Software Requirements Specification

for

Federation Architecture for Composed Infrastructure Services (FACIS)

Digital Contracting Service FACIS.DCS

Published by

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

Table of Contents

1 Introduction 1
1.1 Document Purpose 1
1.2 Product Scope 1
1.3 Definitions, Acronyms and Abbreviations 3
1.4 References 7
1.5 Document Overview 10
2 Product Overview 10
2.1 Product Perspective 10
2.2 Product Functions 12
2.2.1 Template Repository 13
2.2.2 Contract Workflow Engine 14
2.2.3 Signature Management 15
2.2.4 Contract Storage and Archive 16
2.2.5 Process Audit and Compliance Management 17
2.2.6 DCS-to-DCS Communication 17
2.3 Product Constraints 18
2.4 User Classes and Characteristics 19
2.5 Operating Environment 20
2.6 User Documentation 21
2.7 Assumptions and Dependencies 21
2.8 Apportioning of Requirements 22
3 Requirements 22
3.1 Interfaces 22
3.1.1 User Interfaces 22
3.1.2 Hardware Interfaces 39
3.1.3 Software Interfaces 39
3.1.4 Communications Interfaces 40
3.2 Functional Requirements 41
3.2.1 Template Repository 41
3.2.2 Contract Workflow Engine 44
3.2.3 Signature Management 47
3.2.4 Contract Storage and Archive 49
3.2.5 Process Audit & Compliance 51
3.3 Other Nonfunctional Requirements 52
3.3.1 Performance Requirements 52
3.3.2 Safety Requirements 52
3.3.3 Security Requirements 53
3.3.4 Software Quality Attributes 55
3.4 Business Rules 56
3.5 Compliance 57
3.6 Design and Implementation 57
3.6.1 Installation 57
3.6.2 Distribution 57
3.6.3 Maintainability 57
3.6.4 Reusability 57
3.6.5 Portability 58
3.6.6 Cost 58
3.6.7 Deadline 58
3.6.8 Proof of Concept 58
4 System Features 58
4.1 UC-01 User Authentication & Authorization 58
4.2 UC-02 Contract Template Management 59
4.3 UC-03 Contract Creation 61

4.4 UC-04 Contract Signing 62
4.5 UC-05 Contract Deployment 63
4.6 UC-06 Contract Lifecycle Management 63
4.7 UC-07 Contract Storage and Security 64
4.8 UC-08 Contract Compliance & Auditing 65
4.9 UC-09 DCS Administration 66
4.10 UC-10 Contract Automation & Integration 67
4.11 UC-11 API & System Integrations 67
4.12 UC-12 System-Based Contract Management 68
4.13 UC-13 External System Contract Execution 69
4.14 UC-14 Identity and PoA Credential Acquisition 69
4.15 UC-15 Access Rights Revocation 69
5 Other Requirements 70
6 Verification 72
7 Appendix 76

1 Introduction

This document provides the technical specifications of the Digital Contracting Service (DCS), a component developed within the "Federation Architecture for Composed Infrastructure Services" (FACIS) project to support contract lifecycle management in federated cloud and edge service environments.

1.1 Document Purpose

This document specifies the requirements for version 1.0 of the Digital Contracting Service. It serves as a formal specification that outlines the system's functional and non-functional requirements, as well as its technical constraints, architectural context, and verification methods. The primary objective of the DCS is to enable secure, structured, and standards-based digital contracting processes that are interoperable across decentralized infrastructure and compliant with relevant legal and regulatory frameworks.

This specification is prepared in the context of tendering. The intended audience is potential contractors able to deliver an open-source solution that combines an intuitive front-end user interface with back-end capabilities to support core scenarios such as contract creation, structured negotiation, secure digital signing, long-term archiving, and automated compliance monitoring, in alignment with relevant European legal and technical standards.

1.2 Product Scope

The product scope covers the functionalities of the DCS, a modular, standards-based platform for secure, auditable, and legally compliant digital contract lifecycle management in federated cloud and edge environments. The product supports both bilateral and multilateral contracting use cases and is developed to interoperate seamlessly with emerging European trust infrastructures. The DCS as a product will form the foundation for extending the Federation Architecture Pattern for Digital Contracting, as part of the FACIS strategy.

A contract is a legally binding agreement between two or more parties with resulting rights and obligations. Digital contracts are digital agreements between transacting parties that are written in computer code [Ref-1], and a service for digital contracts consists of programs that implement and enforce the execution of a contract [Ref-2]. They rely on electronic signatures for authentication and verification. In legal terms, the conclusion of a contract occurs when two or more parties reach a binding agreement through mutual declarations of intent – specifically, an offer and a corresponding acceptance. An offer is a clear and definite proposal made with the intention of being legally bound if accepted, while acceptance is an unconditional agreement to the exact terms of the offer. Both declarations must be communicated and reflect a shared intention to create legal relations.

The scope of the specification initially targets business-to-business (B2B) contacting scenarios rather than business-to-consumer (B2C). To be more specific, in FACIS a work on SLA contracts is being carried out to define high-level legal requirements for both human-readable and machine-readable contract formats, along with their data processing and conversion mechanisms. B2C support might be considered as an optional functionality.

In the DCS context, a digital contract is a contract expressed, managed, and executed in digital form, existing in both machine-readable and human-readable formats. Each contract is represented as a contract object, a digital instance with a defined lifecycle, semantic conditions, and states such as offered, accepted, rejected, and withdrawal. The contract metadata contains structured information such as title, version, creation date, governing law, and unique identifiers, ensuring traceability. Contract parties include all entities involved in the agreement, with their roles, identifiers, and contact details explicitly defined. The agreement content is

organized into contract components, which may include the main contract text, annexes, and attachments, each with its own precedence rules. The scope of work is described under service scope, accompanied by acceptance criteria defining the evidence and conditions required for formal approval. Where applicable, licensing Information outlines the usage rights for software, data, or intellectual property. The contract also incorporates contract conditions, which are specific clauses, obligations, or regulatory requirements, and semantic conditions, which are machine-readable clauses enabling automated validation. Additional agreements & clauses, such as confidentiality or liability limitations, may also be included. Signature Information records the signatories, their signature types, timestamps, and legal evidence to ensure authenticity and enforceability.

A contract scenario in the DCS defines a legally binding agreement based on approved templates that embed usage policies and constraints. To give an example of such a scenario, a cultural institution could create a contract template to regulate access to its data API. The contract specifies geospatial and temporal restrictions (e.g., access limited to German organizations, with an expiry date). Usage policies are expressed as machine-readable semantic conditions that DCS validates at approval/signing and deploys to the Contract Target System (e.g., API gateway) for automated runtime enforcement. An interested organization negotiates and reviews the contract, and after mutual signing and verification of credentials, access to the API is granted under the agreed terms. Lifecycle management governs state transitions (offered → accepted → executed → active → terminated → archived), triggers renewal and expiry alerts, and revokes access if credentials are invalidated or terms are breached. Executed contracts, signatures, and validation artifacts are preserved with verifiable timestamps in a tamper-evident archive, ensuring auditability and legal traceability. A JSON-LD contract example is provided in the Appendix.

Key functional capabilities within the product scope are as follows:
- Multi-Contract Signing: Enables multi-party contract execution within a single integrated workflow.
- Automated Workflows: Automates contract generation, execution, and deployment to ensure legal consistency and efficiency.
- Lifecycle Management: Monitors contracts with alerts for renewals, expirations, or required actions.
- Signature Management: Links contract signatures to verifiable digital identities to maintain legal validity and trust.
- Secure Archiving: Stores signed contracts in a tamper-evident archive compliant with retention policies.
- Machine Signing: Supports automated signing for high-volume or routine transactions.

Digital contracts rely on electronic signatures for authentication and verification. Regarding signatures, it should be noted that digital signatures and seals must be qualified to sign legally valid contracts. Qualified electronic signatures (QES) must be traceable to a natural person. Qualified electronic seals are their equivalent for organizations. JAdES, PAdES, and XAdES are specifications that must be complied with when creating so-called advanced signatures on JSON, PDF, and XML documents.

The product scope focuses on the use of Advanced Electronic Signatures (AES) for authentication, allowing for the descope QES and the integration of remote signing services and Trust Service Providers (TSP). Should the product move to production, it will be the responsibility of the integrating party to connect to remote signing services and TSPs. Specifically, the company will sign DCS usage contract via OID4VP, with the user signing it using their AES PID. This contract is then stored and referenced through a role, along with the signed contracts.

The architecture SHOULD include an optional module to integrate a remote QES signing services of a Trust Service Provider (e.g., D-Trust) with the “Signature Management” Module of the functional architecture. This integration can be considered “optional.”

DCS provides both an intuitive web interface and back-end services, integrating with XFSC components. For this reason, the product extension must include interfaces (APIs) to integrate the DCS smoothly with XFSC

components and identity wallets. Building and operating these XFSC components are considered out of the product scope.

Regarding policies, DCS considers only two types of policies: namely DCS-to-DCS data exchange and access control for users to access DCS. In DCS-to-DCS data exchange, DCS is limited to offering the data information endpoint that provides the relevant contract information to be used for policy enforcement. For authorization and authentication of users to act with the defined roles in the system, the company or organization using DCS is expected to register employees or organization members with the user roles to the DCS over identity wallets.

1.3 Definitions, Acronyms and Abbreviations

Acronyms and abbreviations used in this document are defined in Table 1, while Table 2 contains the definitions of key terms essential for interpreting the requirements. Additional or supplementary definitions that provide extended context are collected in Appendix A: Glossary.

Table 1 – List of acronyms and abbreviations

Acronym or Abbreviation	Term	Definition
AES	Advanced Electronic Signature	A type of electronic signature that is uniquely linked to the signer and capable of identifying them, offering a higher level of assurance than simple electronic signatures.
API	Application Programming Interface	A set of rules and protocols for building and interacting with software applications, enabling communication between different systems.
B2B	Business-to-Business	Commercial transactions conducted directly between businesses.
B2C	Business-to-Consumer	Commercial transactions conducted directly between a business and end consumers.
BDD	Behavior-Driven Development	A development approach that defines software behavior in natural language for shared understanding.
C2PA	Coalition for Content Provenance and Authenticity	Standards for attaching provenance metadata and cryptographic assertions to digital content.
CAT	Federated Catalogue	A service for publishing, discovering, and requesting assets within the federation.
CRUD	Create, Read, Update, and Delete	Four basic operations performed on persistent data in databases or storage systems.
CSA	Contract Storage and Archive	Digital Contracting Service software component for archiving signed contracts in tamper-proof, long-term storage.
CWE	Contract Workflow Engine	Digital Contracting Service software component managing the life cycle and progression of contracts through defined steps.
DCS	Digital Contracting Service	A modular, standards-aligned platform for the secure, verifiable, and automated lifecycle management of digital contracts.
DSS	Digital Signature Service	An open-source library for creating and validating electronic signatures in compliance with European eIDAS regulations, usable in apps, standalone, or server solutions.

DID	Decentralized Identifier	A globally unique identifier enabling verifiable, self-sovereign digital identity.
ECO	ECO – Verband der Internetwirtschaft e.V	The German Association of the Internet Industry, representing companies involved in internet infrastructure, services, content, and applications.
ERP	Enterprise Resource Planning	An integrated business software suite for business operations.
ETSI	European Telecommunications Standards Institute	European standards organization for ICT
EUDI	European Digital Identity	An EU framework for a secure and interoperable digital identity wallet.
FACIS	Federated Architecture for Cloud and Edge Services	An initiative framework for secure and interoperable federation-based service delivery.
FR	Functional Requirement	A specification of a function or behavior the system must perform to meet business objectives.
IPCEI-CIS	Important Project of Common European Interest – Cloud Infrastructure and Services	An EU-driven initiative to foster large-scale, cross-border cloud and infrastructure projects of strategic importance.
JAdES	JSON Advanced Electronic Signatures	An ETSI standard specifying advanced electronic signatures in JSON-based documents.
JSON-LD	JavaScript Object Notation for Linked Data	A JSON-based format to serialize Linked Data, i.e., RDF, enabling integration with semantic web technologies.
NFR	Non-Functional Requirement	A specification of a quality attribute, constraint, or performance criterion that defines how a system must operate rather than what it should do
OCM W	Organizational Credential Manager – Wallet	A wallet component for managing organizational credentials in federated environments.
PACM	Process Audit & Compliance	DCS software component that maintains tamper-proof records and supports compliance and auditing functions.
PAdES	PDF Advanced Electronic Signatures	An ETSI standard for advanced electronic signatures embedded in PDF documents.
PCM	Personal Credential Manager	A wallet component for managing credentials of natural persons.
PoA	Power of Attorney	A credential granting the holder authority to act on behalf of another party.
PR	Participant	A legal entity officially onboarded to a Federation, capable of taking on roles such as Provider, Consumer, or Federator.
RBAC	Role-Based Access Control	A security model that restricts system access based on users' roles and assigned permissions.
RDF	Resource Description Framework	W3C standard model for data interchange on the web.
QES	Qualified Electronic Signature	The highest standard of electronic signature under eIDAS, with the same legal standing as a handwritten signature.
REST	Representational State Transfer	An architectural style for distributed hypermedia systems, often used in web service APIs.

SES	Simple Electronic Signature	An electronic form of a signature, such as typing a name in an email, used to sign or associate with electronic data.
SLA	Service Level Agreement	A formal contract between a service provider and a customer defining performance and service standards.
SHACL	Shapes Constraint Language	A W3C standard for validating RDF data against a set of conditions (shapes).
SM	Signature Management	A component responsible for applying, verifying, and managing electronic signatures on contracts.
TR	Template Repository	DCS software component for storing and managing contract templates in both machine-readable and human-readable formats.
TSP	Trust Service Provider	An entity that provides and manages digital certificates and related trust services under eIDAS.
UI	User Interface	Visual and interactive elements through which a user interacts with the system.
UUID	Universally Unique Identifier	A 128-bit identifier used to uniquely identify information in computer systems.
VC	Verifiable Credential	A cryptographically verifiable statement issued about a subject, following the W3C standard.
W3C	World Wide Web Consortium	The main international standards organization for the World Wide Web.
XFSC	Cross Federation Services Components	A set of modular components enabling federation-level integration, orchestration, and interoperability.

