 SSI Event Handling
 IDM.OCM.00017 SSI Admin Interface
Functions
 IDM.OCM.00092 Core SSI functionality

Table 9: Functional Requirements SSI Abstraction Service

4.7. Profile Manager

4.7.1. Description and Priority

The Public Profile is an endpoint listed in the DID Document of the Participant. It returns a JSON-LD Verifiable Presentation that contains various Verifiable Credentials - those VC, the Participant marked as publicly

²⁰<https://github.com/hyperledger/aries-rfcs/blob/8526600048b88115b5aaa6d9b4b0120bc4e69bae/concepts/0302-aries-interop-profile/README.md>

available. The Self-Description served by the Public Profile can be extended through additional JSON-LD contexts.

A Private / Permissioned Endpoint to serve Self-Description to authenticated requests is out of scope for iteration 1 and targeted for future releases.

4.7.2. Functional Requirements

<i>Endpoints</i>
 IDM.OCM.00030 Create Public Profile Endpoint
 IDM.OCM.00031 Update Public Profile Endpoint
 IDM.OCM.00032 Delete Public Profile Endpoint
 IDM.OCM.00033 Update Self Description
 IDM.OCM.00034 Serve Public Profile
 IDM.OCM.00035 Unserve Public Profile
 IDM.OCM.00036 CRUD Service-Offering Endpoints
 IDM.OCM.00037 Publish Service-Offering Endpoint
 IDM.OCM.00038 Un-Publish Service-Offering Endpoint
<i>Functions</i>
 IDM.OCM.00093 Credential Transformation
 IDM.OCM.00094 CredDefProofType
 IDM.OCM.00095 Self Description Output
 IDM.OCM.00096 Self Description Content
 IDM.OCM.00097 Create Service-Offering CRUD
 IDM.OCM.00098 Service Offering Export

Table 10: Functional Requirements Profile Manager

5. Other Requirements

-  IDM.OCM.00148 **Publish new DID**

In context of Hyperledger Indy a Participant MUST have a specific role (TRUST-ANCHOR²¹) defined on the identity ledger to be able to issue credentials, schemas, credential definitions and publish a did-document. The relevant NYM transaction MUST be sent to the ledger by AISBL through the onboarding process for a new onboarding Participant. ☐

6. Verification

☐ IDM.OCM.00149 **Behavior Driven Design**
Verification of fulfillment of the requirements and characteristics MUST be done using automated tests which are part of the deliverables. They SHOULD be done by patterns of the [Behavior Driven Development \(BDD\)](#) using the “Gherkin Syntax”. ☐

☐ IDM.OCM.00150 **Automated Test Environment**
All functionalities MUST be demonstrated in a complex test environment within a sandbox, with the following infrastructure components:

- Load Balancer, e.g., HAProxy
- API Gateway, e.g., Kong
- Service Mesh, e.g., Linkerd/Istio
- DNS
- Multiple Servers
- Firewalls

All security tests MUST be passed in this test environment automatically. ☐

☐ IDM.OCM.00151 **Load Tests**
Scalability and Performance around the high workload scenarios MUST be demonstrated, by using any kind of Load Test Framework for HTTP APIs. e.g., Gatling.²² ☐

²¹ https://github.com/hyperledger/indy-node/blob/master/docs/source/auth_rules.md

²² <https://gatling.io/>

Appendix A: Glossary

For the glossary refer to IDM.AO Glossary/Terminology [\[IDM.AO\]](#)

Appendix 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.

All together the deliverables of the first GXFS project phase are specifications for 17 lots, that are being awarded in EU-wide tenders:



Further general information on the Federation Services can be found in [\[TAD\]](#).