Table 2 – List of terms and definitions

Term	**Definition**	**Link**
Acceptance	An unconditional agreement to the exact terms of an offer, communicated between parties with the intent of creating legal relations.	See section 1.2
Additional Agreements & Clauses	Supplementary legal provisions included in a contract, such as confidentiality terms, liability limitations, and penalties for non-compliance.	See section 1.2
Audit-Proof Storage	Storage method ensuring contracts cannot be altered or deleted without detection, with verifiable proof of integrity over time.	See section 2.2.4
Audit Trail	An immutable record of significant actions taken during the contract lifecycle, including timestamps, user or system role, and action details.	See section 3.2.5
Compliance Check	Verification mechanism that a contract meets defined legal, regulatory, or policy requirements.	See section 3.2.5
Contract	A legally binding agreement between two or more parties with enforceable rights and obligations, containing an offer, acceptance, mutual intent, legal capacity, lawful purpose, and where applicable, consideration.	See section 1.2
Contract Adjustment	Adding, removing, and modification of specific clauses, terms, or data points in an existing contract during negotiation phase between the parties. When an adjustment is accepted by both parties, it creates a new version of the contract under the same contract ID, and the system re-renders the human-readable view to ensure consistency across formats.	See section 2.2.2

Contract Assembly	The process of combining selected metadata and components from the template with additional agreements & clauses into a complete contract document.	See section 3.2.2
Contract Change Request	A formal, version-controlled request to modify an already-created or active contract.	See section 2.2.2
Contract Conditions	Specific clauses, terms, or requirements that form part of a contract's content and may be negotiated before signing.	See section 1.2
Contract Deployment	The release of a finalized, machine-readable contract to designated service endpoints for automated execution.	See section 2.2.2
Contract Identifier	A unique and persistent reference assigned to each contract for tracking and retrieval across systems.	See section 2.2.4
Contract Initiation	The process of starting contract creation by submitting a request, identifying the initiating and responding parties, and selecting a matching template.	See section 2.2.2
Contract Lifecycle	The complete set of stages from contract creation through negotiation, approval, signing, performance tracking, renewal or termination, and archival.	See section 2.1
Contract Negotiation	A collaborative process where involved parties propose, review, and agree on changes to a draft contract before finalization.	See section 3.2.2
Contract Object	A digital representation of a contractual agreement content with a defined lifecycle, semantic conditions, and states (e.g., offered, accepted, rejected, withdrawal).	See section 1.2
Contract Performance Monitoring	The tracking and assessment of contractual obligations against agreed metrics and deadlines throughout the contract lifecycle.	See section 4.6
Contract Renewal	The continuation of an existing contract's validity through an agreed extension before its expiration date. Contract adjustments MAY be made during contract renewal phase.	See section 4.6
Contract Request	The submission that starts contract creation, identifying parties and the contract template to use.	See section 2.2.2.
Contract Target System	An external system that receives and executes deployed contracts.	See section 4.5
Contract Versioning	Contract versioning is the practice of creating and managing uniquely identified, immutable records of each draft, revision, and executed form of a contract, ensuring a clear history of changes, traceable authorship, and verifiable integrity over time.	See section 2.2.4
Contract Termination	The structured process of ending a contract, either by mutual or multi-party agreement or predefined conditions, including archival.	See section 4.6
Digital Contract	A contract expressed, managed, and executed in digital form, existing either in both machine-readable or human-readable formats, or both.	See section 1.2
Electronic Signature	Data in electronic form attached to or associated with a document to sign it.	See section 1.2
Human-Readable Contract	A natural language version of a contract intended for human interpretation.	See section 1.2
Identity Verification	The process of confirming the identity of contracting parties, possibly via digital wallets, DSS, or other secure methods.	See section 2.2.3
Initiator	The party that submits the initial contract request in a bilateral or multilateral workflow.	See section 2.2.2

Immutable Signing	The property that once a contract has been signed by all parties, its content cannot be altered without invalidating the signatures.	See section 3.2
Legal Capacity	The ability of a party to enter into a binding contract, typically requiring legal age and mental competence.	See section 1.2
Lifecycle States	The defined progression of a contract object, including at least the states: offered, accepted, rejected, and withdrawal.	See section 2.2.2
Machine-Readable Contract	A structured, computer-processable representation of a contract.	See section 1.2
Multilateral Contract	A contract involving more than two parties.	See section 1.2
Offer	A clear, definite proposal to enter into a contract, made with intent to be legally bound if accepted.	See section 1.2
Policy	A formal, machine-enforceable set of rules and constraints that govern the behavior of the DCS. The scope of policies is limited to DCS-to-DCS connection and user access to DCS.	See section 1.2
Provenance Tracking	Recording the origin, history, and changes to a document or template to ensure authenticity and traceability.	See section 2.2.1
Responder	The party (or parties) that receive a contract request in a bilateral or multilateral workflow.	See section 2.2.2
Revocation	The process of invalidating credentials or signatures to prevent further use.	See section 4.15
Semantic Conditions	Machine-readable conditions or clauses within a contract that can be processed, validated, and tracked automatically.	See section 1.2

1.4 References

Table 3 – List of references

Reference ID	Description	Link
[AES]	Advanced Electronic Signature (eIDAS).	https://eur-lex.europa.eu/eli/reg/2014/910/oj
[Animo]	Animo Solutions framework for SSI wallet and credential integration.	https://animo.id
[ARF]	EU Digital Identity Wallet Architecture and Reference Framework (ARF).	https://digital-strategy.ec.europa.eu/en/library/european-digital-identity-wallet-architecture-and-reference-framework
[ArgoCD]	GitOps continuous delivery for Kubernetes.	https://argo-cd.readthedocs.io/en/stable/
[BDD Executor]	XFSC bdd-executor – Framework to define and run executable Behavior-driven development scenarios	https://github.com/eclipse-xfsc/bdd-executor
[BSI]	German Federal Office for Information Security (crypto guidance).	https://www.bsi.bund.de/EN

[CAdES]	CMS Advanced Electronic Signatures.	https://en.wikipedia.org/wiki/CAdES_(computing)
[CAT.AD]	Architecture of XFSC Catalogue.	https://gaia-x.eu https://gaia-x.gitlab.io/data-infrastructure-federation-services/cat/architecture-document/architecture/catalogue-architecture.html
[DID]	W3C Decentralized Identifiers v1.0.	https://www.w3.org/TR/did-1.0/
[DIDcomm]	DIDComm Messaging v2.	https://identity.foundation/didcomm-messaging/spec/v2/
[Docker]	Container runtime / tooling.	https://www.docker.com
[EUDI]	European Digital Identity Wallet framework.	https://digital-strategy.ec.europa.eu/en/policies/eudi-wallet
[Gaia X]	Federation / data space initiative.	https://gaia-x.eu
[GDPR]	EU General Data Protection Regulation.	https://eur-lex.europa.eu/eli/reg/2016/679/oj
[GitHub Actions]	CI/CD service.	https://docs.github.com/actions
[Helm]	Kubernetes package manager.	https://helm.sh
[IPCEI CIS]	Important Project of Common European Interest – Cloud Infrastructure & Services.	https://digital-strategy.ec.europa.eu/en/policies/ipcei-cloud
[ISO/IEC 27001]	Information security management.	https://www.iso.org/standard/27001.html
[JAdES]	ETSI JSON Advanced Electronic Signatures.	https://www.etsi.org/committee/esi
[JSON LD]	JSON for Linked Data 1.1.	https://www.w3.org/TR/json-ld11/
[Kubernetes]	Container orchestration.	https://kubernetes.io
[NATS]	High performance messaging.	https://nats.io
[NodeRED]	Flow-based development tool for visual programming.	https://nodered.org
[OAuth2]	OAuth 2.0 authorization framework.	https://www.rfc-editor.org/rfc/rfc6749
[OIDC]	OpenID Connect Core 1.0.	https://openid.net/specs/openid-connect-core-1_0.html
[OpenID4VC]	OpenID for Verifiable Credential Issuance (OID4VCI) 1.0.	https://openid.net/specs/openid-4-verifiable-credential-issuance-1_0.html
[OpenID4VP]	OpenID for Verifiable Presentations (OID4VP) 1.0.	https://openid.net/specs/openid-4-verifiable-presentations-1_0.html
[Ory/Hydra]	Open source OAuth2/OIDC server	https://www.ory.sh/hydra/docs/

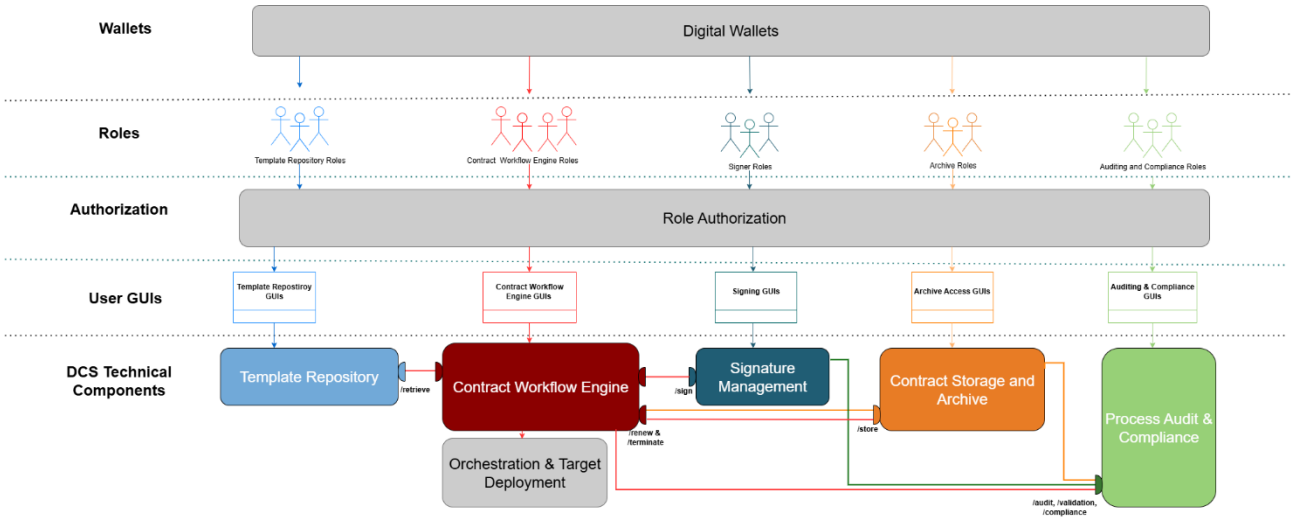
	used for secure access and authentication.	
[PAdES]	PDF Advanced Electronic Signatures.	https://en.wikipedia.org/wiki/PAdES
[PDF/A-3]	Archival PDF (ISO 19005 3).	https://www.iso.org/standard/54534.html
[PoA]	Power of Attorney credential chain.	https://en.wikipedia.org/wiki/Power_of_attorney
[PostgreSQL]	Relational database (Postgres compatible).	https://www.postgresql.org
[QES]	Qualified Electronic Signature (eIDAS).	https://eur-lex.europa.eu/eli/reg/2014/910/oj
[REST]	Representational State Transfer (architectural style).	https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf
[RFC 2119]	Keywords to state requirement levels.	https://www.rfc-editor.org/rfc/rfc2119
[SCS]	Sovereign Cloud Stack.	https://sovereigncloudstack.org
[SDE.DCS]	Gaia-X Federation Services Sovereign Data Exchange Data Contract Service.	https://www.gxfs.eu/data-contract-service/
[SHACL]	Shapes Constraint Language.	https://www.w3.org/TR/shacl/
[SOG-IS]	Senior Officials Group – Information Systems Security (crypto guidance).	https://www.sogis.eu
[SSI]	Self-Sovereign Identity concept.	https://en.wikipedia.org/wiki/Self-sovereign_identity
[TLS 1.3]	Transport Layer Security v1.3.	https://www.rfc-editor.org/rfc/rfc8446
[TSP]	Trust Service Provider (eIDAS/ETSI).	https://www.etsi.org/technologies/trust-service-providers
[UUID]	Universally Unique Identifier.	https://www.rfc-editor.org/rfc/rfc4122
[VC]	W3C Verifiable Credentials Data Model v2.0.	https://www.w3.org/TR/vc-data-model-2.0/
[WACI]	Wallet & Credential Interaction – Presentation Exchange.	https://identity.foundation/waci-presentation-exchange/
[WCAG]	Web Content Accessibility Guidelines.	https://www.w3.org/TR/WCAG21/
[XFSC]	Eclipse Cross Federation Services Components.	https://projects.eclipse.org/projects/technology.xfsc
[Ref-1]	Law, A. Smart contracts and their application in supply chain management	https://dspace.mit.edu/handle/1721.1/114082

[Ref-2]	Das, A. et al. Resource-Aware Session Types for Digital Contracts	https://doi.org/10.1109/CSF51468.2021.00004

1.5 Document Overview

This document describes the product perspective, functions, and constraints of the Digital Contracting Service. It defines the system features in detail, including both functional and non-functional requirements, and specifies binding requirements for the development and operation of the system. All functional requirements are identified by a unique ID in square brackets following the format [DCS-FR-<Component Code>-<Number>], and all non-functional requirements follow the format [DCS-NFR-<Number>]. Component codes (e.g., TR for Template Repository, CWE for Contract Workflow Engine, SM for Signature Management, CSA for Contract Storage and Archive, PACM for Process Audit & Compliance) correspond to the major subsystems described in this document. Requirements use the keywords MUST, MUST NOT, SHOULD, SHOULD NOT, and MAY as defined in RFC 2119 [RFC 2119].

2 Product Overview



2.1 Product Perspective

Fig.1 – Simplified Digital Contracting Service diagram

FACIS project defines a governance framework and a set of Federation Architecture Patterns (FAPs) that bundle multi-provider Cloud-Edge capabilities into interoperable service clusters with agreed Service-Level Agreements. All artifacts are released as Free and Open-Source Software to speed adoption and ensure cross-domain interoperability. DCS is a modular and standards-aligned software platform developed within the governance framework of FACIS. Its purpose is to enable the secure, verifiable, and automated lifecycle management of digital contracts, with a strong focus on business-to-business (B2B) use cases. It supports structured SLAs as the primary contract type and operates with machine-readable document formats that can be rendered into human readable formats. DCS enables workflows such as contract creation and negotiation, verifiable signing with identity and PoA credentials, SLA monitoring and performance tracking, deployment of machine-readable contracts to external systems, and secure storage, audit logging, and compliance reporting. These scenarios reflect the core product perspective of DCS. The product reuses and builds upon open standards such as W3C Verifiable Credentials (VCs), JSON-LD, SHACL, PAdES/JAdES, and

UUID/DID identifier schemes, and the system has the capability to interface with external systems via RESTful APIs.

DCS encompasses a set of functional components that work together to provide end-to-end digital contract handling. These include a template repository for managing contract templates in machine-readable and human-readable formats; a contract workflow engine that drives lifecycle automation and ensures consistency across representations; a signature management module that ensures legally valid and verifiable electronic signatures with support for role-based signing; a contract storage and archival system that guarantees secure retention of signed agreements; and a compliance and audit module that maintains tamper-proof records and supports regulatory reporting. Fig. 1 illustrates the major DCS components marked with colors. The user roles categorized for each technical component in Fig.1 are defined in Section 2.4. Furthermore, DCS integrates and interfaces with the XFSC components, which are listed as follows:

- XFSC Catalogue: Template repository features defined in DCS are aligned with XFSC Catalogue functionalities and discovering and requesting templates via the Catalogue MUST be made available. The link to the XFSC Catalogue Architecture document is provided in Appendix F.
- XFSC's OCM W-Stack as the digital wallet for organizations. OCM W-Stack provides OpenID for Verifiable Credential Issuance (OpenID4VCI) functionalities and is provided as an example in Appendix D.
- A digital wallet for natural persons supporting OpenID for Verifiable Credential Issuance (OpenID4VCI) and OpenID for Verifiable Presentations (OpenID4VP) as profiled by the Architecture and Reference Framework (ARF). The selection of a specific wallet implementation is out of scope; any chosen wallet MUST demonstrate ARF compliance.
- XFSC Orchestration Engine: Contract Workflow Engine features defined in DCS are aligned with XFSC Orchestration Engine. The deployment of the XFSC Orchestration Engine is not required as the engine can be hosted by ECO. Since XFSC Orchestration Engine is based on Node-RED, all DCS components MUST expose Node-RED–compatible interfaces to enable workflow execution via the orchestration engine. XFSC orchestration documentation is provided as an Appendix to the specification and training will be provided by ECO for orchestration engine when requested.
- Revocation List: The contractors are advised to use XFSC-compatible solutions for the revocation features of digital wallets. XFSC's Status List Service is part of OCM W-Stack, and can be used stand alone for creating status lists for revocation. An example is provided in Appendix D and E.
- SD-JWT: XFSC's SD-JWT service is a micro service for creating SD-JWTs. It is also used by the TSA signing/verification over the Crypto Provider Service, but it can be used as standalone service as well
- Authentication and Authorization Service for user roles: This service is required to log in users with authorization flows for the given roles based on credentials stored in digital wallets. Ory/Hydra is an open-source OAuth2 and OpenID Connect (OIDC) server and is recommended as a solution used for authorization flows and role authentication with digital wallets. However, use of Ory/Hydra is optional.

In addition to this representation, the Reference Architecture of the Digital Contracting Service is shown in Fig. 2. This figure depicts the main components together with their interfaces. Detailed descriptions of the components and their interfaces are provided in the subsequent section.

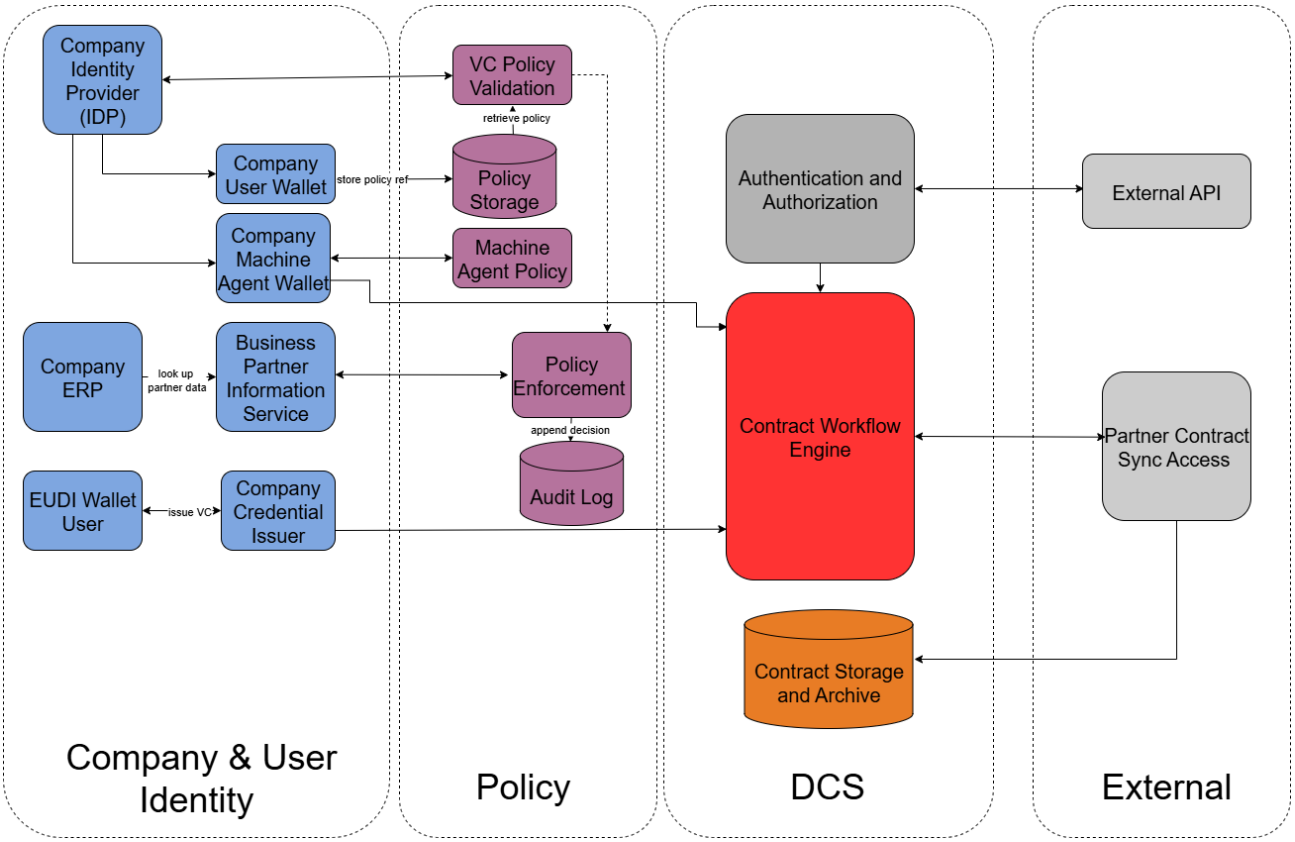


Fig.2 – DCS reference architecture and main flows

2.2 Product Functions

As mentioned in Section 2.1, DCS consists of five key functional components. The functions required in each component are given in Fig. 3 and are listed in this section. In addition, the DCS-to-DCS communication pattern is illustrated in Fig. 4 as a separate product function, showing how two DCS instances exchange contracts, state updates, and revocations across organizational boundaries.

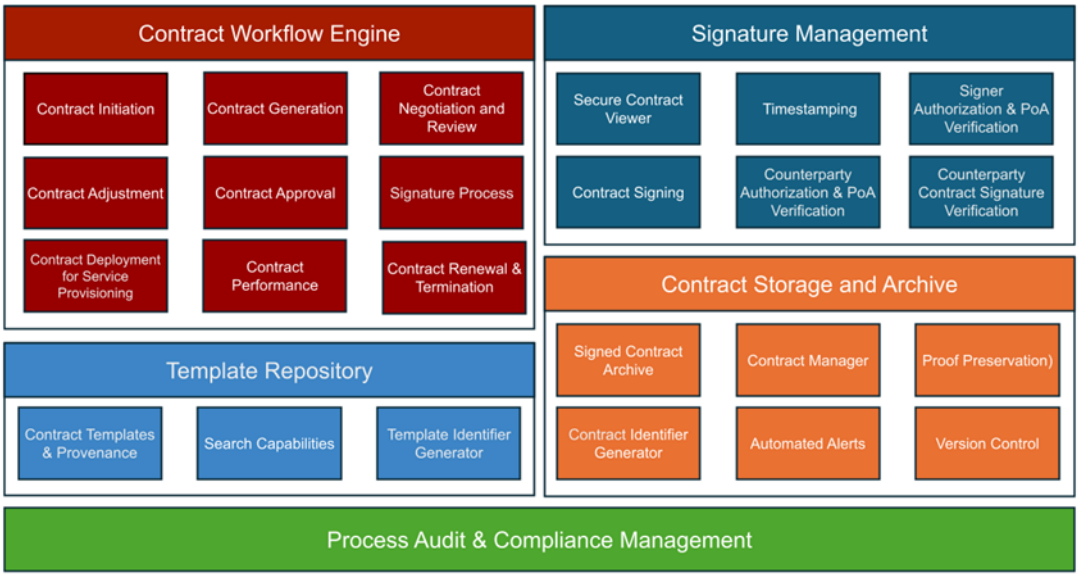


Fig.3 – Conceptual technical components of Digital Contracting Service

2.2.1 Template Repository

Template Repository (TR) facilitates the storage and management of contract templates in machine readable source forms. Below are the key functions of template repository:

- **Contract Templates & Provenance:** Repository stores contract templates in machine readable source forms, with the capability of converting them to human-readable formats when requested. The templates are designed to ensure standardized usage in the service. Template provenance means tracking who or which system role uploaded or modified a template to maintain regulatory alignment. The repository MUST maintain provenance tracking for all contract templates, recording their creation, modifications, approvals, and historical versions. All tracking data MAY be logged into an external system.
- **Search Capabilities:** Includes advanced search features that allow users to efficiently locate templates based on metadata, categories, or specific keywords. Repositories can also support clause-specific searches within internal templates, enhancing precision and usability.
- **Template Identifier Generator:** A generator that assigns globally UUIDs or DIDs to templates, ensuring they are universally identifiable and traceable across contract workflows.

The interfaces required for the Template Repository can be listed as follows:

- **TR → CWE:** The Template Repository exposes retrieval and verification endpoints so the Workflow Engine can select an **approved** template and verify its integrity before contract creation.
- **TR ↔ PACM:** Every template action (create, modify, approve, delete) emits an auditable event to Process Audit & Compliance Management.
- **TR → Users (REST/UI):** The repository provides CRUD, search, verify, and approval endpoints and UIs for managers/reviewers to manage templates under Role-Based Access Control (RBAC). The list of user interfaces and their requirements are given in Section 3.1.1.

XFSC Federated Catalogue (CAT) SHOULD be used as the component to build these functionalities for the template repository, and to adhere to the functional requirements listed in Section 3.2. CAT will first be used

to implement a Minimum Viable Product (MVP) template repository with limited incremental changes. This version contains the following features: Templates are stored in machine-readable form within an RDF/SHACL-centric model. Validation occurs at ingestion, which includes syntactic RDF/JSON-LD parsing and semantic integrity to registered identifiers. In addition, references to human-readable renderings are maintained. Metadata and full-text indexing support search and retrieval. An identifier mechanism issues UUIDs or DIDs for templates and versions. RBAC governs access to objects and lifecycle operations. Provenance is recorded for creation, modification, approval and publication events and may be written to an external audit facility. Collectively, these capabilities allow the repository to store, search, retrieve, verify, version and govern templates with fine-grained access control.

Planned platform extensions focus on interoperability and governance. The repository SHOULD maintain durable links between machine-readable sources and human-authored artefacts, with cryptographic checksums for integrity. A dependency model SHOULD capture includes/extends/requires relations among templates and schemas. A unified export SHOULD package versioned templates together with dependencies and artefacts for consumption by external systems. Workflow support SHOULD cover multi-party review and approval with defined states, assignments and comments, exposed through administrative and steward user interfaces that provide search, diff and lifecycle controls. Provenance SHOULD be expanded into a graph that relates contributors, approvals, artefact hashes and upstream dependencies. An optional subscription mechanism may notify consumers of relevant changes, including impact analysis across dependency trees where feasible. Where non-RDF structures (e.g. JSON Schema) are required, adapters SHOULD permit side-by-side storage while preserving CAT as the canonical source.

2.2.2 Contract Workflow Engine

Contract Workflow Engine automates the contract lifecycle, from initiation to execution. It is responsible for managing the contract lifecycle according to best practices. It provides a set of features required for seamless digital contracting. The functions of the workflow are given below:

- **Contract Initiation:** This feature allows participants to initiate the contract creation process by submitting a contract request. The party that submitted the request is the Initiator and the party that received the request is the Responder in bilateral contracts. Multilateral contracts have one Initiator, but they can have multiple Responders. The Initiator is then allowed to select an appropriate matching template of the machine-readable source reference which is converted to an identical human-readable contract format. The source reference and the human-readable result MUST be linked to each other with the same Template UUID, and after this point in the workflow, all changes MUST occur identically in both formats.
- **Contract Generation:** This feature dynamically populates the selected contract templates with the metadata provided by the Initiator during the contract initiation phase and then fills in the necessary placeholders in the contract template. At the end of this step, the filled-out contract MUST be ready to be sent to the Responder or Responders to start the next phase in the workflow.
- **Contract Negotiation and Review:** This workflow starts with the first review of the Responder or Responders to the contract generated by the Initiator. After this review, Responders have the option to accept, negotiate, or refuse the contract. If Responders choose to negotiate the contract request, negotiation phase starts for collaborative editing and review of the contract. Multiple stakeholders can suggest and review changes. It provides version control to track edits and contains review workflows for finalizing contract review phase, sending confirmation to all parties on successful completion. Negotiation includes a dedicated Contract Adjustment sub-workflow for clause-level edits; each accepted adjustment produces a contract version under the same UUID/DID.
- **Contract Adjustment:** During contract negotiation phase, parties may apply clause-specific, minor updates (add, remove, or modify clauses, terms, or data fields) to the machine-readable source of record. If an adjustment is agreed by both parties, a new version of the contract is created under the same contract ID, and audit logs of this adjustment event are recorded. For each accepted adjustment, the system re-renders the human-readable document to keep both formats aligned.

- **Contract Approval:** This feature provides a mechanism for stakeholders to approve the finalized contract. It also sends automated notifications and reminders to stakeholders that are part of the approval process.
- **Signature Process:** After approval, the Signature Process ensures the electronic signing of contracts using the features provided by the Signature Management component of the DCS. It manages the structured workflow and role-based responsibilities to ensure a compliant signing process.
- **Contract Deployment for Service Provisioning:** After the contract is finalized and signed, this feature ensures the deployment of machine-readable source reference to service endpoints for execution. This approach ensures that service provisioning aligns with the contractual terms and enables automation via APIs.
- **Contract Performance:** This feature ensures that contractual obligations are monitored, measured, and enforced throughout the contract lifecycle with the support of the Contract Storage and Archive component. Key performance indicators are used to assess whether contractual terms are being met, evaluate contract performance, and optimize future agreements. Additionally, the Automated Alerts feature of the Contract Storage and Archive component sends notifications for contract renewals, expirations, or key deadlines, ensuring that stakeholders remain informed about critical contract events.
- **Contract Renewal and Termination:** This feature handles both the automated renewal and structured termination of contracts. For renewals, it ensures contracts are renewed based on predefined terms, sending alerts and notifications to stakeholders about upcoming deadlines, leveraging the timestamping feature of the Signature Management component. For terminations, it supports the proper closure of contracts through final reviews and archival in the Contract Storage and Archive component.

The interfaces required for the Contract Workflow Engine (CWE) can be listed as follows:

- **CWE ← TR:** The Workflow Engine retrieves approved templates and their metadata to assemble and initialize contract instances with synchronized machine- and human-readable forms.
- **CWE ↔ SM:** After approval, the Workflow Engine invokes Signature Management to run the structured signing process and track signature progress.
- **CWE → CSA:** When execution completes, the Workflow Engine triggers archival of the finalized contract and evidence in Contract Storage & Archive.
- **CWE ↔ PACM:** All lifecycle actions (creation, negotiation, approval, state transitions) are logged as immutable audit events for compliance reporting.
- **CWE ↔ Policy/Access Controls:** Role-based access and policy checks gate workflow actions for creators, reviewers, approvers, and signers.

2.2.3 Signature Management

The Signature Management (SM) component is responsible for ensuring secure, compliant, and verifiable signatures on contracts. This component plays a pivotal role in safeguarding the validity of contracts by adhering to standards such as the European Union's eIDAS regulations. The following functions define the component:

- **Secure Contract Viewer:** This feature ensures that contracts can only be accessed in a secure and controlled environment. It prevents unauthorized access and provides a protected interface for reviewing contract details prior to signing.
- **Timestamping:** This feature facilitates the logging of critical time-bound actions, such as setting deadlines, sending reminders, and recording the exact time of signing.
- **Signer Authorization & Power of Attorney (PoA) Credential Chain Verification:** This feature validates the authorization of individuals signing the contract, ensuring they have the necessary authority to act on behalf of their organizations. If a PoA exists, the system performs a credential chain verification using a PoA Trust Service to validate the PoA credentials. This ensures that each signature is legally

binding and represents an authorized party, regardless of whether authorization is direct or via a PoA.

- **Contract Signing:** This feature manages the actual electronic signing process. It utilizes secure digital signature technologies to ensure that each signature is tamper-evident and legally valid.
- **Counterparty Authorization & PoA Credential Chain Verification:** This feature ensures that the counterparty involved in the contract has the required authority and authenticity to sign. It verifies the credentials, authenticates the identity, and confirms the proper authorization of the counterparty to act on behalf of their organization. In cases where a PoA exists, a PoA Trust Service is used to perform credential chain verification to validate the PoA credentials. This process guarantees compliance and trust, regardless of the form of authorization.
- **Counterparty Contract Signature Verification:** This feature verifies the validity of the signature provided by the counterparty, ensuring it is authentic, valid at the time of signing, and complies with regulatory standards. It cross-checks the cryptographic integrity of the digital signature to confirm that the signed document has not been tampered with after signing.

The interfaces required for Signature Management can be listed as follows:

- **SM ← CWE:** Signature Management receives an approved contract and its envelope from the Workflow Engine to verify content, apply signatures, and validate results.
- **SM ↔ Identity/Credential Services:** Signer authorization and PoA-credential chain verification are performed against trusted anchors before a signature is accepted.
- **SM → CSA:** Upon successful signing, the executed contract and all validation artifacts are stored in the archive for long-term retention.
- **SM ↔ PACM:** Retrieval, verification, signing, validation, revocation, and compliance-check events are appended to the audit trail.
- **SM → External Clients (REST/UI):** Signature APIs and viewers provide secure retrieval, verify/apply/validate/revoke operations, and compliance reporting to authorized users and systems.

2.2.4 Contract Storage and Archive

The Contract Storage and Archive (CSA) component is deployed for each ecosystem participant, ensuring that their signed contracts are managed, stored, and organized securely. This approach allows each participant to maintain control over their contract management. Its core functions are detailed below:

- **Signed Contract Archive:** This feature maintains a secure repository for all signed contracts in both machine-readable source forms and their human readable end forms, ensuring they are easily accessible while protecting their confidentiality and integrity.
- **Contract Manager:** This feature enables users to access and control other features within the Contract Storage and Archive component. It provides an interface for interacting with functionalities such as signed contract retrieval, proof preservation, automated alerts, and version control, enhancing the overall usability of the contract storage system.
- **Proof Preservation:** This feature ensures the preservation of evidence for legal and compliance purposes, maintaining tamper-evident records of all contracts to protect against disputes and audits.
- **Contract Identifier Generator:** This feature provides UUIDs or DIDs for contracts, ensuring their traceability and enabling easy reference within the system. The generated contract UUID is separate from the template UUID obtained from the template repository.
- **Automated Alerts:** This feature sends timely notifications for key events, such as contract renewals, expirations, or deadlines, helping users stay proactive and ensuring compliance with agreed timelines.

The interfaces required for the Contract Storage and Archive can be listed as follows:

- **CSA ← SM:** The archive automatically ingests executed contracts with signature containers and metadata after the signing workflow completes.
- **CSA ↔ CWE/SM:** Authorized components can retrieve archived contracts and evidence through search/retrieve APIs and dashboards for further workflow or compliance actions.
- **CSA → PACM:** Every archival, retrieval, update, and deletion operation produces an immutable audit record with actor, timestamp, operation, and outcome.
- **CSA → CWE (Events):** Rules-based monitoring generates alerts for expirations, renewals, and deadlines that feed back into workflow tasks.
- **CSA → External Clients (REST/UI):** Archival, metadata update, tagging, and retrieval functions are exposed via authorized APIs and archive manager UIs.

2.2.5 Process Audit and Compliance Management

Process audit and compliance management service ensures all contract-related activities adhere to legal, regulatory, and organizational standards. This service maintains tamper-proof audit trails for the entire contract lifecycle, capturing all steps in the contract workflow engine. Furthermore, it enables proactive compliance by identifying potential risks, such as unsigned documents or missed deadlines, and issuing alerts to mitigate them. Its core functions are listed as follows:

- **Tamper-Proof Audit Trail for Contract Lifecycle:** Maintain a complete, tamper-proof audit trail across the contract lifecycle.
- **Automated Regulatory and Policy Compliance Checks:** Validates contracts against legal, regulatory, and organizational rules before execution or signing
- **RBAC for Audit Logs:** Restricts audit log access to authorized roles like auditors and compliance officers.
- **Contract Non-Compliance Investigation and Reporting:** Provides tools to analyze and report on contract-related policy violations or missed obligations.

The interfaces required for Process Audit and Compliance Management can be listed as follows:

- **PACM ← TR/CWE/SM/CSA:** PACM ingests audit hooks from all modules to maintain a tamper-proof trail of actions, state changes, and compliance operations.
- **PACM ↔ CWE/SM/CSA:** Automated compliance checks and reports are available to workflow, signature, and archive functions for regulatory verification and investigation.

2.2.6 DCS-to-DCS Communication

DCS supports direct interoperability between two or more DCS instances, enabling automated contract lifecycle operations across organizational boundaries. This communication supports cross-company contract document sharing, controlled by policies for the Review, Negotiation and Signing and Renewal Process. Fig.4 illustrates this DCS-to-DCS communication pattern, showing how contracts are created, negotiated, signed, and updated between parties. The flow begins with integration into enterprise systems such as an ERP via an internal API. The initiating DCS instance leverages its Template Repository and associated Management Interfaces (Contract Management, Lifecycle Management, Negotiation UI) to assemble a contract. Each contract is uniquely identified through a Contract DID Service.

Organizational credential management is handled through the OCM W-Stack, which allows the company to issue and present verifiable credentials required for contract negotiation and signing. Contract templates and offers may also be published and retrieved through the Federated Catalogue, ensuring discoverability and reuse of standardized contract structures. The negotiation process occurs in a dedicated Negotiation Space, where offers, counteroffers, and modifications are exchanged and tracked. Once the contract is finalized, the

Workflow Engine coordinates contract state transitions and prepares the final agreement for signing. The Counterparty Signature step involves wallet-based signing with the required credentials.

Contracts and state changes are exchanged between DCS instances through the External API, which supports:

1. Create Contract – transmitting a new contract offer from one DCS instance to another.
2. Status Updates – synchronizing state changes (e.g., acceptance, counteroffer, withdrawal, revocation).

The figure also shows the revocation path, which allows either party to retract or terminate a contract in accordance with defined policies and lifecycle rules. This communication model ensures that two DCS systems can interoperate using standardized identifiers, verifiable credentials, and federated catalogue references.

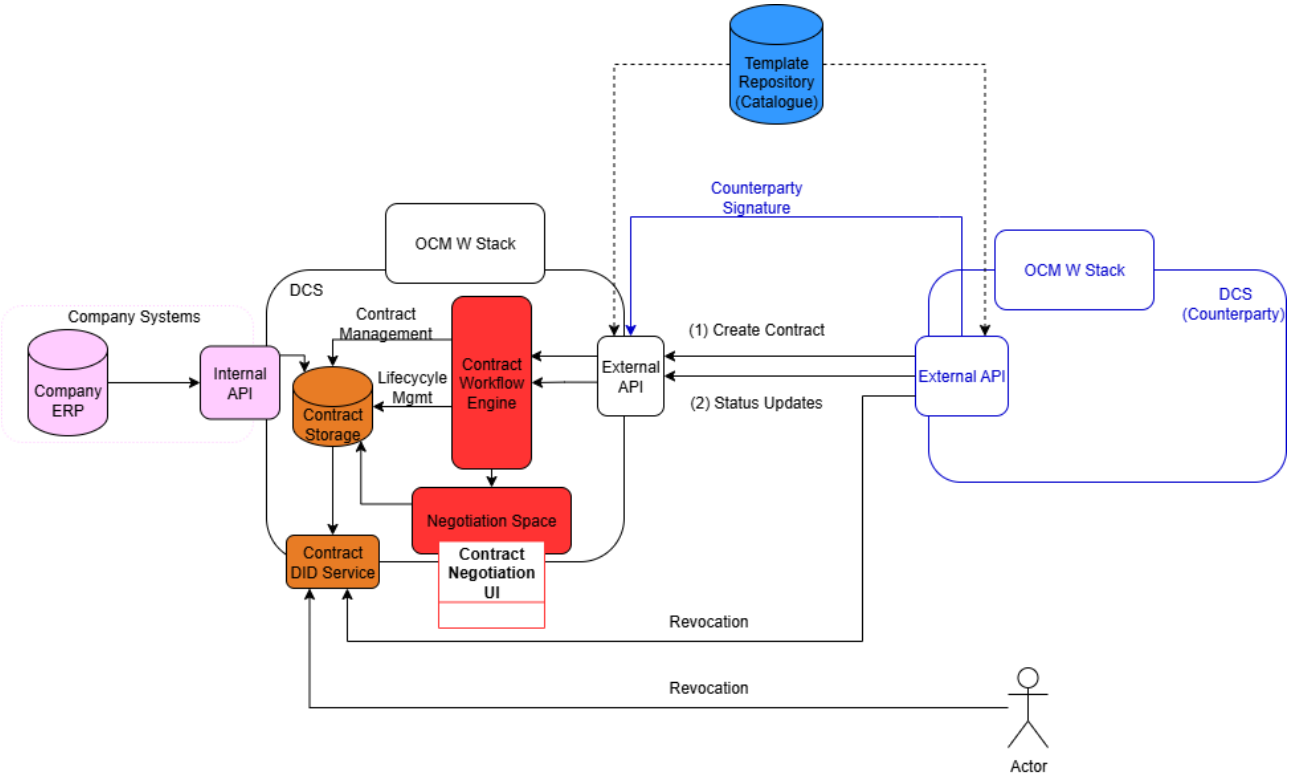


Fig.4 – DCS-to-DCS communication flow

2.3 Product Constraints

[DCS-PC-01] Use of XFSC Components

The DCS MUST leverage existing XFSC components (e.g., Catalogue, Orchestration Engine, Revocation List) wherever applicable. Alternative implementations of these functionalities are not permitted unless an XFSC equivalent is demonstrably unsuitable. The goal is to maximize reuse of established and standards-compliant ecosystem services.

[DCS-PC-02] Legal Contract Definition

The DCS may only handle contracts that meet EU legal validity requirements. In the initial scope, this includes contracts signed using Advanced Electronic Signatures (AES) for natural persons, with optional support for

Qualified Electronic Signatures (QES) or organizational seals in future releases. Contract forms requiring legal processes outside these signature methods are excluded unless explicitly defined in future specifications.

[DCS-PC-03] Technology Stack Preferences

The preferred implementation language is Go (Golang) due to its suitability for confidential computing and compatibility with XFSC architecture. APIs SHOULD be designed using the Goa design-first framework with Go-based code generation. Java MAY be used only if justified (e.g., in modules that use or extend the Catalogue) and if it does not conflict with Goa-based code generation. Recommended frameworks and tools include Goa for REST APIs, NATS for eventing, Docker containers, and databases with PostgreSQL-equivalent functionality (database vendor MUST remain abstracted).

[DCS-PC-04] Deployment Environment

The DCS MUST support deployment on Kubernetes clusters, with deployment configurations provided via Helm charts and ArgoCD. Docker Compose deployments are not acceptable for demonstration purposes. Continuous Integration/Continuous Deployment (CI/CD) pipelines MUST be implemented using GitHub Actions or equivalent services.

[DCS-PC-05] Ecosystem Constraints for Identity Wallets and TSP Services

The DCS MUST integrate with the existing identity wallet infrastructures (e.g., Animo solutions framework for XFSC) and Trust Service Providers (TSPs) for the provisioning of AES and, optionally, QES or organizational seals. These services are provided externally and will not be developed or operated within DCS. Version 1 will focus exclusively on AES for natural-person signatures, with QES/seal support planned for future iterations.

[DCS-PC-06] Template and Data Formats SLA and Data Exchange

Contract templates MUST be supported in machine-readable formats with semantic definitions. JSON-LD is the preferred format for representing these templates to ensure semantic interoperability across systems.

2.4 User Classes and Characteristics

The DCS is designed to serve a variety of user roles, each with specific access rights, responsibilities, and privileges. These roles are grouped into two main categories: Human Users and System Users. Human users interact with the system through dedicated dashboards and role-based interfaces for review, signing, and approval. System users, on the other hand, are typically components of automated environments or integrated third-party platforms and interact with DCS via APIs and orchestration layers. Human user classes and characteristics are given in Table 4 and Table 5, respectively.

The detailed functional requirements for the defined roles are specified in Section 3.2. For general characteristics, the role management requirements can be summarized in this section as follows:

- All users must authenticate using VCs.
- RBAC governs available actions and interface access.
- Each action (create, review, approve, sign) is traceable and associated with role-specific audit logs.
- Some roles (e.g., Validator, Compliance Officer) are tightly coupled with domain-specific regulatory policies.

Table 4 – Human user classes and characteristics

Role	Characteristic
Template Creator	Creates contract templates to ensure standardized and consistent structures.

Template Reviewer	Validates templates before approval, focusing on compliance and quality.
Template Approver	Gives final approval to templates, preventing use of incorrect structures.
Template Manager	Oversees all template lifecycle operations, including updates and deprecation.
Contract Creator	Drafts new contracts using approved templates, initiating the contracting process.
Contract Reviewer	Reviews draft contracts to verify accuracy and alignment with business terms.
Contract Approver	Validates final versions of contracts before signing to ensure compliance.
Contract Manager	Manages contract execution, renewal, and lifecycle updates.
Contract Signer	Signs contracts legally using qualified or advanced electronic signatures.
Contract Observer	Has read-only access to contracts, typically for monitoring or oversight.
Archive Manager	Manages storage, retrieval, and archiving of contracts.
Auditor	Conducts audits on contract data and process history to ensure compliance.
System Administrator	Maintains system configurations, permissions, and user access.
Process Orchestrator	Automates contract workflows within ERP or AI-driven environments.
Validator	Runs integrity and policy compliance checks on contracts before execution.
Compliance Officer	Ensures regulatory compliance of templates and contracts before approval.
Integration Manager	Manages third-party system integrations (e.g., ERP, Trust Services).

Table 5 – System user classes and characteristics

Role	Description
System Contract Creator	Creates contracts through API integration for automation use cases.
System Contract Reviewer	Conducts automated validation checks during contract review.
System Contract Approver	Programmatically approves contracts as part of integrated workflows.
System Contract Manager	Manages and modifies contracts within integrated systems (e.g., ERP, AI agents).
System Contract Signer	Performs API-based digital signing for autonomous or IoT systems.
Contract Target System	External system that receives and executes deployed contracts.

2.5 Operating Environment

[DCS-OE-01] Kubernetes Environment

The product MUST be operable on standard Kubernetes-based environments without any hardware restrictions. The reference environment for demonstration and development purposes MUST be deployed on IONOS Kubernetes as well as T-Systems Open Sovereign Cloud (OSC).

[DCS-OE-02] Containerization

All software components MUST be delivered as Linux-based Docker containers for deployment in Kubernetes environments. Docker Compose deployments are NOT acceptable for demonstration purposes.

[DCS-OE-03] Deployment Tooling

Deployment configurations MUST be provided as Helm charts, and GitOps-based delivery MUST be supported via ArgoCD.

[DCS-OE-04] CI/CD Pipelines
Continuous Integration/Continuous Deployment (CI/CD) MUST be implemented using GitHub Actions or an equivalent platform.

[DCS-OE-05] Database Support

The product MUST support PostgreSQL or databases with equivalent functionality. No hard dependency on a specific vendor is permitted.

[DCS-OE-06] Ecosystem Integrations

The product MUST integrate with:

- XFSC components (Catalogue, Orchestration Engine, Revocation List).
- Identity wallet infrastructures for authentication and authorization. TSPs for AES and optional QES/seals.
- External systems via RESTful APIs for contract automation and lifecycle integration.

2.6 User Documentation

The following documentation components will be delivered along with the software:

1. **End-User Documentation:** This documentation MUST contain short product overview, user roles, step-by-step task flows (create/review/approve/sign; retrieve from archive), and basic troubleshooting and FAQ.
2. **Administrator Documentation:** This documentation MUST contain installation & deployment guide, Configuration & RBAC, backup/restore and basic monitoring, and upgrade/migration procedures.

2.7 Assumptions and Dependencies

The development and operation of the DCS relies on the following assumptions and external dependencies. These factors are outside the direct control of the DCS development team and may affect the system requirements if they change or are not met.

Assumptions

- Identity wallet infrastructures conforming to EUDI standards will be available for authentication, authorization, and provisioning of AES and, optionally, QES or seals.
- TSPs will offer compatible APIs for remote AES/QES signing that can be integrated with the DCS Signature Management module.

Dependencies

- Availability and correct functioning of external XFSC components and services (Catalogue, Orchestration Engine, Revocation List) as maintained by the FACIS ecosystem.
- Integration with external systems via RESTful APIs for contract automation and lifecycle operations.
- Continued support of standards and specifications critical to DCS operation, including W3C Verifiable Credentials, JSON-LD, SHACL, and PAdES/JAdES.

2.8 Apportioning of Requirements

The functional and non-functional requirements of the DCS are allocated to specific software elements within the system's modular architecture. Each functional requirement is mapped to the component responsible for its implementation, as seen in Section 3.2. Non-functional requirements span multiple elements due to cross-cutting concerns such as security or compliance. The following capabilities are optional for DCS Version 1 and are planned for later releases:

- QES Support
- B2C Contract Support
- Integration with Remote QES Signing Services

3 Requirements

Requirements are prefixed 'DCS-IR' for interfaces, 'DCS-FR' for functionality, and 'DCS-NFR' for non-functional requirements, with component codes (e.g., TR, CWE, SM, CSA, PACM) appended to functional requirements where applicable.

3.1 Interfaces

3.1.1 User Interfaces

This section defines the DCS's user interfaces and flows for the user classes defined in Section 2.4.

3.1.1.1 Template Repository

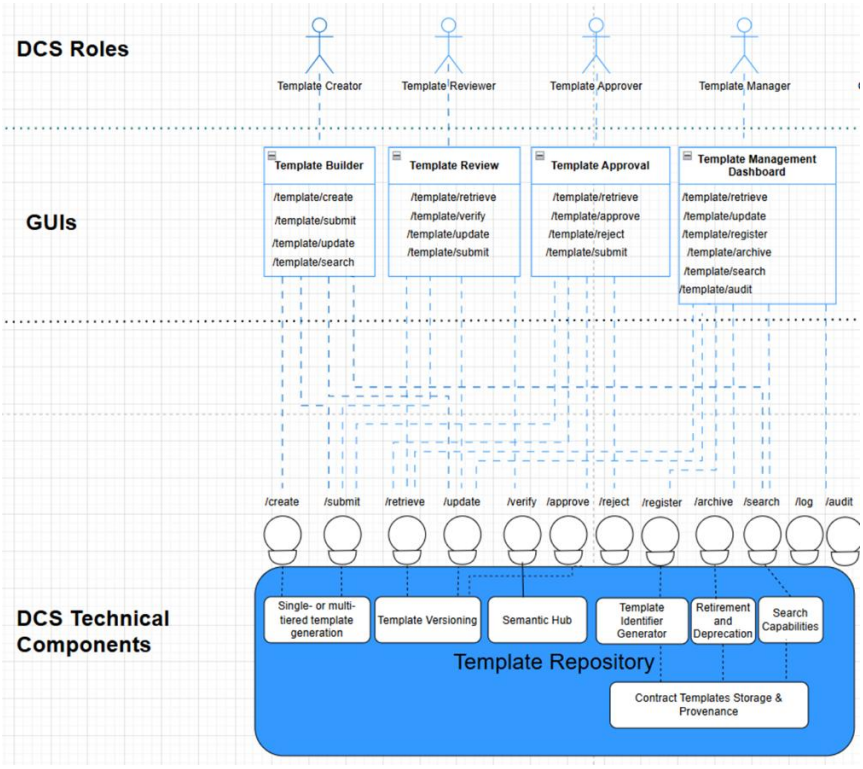


Fig.5 – Template Repository functional overview and UI

As shown in Fig.5, Template Repository contains the following user interfaces: Template Builder, Template Review, Template Approval, and Template Management Dashboard. The requirements, pre-conditions, steps, post-conditions, and the logical characteristics of these user interfaces are provided in this section.

Template Builder UI

Requirements

[DCS-IR-TR-01] Template Builder MUST allow Template Creator to create new contract templates and update existing ones.

[DCS-IR-TR-02] Template Builder MUST allow searching and retrieving existing templates for reuse or modification.

Pre-conditions

- Users assigned with Template Creator are authenticated with permissions to manage templates.
- Required metadata schemas, libraries, and semantic conditions are available and accessible.

Steps

1. Open Template Builder interface from the dashboard.
2. Enter template details, including metadata, clauses, and semantic conditions.
3. Optionally, search for existing templates by keyword, category, or metadata.

4. Update or refine template content as needed.
5. Submit the template for review and approval.

Post-conditions

- Draft template is stored in the repository with provenance tracking (see [DCS-FR-TR-08]).
- Submission generates a review task, assigned to the appropriate Template Reviewer.
- A unique identifier is assigned to the template for traceability.

Logical Characteristics

- Layout: Form-based interface with structured fields for metadata and clause insertion.
- Controls: Standard buttons (Create, Update, Save Draft, Submit, Cancel, Help).
- Error Handling: Inline validation for required fields; error messages displayed contextually.
- Usability & Accessibility: Search bar with autocomplete; support for keyboard shortcuts
- Audit Hooks: Every action logged with timestamp and user ID for compliance.
- Security: Role-based access control to prevent unauthorized template management.

Template Review UI

Requirements

[DCS-IR-TR-03] Template Review MUST allow Reviewers to retrieve, verify, update, and submit templates.

[DCS-IR-TR-04] Template Review MUST support forwarding a verified template to approval or returning it to draft with comments.

Pre-conditions

- A template exists in Submitted state.
- Reviewer is authenticated and authorized to review templates.
- Required validation rules (policy, semantic, schema) are available.

Steps

1. Retrieve the submitted template.
2. Verify content (policy checks, semantic/SHACL validation, schema completeness).
3. Optionally update metadata/clauses/semantic conditions or add review comments.
4. Either Forward to Approval or Return to Draft with comments.
5. Submit changes.

Post-conditions

- If accepted: template transitions to Approval state.
- If changes required: template transitions to Draft with reviewer comments and assigned tasks.
- All actions are logged with timestamp, user ID, and rationale.

Logical Characteristics

- Layout: Split view with (a) template details & metadata, (b) clause/semantic editor, (c) validation results panel, (d) comment thread.
- Controls: Retrieve, Verify, Save, Submit, Forward to Approval, Return to Draft, Cancel, Help.

- Validation & Errors: Inline field errors; non-blocking warnings; blocking errors for failed policy/semantic checks; retry guidance.
- Usability & Accessibility: Keyboard navigation, clear focus states, ARIA roles, WCAG-compliant contrast.
- Audit Hooks: View, verify, edit, submit, state-transition, and comment events logged.
- Security: RBAC; redact sensitive fields in comments/export; immutable audit for state changes.

Interface Endpoints

- GET /template/retrieve – load submitted template and history/provenance summary.
- POST /template/verify – run policy, schema, and semantic validations; return findings.
- PUT /template/update – persist reviewer edits (metadata/clauses/semantics).
- POST /template/submit – with action flag *{ forwardTo: "approval" | "draft" }* and optional *reviewComments*.

Template Approval UI

Requirements

[DCS-IR-TR-05] Template Approval MUST allow Approvers to retrieve, approve, reject, or resubmit templates.

[DCS-IR-TR-06] Template Approval MUST ensure that only validated templates enter the pool of contract-ready assets.

Pre-conditions

- A template exists in Reviewed state.
- Approver is authenticated and authorized to approve templates.
- Review history and comments are accessible for decision-making.

Steps

1. Retrieve the reviewed template.
2. Inspect review results, comments, and validation reports.
3. Decide: Approve (template becomes contract-usable), Reject (return to draft with reasons), or Request Resubmission (minor issues flagged).
4. Submit decision.

Post-conditions

- Approved: Template transitions to Approved state and becomes selectable in Contract Creation.
- Rejected: Template transitions to Draft with approver's rejection reason attached.
- Resubmission: Template transitions to Submitted with comments for another review cycle.
- All actions are logged with timestamp, user ID, and rationale.

Logical Characteristics

- Layout: Template details, validation results panel, comment history, and decision buttons.
- Controls: Approve, Reject (with reason input), Request Resubmission, Save, Cancel, Help.
- Validation & Errors: Prevents approval if mandatory metadata/clauses are missing; requires textual reason for rejection/resubmission.
- Usability & Accessibility: Keyboard shortcut for quick decisioning, accessible error messages, consistent placement of action buttons.

- Audit Hooks: Approver identity, timestamp, decision, and rationale captured.
- Security: RBAC enforcement; read-only access to provenance data; immutable logging of decisions.

Interface Endpoints

- GET /template/retrieve – fetch reviewed template with metadata, review history, and validation results.
- POST /template/approve – mark template as approved, with optional decision notes.
- POST /template/reject – mark template as rejected, requiring reason field.
- POST /template/submit – allow resubmission path with approver comments.

Template Management Dashboard UI

Requirements

[DCS-IR-TR-07] Template Management Dashboard MUST allow Managers to register, archive, update, search, and audit templates.

[DCS-IR-TR-08] Template Management Dashboard MUST provide lifecycle oversight of all templates in the repository.

Pre-conditions

- User is authenticated with Manager role.
- Template repository is available with searchable metadata and audit logs.

Steps

1. Access dashboard view of all templates (with filters: status, owner, date, version, etc.).
2. Perform actions:
    a. Register a new template into the repository.
    b. Update metadata or classifications.
    c. Archive obsolete templates.
    d. Search by identifiers, keywords, or semantic conditions.
    e. Audit template history for provenance and compliance.
3. Confirm changes or run audit reports.

Post-conditions

- Repository updated with new/modified/archived template entries.
- Full audit trail logged for all management actions.
- Dashboard reflects the latest template lifecycle state.

Logical Characteristics

- Layout: Central table/list of templates with filters, status indicators, and action buttons.
- Controls: Register, Update, Archive, Search, Audit, Export (CSV/PDF), Help.
- Validation & Errors: Prevents duplicate registration; blocks archiving of active templates still in use; requires justification text for archival.
- Usability & Accessibility: Batch operations (multi-select), search bar with auto-suggest, sortable columns, role-based tooltips.
- Audit Hooks: Logs who registered/updated/archived a template, with timestamps and rationale.

- Security: RBAC enforced; managers cannot alter provenance history; sensitive audit logs are read-only.

Interface Endpoints

- GET /template/retrieve – fetch all template entries for dashboard view.
- POST /template/update – update metadata or status.
- POST /template/register – register new template into the repository.
- POST /template/archive – archive obsolete template.
- GET /template/search – perform filtered searches.
- GET /template/audit – retrieve audit history of template actions.

3.1.1.2 Contract Workflow Engine

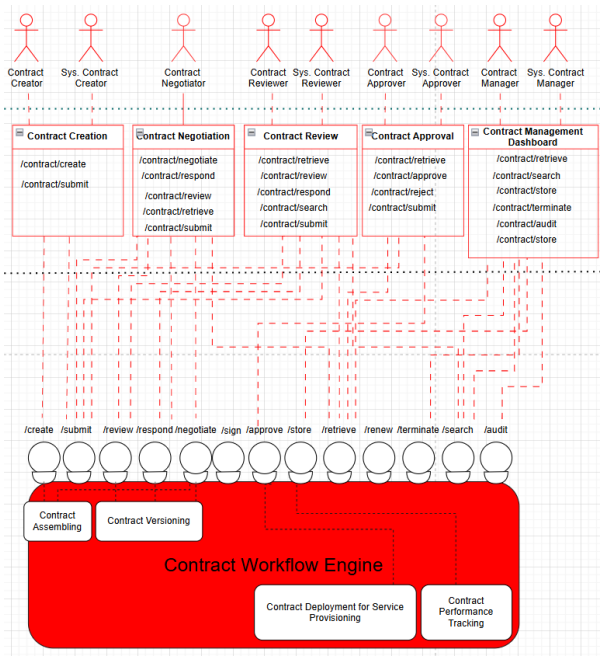


Fig.6 – Contract Workflow Engine functional overview and UI

As shown in Fig.6, Contract Workflow Engine contains the following interfaces: Contract Creation, Contract Negotiation, Contract Review, Contract Approval, and Contract Management Dashboard. The requirements, pre-conditions, steps, post-conditions, and the logical characteristics of these user interfaces are provided in this section.

Contract Creation

Requirements

[DCS-IR-CWE-01] Contract Creation UI MUST allow Contract Creators to create and submit contracts from approved templates.

[DCS-IR-CWE-02] Contract Creation UI MUST enable population of contract data, including parties, assets, policies, and evidence.

Pre-conditions

- At least one approved template exists in the Template Repository.
- User is authenticated and authorized with Contract Creator role.
- Required metadata schemas and identity verification services are available.

Steps

1. Access Contract Creation module.
2. Select an approved template from the repository.
3. Fill in required fields (e.g., contracting parties, assets/services, usage policies, semantic clauses).
4. Attach evidence or supporting credentials (e.g., company license, role-based PoA).
5. Validate draft against schema and policy checks.
6. Submit contract for negotiation or review.

Post-conditions

- Contract stored as draft (if incomplete) or as a submitted item ready for further workflow.
- Contract provenance (creator, timestamp, version, linked template ID) is logged.
- Notifications are sent to designated reviewers or counterparties.

Logical Characteristics

- Layout: Form-based UI with sections for parties, assets, policies, evidence attachments, and preview pane.
- Controls: Save draft, Validate, Submit, Attach credential, Help.
- Validation & Errors: Ensures all mandatory fields (parties, expiration date, jurisdiction, etc.) are completed; highlights semantic inconsistencies.
- Usability: Auto-fill for known parties/assets, drag-and-drop evidence attachment, guided template-based wizard.
- Audit Hooks: Logs all creation events, including metadata, drafts, and submission actions.
- Security: Enforces RBAC for contract creation; sensitive evidence data encrypted at rest.

Interface Endpoints

- POST /contract/create – initiate new contract draft from template.
- POST /contract/submit – finalize and submit contract for negotiation/review.

Contract Negotiation

Requirements

[DCS-IR-CWE-03] Contract Negotiation UI MUST allow parties to exchange responses, redlines, and comments prior to contract approval.

[DCS-IR-CWE-04] Contract Negotiation UI MUST support comparison of contract versions for transparency and traceability.

Pre-conditions

- A contract exists in negotiation state.
- Both parties are authenticated and authorized with appropriate roles (e.g., Contract Creator/Reviewer).

- Versioning and provenance tracking are enabled.

Steps

1. Retrieve current contract version.
2. Compare changes with prior versions (diff view).
3. Propose modifications (redlines, comments, clause edits).
4. Respond to counterpart proposals.
5. Validate draft against schema and policy rules.
6. Submit negotiated version for further review/approval.

Post-conditions

- An agreed version is forwarded to review or approval workflow.
- Contract provenance is updated with version history, actors, and timestamps.
- Notifications are issued to involved parties upon submission.

Logical Characteristics

- Layout: Side-by-side or inline diff view of clauses and policies; comment thread panel.
- Controls: Accept/Reject change, Propose change, Save draft, Compare versions, Submit.
- Validation & Errors: Detects conflicting edits or policy violations (e.g., exceeding jurisdiction scope).
- Usability: Highlighted redlines, inline commenting, track changes history, revert function.
- Audit Hooks: Immutable record of all negotiation steps, linked to contract ID.
- Security: Role-based permissions; ensures only authorized parties can propose or accept changes.

Interface Endpoints

- POST /contract/negotiate – propose changes.
- POST /contract/respond – respond to counterpart changes.
- GET /contract/review – retrieve latest draft for comparison.
- POST /contract/submit – finalize and submit negotiated version.

Contract Review

Requirements

[DCS-IR-CWE-05] Contract Review UI MUST allow Reviewers to retrieve, inspect, and validate contracts after negotiation.

[DCS-IR-CWE-06] Contract Review UI MUST allow Reviewers to respond with findings, request modifications, or forward contracts for approval.

[DCS-IR-CWE-07] Contract Review UI MUST provide search capabilities to locate contracts by metadata, parties, or template references.

Pre-conditions

- A contract exists in submitted state after negotiation.
- Reviewer role is authenticated and authorized.
- Semantic and policy validation rules are available.

Steps

1. Retrieve submitted contract.
2. Verify content against policy and semantic checks (e.g., jurisdiction, expiration, SLA clauses).
3. Search or filter related contracts if needed for comparison.
4. Respond with findings, request changes, or annotate clauses.
5. Forward contract to approval or return it to negotiation.

Post-conditions

- Contract is either advanced to approval state or reverted to negotiation state with Reviewer comments.
- Validation and review outcome are logged in the audit trail.
- Parties are notified of the decision.

Logical Characteristics

- Layout: Contract viewer with clause highlighting; policy/semantic check results shown in a validation pane.
- Controls: Approve for forwarding, Request changes, Add comments, Save review draft, Submit decision.
- Validation & Errors: Automated semantic validation flags missing/invalid attributes.
- Usability: Inline annotations, search/filter bar, compare to prior version option.
- Audit Hooks: Every review action is timestamped, signed, and appended to the contract provenance.
- Security: Role-restricted access; Reviewer signatures required for completed reviews.

Interface Endpoints

- GET /contract/retrieve – fetch submitted contract.
- POST /contract/respond – provide feedback/findings.
- GET /contract/search – locate contracts by metadata or state.
- POST /contract/submit – finalize review outcome.

Contract Approval

Requirements

[DCS-IR-CWE-08] Contract Approval UI MUST allow Approvers to retrieve contracts in reviewed state.

[DCS-IR-CWE-09] Contract Approval UI MUST allow Approvers to approve, reject (with reason), or resubmit contracts.

[DCS-IR-CWE-10] Contract Approval UI MUST ensure approved contracts are forwarded into the signing workflow and catalogue.

Pre-conditions

- Contract exists in reviewed state.
- Approver role is authenticated and authorized.
- Validation results from Review stage are available.

Steps

1. Retrieve the reviewed contract.
2. Inspect contract metadata, negotiation history, and reviewer findings.
3. Approve or reject the contract, optionally setting visibility rules (e.g., catalogue availability, restricted access).
4. Submit decision to finalize state.

Post-conditions

- Approved contract is available for signing workflow and registered in the catalogue.
- Rejected contract is returned to negotiation with Approver comments.
- All approval actions are logged in the audit trail with timestamp and Approver identity.

Logical Characteristics

- Layout: Contract approval panel with decision buttons (Approve, Reject, Resubmit) and visibility settings.
- Controls: View full contract text, access metadata, open reviewer comments, Approve, Reject (with reason box), Submit.
- Validation & Errors: Warning if mandatory fields, clauses, or signatories are missing.
- Usability: Inline review history, searchable contract metadata, visibility selector.
- Audit Hooks: Approval decision is signed digitally by Approver and appended to provenance log.
- Security: Restricted to Approvers; strong authentication required.

Interface Endpoints

- GET /contract/retrieve – fetch reviewed contract.
- POST /contract/approve – approve and forward contract.
- POST /contract/reject – reject with explanation.
- POST /contract/submit – finalize decision.

Contract Management Dashboard

Requirements

[DCS-IR-CWE-11] Contract Management Dashboard UI MUST allow Managers to retrieve and search contracts across lifecycle states.

[DCS-IR-CWE-12] Contract Management Dashboard UI MUST allow Managers to store evidence, terminate contracts, and perform audits.

[DCS-IR-CWE-13] Contract Management Dashboard UI MUST provide lifecycle monitoring aligned with XFSC lifecycle/log token usage.

Pre-conditions

- Manager role is authenticated and authorized.
- Contracts exist in the system across one or more states.

Steps

1. Retrieve and filter contracts by state (All, Drafts, In Review, Approved, Expired).
2. Select a contract to perform actions:
    a. Terminate contract

b. Store supporting evidence (files, metadata).
c. Run audit (view lifecycle, compliance, and log data).

3. Submit updates or actions.

Post-conditions

- Contract lifecycle state updated (e.g., terminated).
- Evidence stored and linked to the contract.
- Audit results appended to log records and accessible in compliance reports.

Logical Characteristics

- Layout: Dashboard with tabs/filters for lifecycle states and a detail pane for selected contract actions.
- Controls: Search bar, filters, buttons for Terminate, Store Evidence, Run Audit.
- Validation & Errors: System warns if evidence is missing or if termination reason not provided.
- Usability: One-click filtering, sortable contract lists, role-based access.
- Audit Hooks: All Manager actions digitally signed and logged with timestamp.
- Security: Strict access limited to Managers; evidence integrity enforced.

Interface Endpoints

- GET /contract/retrieve – fetch contract(s).
- GET /contract/search – filter/search across lifecycle states.
- POST /contract/terminate – terminate a contract.
- POST /contract/audit – generate audit record.
- POST /contract/store – store evidence.

3.1.1.3 Signature Management

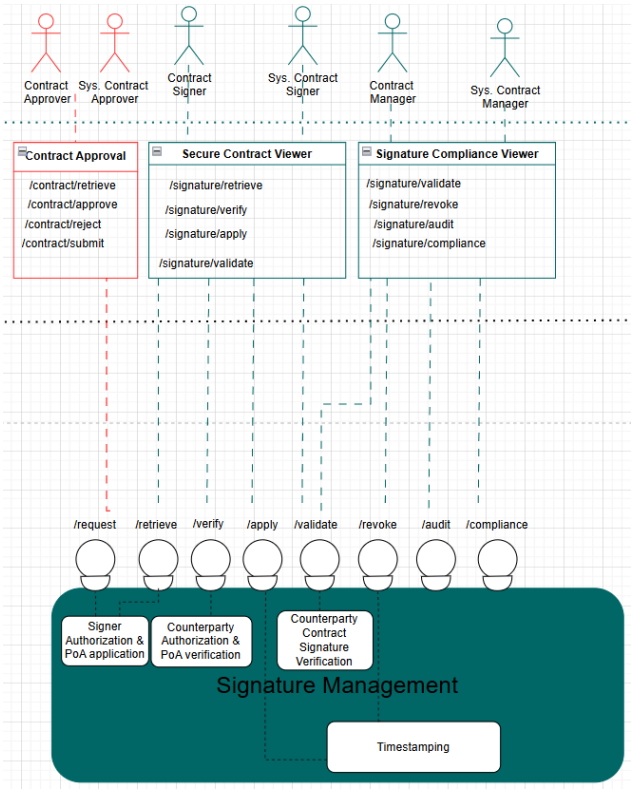


Fig.7 – Signature Management functional overview and UI

As shown in Fig.7, Signature Management component contains the following user interfaces: Secure Contract Viewer and Signature Compliance Viewer. The requirements, pre-conditions, steps, post-conditions, and the logical characteristics of these user interfaces are provided in this section.

Secure Contract Viewer

Requirements

[DCS-IR-SM-01] Secure Contract Viewer UI MUST allow Signers and Managers to retrieve approved contracts prepared for signing.

[DCS-IR-SM-02] Secure Contract Viewer UI MUST allow verification of contract integrity and signature envelopes.

[DCS-IR-SM-03] Secure Contract Viewer UI MUST allow applying signatures with appropriate credentials (e.g., QES/AES with PoA if required).

[DCS-IR-SM-04] Secure Contract Viewer UI MUST allow validation of applied signatures to ensure compliance and integrity.

Pre-conditions

- Contract is approved for signing state.
- Signer/Manager role authenticated with valid signing rights.
- Signature envelope (metadata, policies, required signers) is available.

Steps

1. Retrieve contract and associated signature envelope.
2. Verify contract content integrity and required signing policies.
3. Apply signature using assigned credentials.
4. Validate applied signature(s) against trust policies.
5. Confirm submission of executed contract.

Post-conditions

- Executed contract stored in system with applied signatures.
- Validation artifacts (cryptographic proofs, timestamp, signer ID) linked to contract.
- Audit log updated with retrieval, verification, signature, and validation actions.

Logical Characteristics

- Layout: Split-view screen: contract content viewer on left, signature actions panel on right.
- Controls: Buttons for Verify, Apply Signature, Validate, Submit.
- Feedback: Validation results displayed (success, warnings, or errors with details).
- Error Handling: Alerts if signer lacks valid credentials, contract integrity fails, or validation fails.
- Security: End-to-end encryption of contract during retrieval, signature application, and storage.
- Usability: Step-by-step guided signing wizard; role-based interface adaptation.

Interface Endpoints

- GET /signature/retrieve – fetch contract & envelope.
- POST /signature/verify – check contract integrity & envelope.
- POST /signature/apply – apply digital signature.
- POST /signature/validate – validate applied signature.

Signature Compliance Viewer

Requirements

[DCS-IR-SM-05] Signature Compliance Viewer UI MUST allow compliance users to validate trust anchors, cryptographic proofs, and timestamps of signatures.

[DCS-IR-SM-06] Signature Compliance Viewer UI MUST allow revocation of signatures if required (e.g., signer credentials revoked, policy breach).

[DCS-IR-SM-07] Signature Compliance Viewer UI MUST allow running compliance checks against applicable policies, standards (eIDAS, ETSI), and business rules.

[DCS-IR-SM-08] Signature Compliance Viewer UI MUST allow generating audit reports covering validation and compliance results.

Pre-conditions

- A signed contract exists in the system.
- Compliance role (e.g., Auditor, Compliance Officer) is authenticated.
- Revocation list or registry is accessible (XFSC-compatible or external TSP).

Steps

- Retrieve signed contract and associated validation data.
- Validate trust anchors, cryptographic integrity, and timestamp validity.
- If applicable, revoke signature and record reason.
- Run compliance checks against defined standards/policies.
- Generate and export compliance/audit report.

Post-conditions

- Compliance report recorded and linked to the contract.
- Revocation status updated in contract record and revocation registry.
- Audit trail extended with validation, revocation, and compliance events.

Logical Characteristics

- Layout: Dashboard view with tabs: Validation, Revocation, Compliance Checks, Audit Reports.
- Controls: Buttons for Validate, Revoke, Run Compliance, Export Report.
- Feedback: Real-time compliance results with clear pass/fail indicators and detailed findings.
- Error Handling: Notifications if trust registry unreachable, revocation list unavailable, or compliance checks fail.
- Usability: Filter and search signed contracts by compliance status; export audit reports as PDF and JSON.

Interface Endpoints

- POST /signature/validate – validate contract signature(s).
- POST /signature/revoke – revoke a signature.
- GET /signature/audit – retrieve compliance/audit logs.
- POST /signature/compliance – run compliance check.

3.1.1.4 Contract Storage and Archive

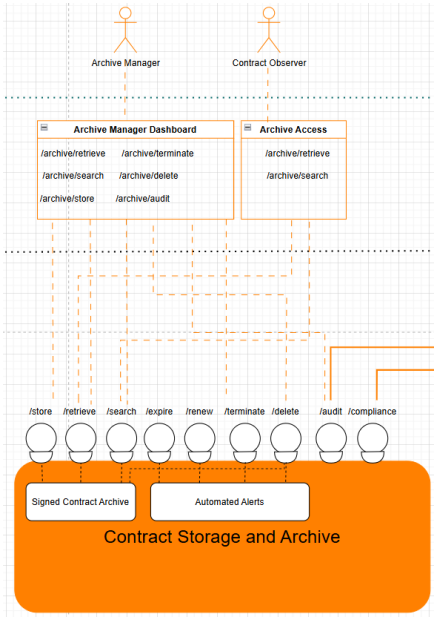


Fig.8 – Contract Storage and Archive functional overview and UI

As shown in Fig.8, Contract Storage and Archive component contains the following user interfaces: Archive Manager Dashboard and Archive Access. The requirements, pre-conditions, steps, post-conditions, and the logical characteristics of these user interfaces are provided in this section.

Archive Manager Dashboard

Requirements

[DCS-IR-CSA-01] Archive Manager Dashboard UI MUST allow Archive Managers to retrieve and search archived contracts and related records.

[DCS-IR-CSA-02] Archive Manager Dashboard UI MUST allow storing new contracts and evidence in the archive.

[DCS-IR-CSA-03] Archive Manager Dashboard UI MUST allow terminating or deleting archived entries under defined policies.

[DCS-IR-CSA-04] Archive Manager Dashboard UI MUST allow running audits on archive operations and integrity.

Pre-conditions

- Archive service is configured and operational.

- Archive Manager role is authenticated with necessary privileges.
- Retention and deletion policies are available in the system.

Steps

1. Retrieve and search archived contracts or evidence.
2. Store new items (contracts, signatures, compliance reports) into archive.
3. Terminate or delete entries in accordance with policy.
4. Run audit report for archive activities and integrity validation.

Post-conditions

- Tamper-evident archive updated with new, modified, or removed items.
- Audit log extended with retrieval, storage, termination, and deletion events.
- Archive state synchronized with compliance and retention policies.

Logical Characteristics

- Layout: Dashboard view with panels for Retrieve/Search, Store, Terminate/Delete, Audit.
- Controls: Buttons for Store New Entry, Terminate, Delete, Run Audit Report.
- Feedback: Confirmation dialogs for termination/deletion; alerts on policy violations.
- Error Handling: Notifications if archive storage is full, unavailable, or policy prevents deletion.
- Usability: Advanced search (by contract ID, signer, date, status); exportable audit reports.

Interface Endpoints

- GET /archive/retrieve – retrieve archived items.
- POST /archive/store – store new contract or evidence.
- POST /archive/terminate – terminate contract/archive entry.
- DELETE /archive/delete – permanently delete entry.
- GET /archive/search – search archived records.
- GET /archive/audit – retrieve audit logs.

Archive Access

Requirements

[DCS-IR-CSA-05] Archive Access UI MUST allow Observers to retrieve and search archived contracts and records with least-privilege access.

[DCS-IR-CSA-06] Archive Access UI MUST ensure that read-only users cannot modify, terminate, or delete entries.

Pre-conditions

- Observer role is authenticated and authorized.
- Archive service is operational.

Steps

1. Retrieve archived records.
2. Filter results by metadata (e.g., contract ID, date, status, party).
3. View contract or evidence in read-only mode.

Post-conditions

- No changes made to archive state.
- All access events logged to audit trail.

Logical Characteristics

- Layout: Simplified dashboard or portal view with search bar and result table.
- Controls: Search, Filter, Export (PDF and JSON).
- Feedback: Read-only access notification; error messages if attempting restricted actions.
- Usability: Advanced search with filters for contract metadata; role-specific minimal UI.
- Convenience: Quick export/download option for permitted formats.

Interface Endpoints

- GET /archive/retrieve – retrieve archived items.
- GET /archive/search – search records by criteria.

3.1.1.5 Process Audit and Compliance Management

As shown in Fig.9, Process Audit and Compliance Management component contains the following user interfaces: Auditing Tool and Non-Compliance Investigation. The requirements, pre-conditions, steps, post-conditions, and the logical characteristics of these user interfaces are provided in this section.

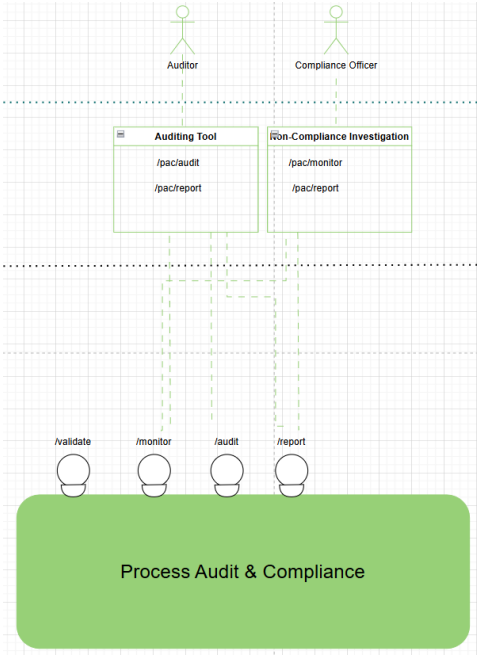


Fig.9 – Contract Storage and Archive functional overview and UI

Auditing Tool

Requirements

[DCS-IR-PACM-01] Auditing Tool UI MUST allow Auditors to initiate audits across contracts, templates, and signatures.

[DCS-IR-PACM-02] Auditing Tool UI MUST provide reporting capabilities with exportable audit results.

Pre-conditions

- Auditor role authenticated and authorized.
- Audit scope, rules, and policies configured in system.

Steps

1. Select audit scope (templates, contracts, signatures, or archive).
2. Execute audit via /pac/audit.
3. Review findings in dashboard (violations, inconsistencies, compliance checks).
4. Generate structured audit report via /pac/report.

Post-conditions

- Audit results and reports stored in tamper-proof log.
- Findings linked to relevant contract or template IDs.

Logical Characteristics

- Layout: Dashboard with filters (date, module, scope).
- Controls: Run Audit, Export Report, View Findings.
- Feedback: Real-time progress indicator; error messages for failed checks.
- Usability: Drill-down into individual findings; severity level indicators.
- Convenience: Report export in multiple formats (PDF, CSV, JSON).

Interface Endpoints

- POST /pac/audit – trigger an audit on selected scope.
- GET /pac/report – generate and retrieve audit reports.

Non-Compliance Investigation

Requirements

[DCS-IR-PACM-03] Non-Compliance Investigation UI MUST allow Compliance Officers to continuously monitor events and policy adherence.

[DCS-IR-PACM-04] Non-Compliance Investigation UI MUST allow incident reporting and linking findings to affected contracts or templates.

Pre-conditions

- Monitoring rules and thresholds defined.
- Compliance Officer role authenticated and authorized.

Steps

1. Access monitoring dashboard via /pac/monitor.
2. Review detected policy breaches, anomalies, or flagged events.
3. Document incident details and attach supporting evidence.
4. Submit findings via /pac/report.

Post-conditions

- Non-compliance case created with unique identifier.
- References to affected contracts, templates, or signatures stored.
- Case status visible in compliance dashboard.

Logical Characteristics

- Layout: Incident dashboard showing event streams, filters, severity levels.
- Controls: Monitor, Flag Incident, Generate Report, Link to Contract.
- Feedback: Alert notifications for detected breaches; confirmation of report submission.
- Usability: Drill-down from high-level alerts into specific event data.
- Convenience: Case export for external regulatory submission.

Interface Endpoints

- GET /pac/monitor – continuous monitoring and event retrieval.
- POST /pac/report – submit non-compliance findings as case records.

3.1.2 Hardware Interfaces

[DCS-IR-HI-01] Interface for Use of Signing Secrets (HSM/QSCD/TPM)
Private keys and other service secrets used by DCS MUST be protected by hardware security mechanisms (e.g., HSM, QSCD, or TPM/Secure Enclave), using standardized interfaces, with secrets held in a non-exportable, tamper-resistant manner; the solution SHOULD support operation in virtualized/containerized environments without weakening hardware protection.

[DCS-IR-HI-02] FIDO2 Security Key Interface
DCS web clients MUST support hardware authenticators (USB/NFC/BLE security keys or platform authenticators) for phishing-resistant login and step-up approval, using standard browser APIs (WebAuthn) with keys remaining protected in the device.

[DCS-IR-HI-03] Platform TPM 2.0 / Secure Enclave Interface
DCS services MUST be able to protect service credentials by sealing them to a platform hardware root of trust (TPM 2.0 or Secure Enclave) and SHOULD support remote attestation to prove platform integrity prior to enabling sensitive operations.

3.1.3 Software Interfaces

[DCS-IR-SI-01] Template Catalogue Integration
An interface MUST be provided between the TR and the XFSC Catalogue for template discovery, request, and registration via application APIs.

[DCS-IR-SI-02] Workflow Orchestration (Node-RED) Integration
An interface MUST be provided between the CWE and the XFSC Orchestration Engine (Node-RED) exposing Node-RED–compatible endpoints and webhook callbacks for automated workflow invocation.

[DCS-IR-SI-03] Platform Authentication & Authorization Integration
Interfaces MUST be provided between all DCS components and the Authentication & Authorization Service implementing OAuth2/OIDC flows for service-to-service and user access control.

[DCS-IR-SI-04] Wallet & TSP Signing Integration
An interface MUST be provided between SM and identity wallets and TSPs supporting OpenID4VCI/4VP for credential issuance/presentation and remote AES/QES signing and validation.

[DCS-IR-SI-05] External Target System API Integration
An interface MUST be provided between DCS (CWE/SM/CSA) and external target systems (e.g., ERP or AI services) exposing contract-processing and automated-interaction APIs for create/deploy actions, status queries, and event callbacks.

[DCS-IR-SI-06] Counterparty DCS Information Endpoint
An interface MUST be provided between a DCS instance and a counterparty DCS offering a policy-gated, read-only contract information endpoint.

[DCS-IR-SI-07] OpenID Provider Discovery & JWKS Consumption
An interface MUST be provided between the DCS Authentication & Authorization Service and external OpenID Providers (IdPs) to consume discovery metadata and JWKS for token validation.

[DCS-IR-SI-08] OpenID4VP Login & Access Control
An interface MUST be provided between the DCS Authentication & Authorization Service and identity wallets to accept OpenID4VP verifiable presentations of authorization credentials for login and access control.

[DCS-IR-SI-09] Credential Status & Revocation Service
An interface MUST be provided between DCS and a status/revocation list service to check and update credential status during validation and enforcement.

[DCS-IR-SI-10] Digital Signature Service (DSS) Authorization & Signing
An interface MUST be provided between SM and a DSS to authorize and execute remote AES/QES operations and return signature artifacts and timestamps.

[DCS-IR-SI-11] Relational Database Access
An interface MUST be provided between DCS components and the relational database (e.g., PostgreSQL) for CRUD access to shared entities using versioned schemas and migrations.

[DCS-IR-SI-12] Crypto Provider & DID/VC Operations
An interface MUST be provided between DCS and the Crypto Provider Service to create/verify DID documents and perform VC/VP signing and verification required by wallet integrations.

3.1.4 Communications Interfaces

[DCS-IR-CI-01] HTTPS/TLS 1.3 Transport
All external service communications for DCS MUST use HTTPS over TLS 1.3 for confidentiality and integrity.

[DCS-IR-CI-02] REST/JSON API Conventions
All application APIs MUST follow REST semantics with application/json request/response bodies; binary

artifacts (e.g., signed PDFs) MUST be served as application/pdf. This applies to the signature and archive endpoints already defined.

[DCS-IR-CI-03] Browser Access over HTTPS
All web UI interactions (Template, Workflow, Signature, Archive, PACM) MUST be delivered via HTTPS, with no additional non-web protocols required.

[DCS-IR-CI-04] OAuth2/OIDC Flows
DCS MUST implement OAuth2/OIDC authorization, token, and introspection flows for both user and service access to APIs.

[DCS-IR-CI-05] OpenID Discovery & JWKS
DCS MUST consume `well-known` OpenID Provider discovery metadata and JWKS endpoints for token and client validation.

[DCS-IR-CI-06] OpenID4VC/VP Bindings
Wallet interactions MUST use OpenID4VCI for issuance and OpenID4VP for presentation during login/authorization and signing flows.

[DCS-IR-CI-07] Orchestration Webhooks
Workflow invocation and callbacks between the XFSC Orchestration Engine (Node-RED) and DCS MUST use HTTP(S) endpoints and webhooks compatible with Node-RED nodes.

[DCS-IR-CI-08] DSS Remote Signing over HTTPS
Remote AES/QES operations with a DSS or TSP MUST be invoked over HTTPS and return standard signature containers.

[DCS-IR-CI-09] Revocation List Synchronization
Credential and contract-status checks MUST query a compatible revocation/status list service; updates to published status MUST be reflected within ≤ 5 minutes.

[DCS-IR-CI-10] PACM Audit Event Transport
Audit and compliance operations MUST use HTTPS JSON endpoints for event submission and report retrieval (/pac/audit, /pac/report).

3.2 Functional Requirements

3.2.1 Template Repository

[DCS-FR-TR-01] Machine-Readable Format
The Template Repository facilitates the structured storage and management of contract templates in machine readable source forms. Ensuring contract templates in machine-readable forms allows for automation and validation across digital contracting systems. Thus, the repository MUST store contract templates in structured, machine-readable formats. This does not indicate that the repository needs to be open to arbitrary contract languages.

[DCS-FR-TR-02] Multi-Tiered Contract Template Management
The Template Repository MUST support hierarchical contract structures, including Frame Agreements, Sub Agreements, and Addendums as digital contracts often involve multiple levels of agreements, requiring structured template management that allows customization at different levels. Examples include contract schemas, credential schemas, provenance schemas, and trust chaining schemas.

[DCS-FR-TR-03] Semantic Hub for Schema Storage
The Semantic Hub facilitates standardization and interoperability across different contract templates by providing a structured schema repository for validation and compliance. For this reason, the repository MUST include a Semantic Hub to store schemas supporting machine-readable formats, such as SHACL shapes and JSON-LD contexts, ensuring interoperability and extensibility. The Semantic Hub MUST also support versioning of the schemas.

[DCS-FR-TR-04] Machine-Readable and Human-Readable Template Linking
All templates MUST have a bidirectional link between machine-readable and human-readable formats.

[DCS-FR-TR-05] Template Version Control
The Template Repository MUST track changes, versions, and approvals for each template. These tracked changes MUST be made available for logging, monitoring, auditing, and reporting purposes to ensure compliance, transparency, and dispute resolution among evolving contract templates. The versions, changes, and approvals can be logged into an external system.

[DCS-FR-TR-06] Role-Based Access Control for Template Repository
The system must implement RBAC to restrict template access and modifications according to user roles, ensuring that only authorized personnel can create, approve, or retrieve templates. RBAC will be managed through role assertions in the form of verifiable credentials stored in users' wallets, which will be presented during the authentication and authorization process. This approach prevents unauthorized alterations that could lead to errors, compliance risks, or unintended contract modifications. Specifically, it ensures that only Template Managers can create templates, Template Approvers can approve them, Template Reviewers can review them, and Template Creators can initiate their creation.

[DCS-FR-TR-07] Compliance & Legal Validation

The Template Repository MUST validate templates against regulatory frameworks before they can be used in contract creation, with the specific frameworks depending on the use case domain. This validation ensures that contract templates meet legal and compliance requirements before they are applied in digital agreements, adhering to the relevant regulations for each domain. The validation occurs over the process audit and compliance component of the Digital Contracting Service, which provides access to a user with the Compliance Officer role over the non-compliance investigation tool.

[DCS-FR-TR-08] Provenance Tracking
The repository MUST maintain provenance tracking for all contract templates, recording their creation, modifications, approvals, and historical versions to prevent unauthorized modifications and enabling verification of the contract's legal history. “Provenance” means being able to track who created, modified, reviewed, or approved a template at each step with logged data. It MUST be ensured that each role – template creator, template reviewer, template approver, and template manager – leaves a traceable record when interacting with the template.

[DCS-FR-TR-09] Template Provenance and Versioning
The repository MUST support the ability for a Template Creator, Reviewer, and Approver to add provenance claims to each template version. Provenance and versioning assertions MUST be linked and MUST be verifiable using W3C VCs in JSON-LD. Template users MUST be able to verify the provenance of a given template by verifying the template provenance credentials.

[DCS-FR-TR-10] Searchable Metadata & Categorization
In order to enable organizations to find appropriate templates, the repository MUST allow advanced searching based on metadata such as contract type, jurisdiction, industry, and regulatory compliance status.

[DCS-FR-TR-11] Template UUID / DID Assignment
Each contract template MUST have a UUID or DID to ensure its traceability across contract workflows, providing consistency and authenticity when templates are referenced in negotiations, agreements, and compliance audits.

[DCS-FR-TR-12] Template Customization
The repository SHOULD support dynamic placeholders and automated population of contract terms based on predefined SLA rules, reducing manual input errors and enhancing efficiency by allowing templates to automatically adjust to specific service agreements.

[DCS-FR-TR-13] Template Creation
The system MUST allow a Template Manager to register a new contract template via the /template/register endpoint to ensure that only authorized personnel can create and manage contract templates.

[DCS-FR-TR-14] Template Submission for Approval
The system MUST enforce an approval workflow where a Template Manager submits a newly created template for review by a Template Approver, ensuring that all templates meet quality and compliance standards before being made available for use.

[DCS-FR-TR-15] Template Approval Process
The system MUST allow a Template Approver to review, approve, or reject a submitted template via a dedicated approval interface, preventing unapproved or incorrect templates from being used in contract workflows.

[DCS-FR-TR-16] Template Update Management
The repository MUST allow authorized users to update existing contract templates while maintaining version history and linking updates to previous versions, ensuring continuity and transparency in template evolution by tracking modifications and preserving historical versions for auditability. The entire history can be maintained by an external system.

[DCS-FR-TR-17] Template Retirement and Deprecation
The repository MUST support the deprecation and retirement of contract templates, ensuring that outdated versions are archived and no longer used in new contracts, while maintaining compliance and contract integrity through the preservation of historical records for reference and audits.

[DCS-FR-TR-18] Template Deletion
The system MUST allow a Template Manager to delete a deprecated template, preventing outdated templates from being used. The deletion process may be subject to specific compliance requirements that are out of scope.

[DCS-FR-TR-19] Template Retrieval
The system MUST allow a Template User to retrieve an approved template via the /template/retrieve/{template_id} endpoint to ensure that only approved and active templates are used in contract workflows.

[DCS-FR-TR-20] Template Compliance and Integrity Verification
The system MUST provide an endpoint (/template/verify/{template_id}) that enables Template Users to verify the integrity and compliance of a retrieved template, ensuring that it adheres to regulatory standards and has not been altered since approval.

[DCS-FR-TR-21] Audit Logs for Template Changes
The system MUST maintain an audit log that records all template creation, modification, approval, and deletion activities, providing transparency and accountability by tracking template-related actions.

[DCS-FR-TR-22] Notification System for Template Updates
The system SHOULD notify Template Users when a contract template they have used has been updated or deprecated, ensuring they work with the latest approved templates and are aware of changes that may impact ongoing contract workflows.

[DCS-FR-TR-23] Structural Dependency Mapping The Template Repository MUST allow Template Managers to define and enforce dependencies between frame contracts, contracts, contract components, and appendices, preventing misconfiguration of multi-part templates and ensuring structural consistency at runtime.

[DCS-FR-TR-24] Structural Export in Unified Format
The Template Repository MUST support export of a fully assembled contract structure (main contract, appendices, components) into a bundled format, enabling contract preview, documentation, and offline review by bundling all elements together.

[DCS-FR-TR-25] Multi-Contract Template Builder
The Template Repository MUST provide a visual interface for Template Managers to compose contract templates with nested contracts, appendices, and components, enhancing usability, reducing errors, and simplifying the management of complex contract structures.

[DCS-FR-TR-26] Logical Validation of Structural Dependencies
The system MUST check for consistency and validity of logical dependencies before usage to prevent invalid combinations of template components and enforces business rules.

[DCS-FR-TR-27] Contract Type Classification
The Template Repository MUST support defining and assigning structured single- and multi-party contract types for standardized filtering and governance.

[DCS-FR-TR-28] Template Management Dashboard (see Section 3.1)
The system MUST provide a Template Management Dashboard that enables the Template Manager to manage and oversee the entire contract template lifecycle. This dashboard is used to manage templates through an interface for tracking, modifying, and approving templates efficiently.

3.2.2 Contract Workflow Engine

[DCS-FR-CWE-01] Multi-Party Contract Management
The system MUST support workflows that involve multiple parties in a single contract or contract package. Each party MUST be able to independently sign, approve, or review their section in accordance with the predefined process.

[DCS-FR-CWE-02] Hierarchical Contract Structures
The system MUST support hierarchical structures such as master agreements, sub-agreements, and annexes. These MUST be logically linked and version-controlled for consistent contract management.

[DCS-FR-CWE-03] Contract Assembling
The system MUST support dynamic contract assembling from reusable clauses and templates. The assembly process MUST validate structure, required metadata, and content logic.

[DCS-FR-CWE-04] Machine-Readable & Human-Readable Contract Synchronization
The system MUST ensure synchronization between machine-readable and human-readable versions of a contract. Both formats MUST be derived from the same source and have matching content hashes.

[DCS-FR-CWE-05] Secure Human-Readable Contract Viewer
The system MUST include a tamper-proof viewer that displays the human-readable version of a contract for review and signing. The viewer MUST prevent changes and provide full visibility before signature.

[DCS-FR-CWE-06] Event-Driven Contract Execution
The system MUST support event-based workflows where contract execution steps are triggered by specific lifecycle events (e.g., all parties signed, approval completed). Events MUST be logged.

[DCS-FR-CWE-07] Role-Based Access Control
The system MUST enforce access to contract workflows based on roles (e.g., Reviewer, Signer, Manager). Role assignments MUST be verifiable via credentials and restrict unauthorized actions.

[DCS-FR-CWE-08] Version Control
Each contract instance MUST maintain version history. Edits MUST generate a new version with timestamps and user attribution. Old versions MUST remain accessible for audit and rollback.

[DCS-FR-CWE-09] SLA & Compliance Monitoring
The system MUST continuously monitor contractual obligations (e.g., service levels, deadlines) and flag SLA violations. Compliance rules MUST be enforced throughout the contract lifecycle.

[DCS-FR-CWE-10] Contract Expiry
The system MUST automatically detect and flag contracts approaching expiration. It MUST trigger alerts for renewal, termination, or archiving workflows depending on configured policies.

[DCS-FR-CWE-11] Contract Renewal
Authorized users MUST be able to renew contracts while retaining linked metadata and signatures from the previous version. Renewals MUST generate a new contract instance with reference links.

[DCS-FR-CWE-12] Termination Handling
The system MUST support formal contract termination, including capturing the reason, author, timestamp, and preserving the contract status for compliance and dispute resolution.

[DCS-FR-CWE-13] Contract Creation
The system MUST allow the creation of new contracts from templates, auto-filling metadata such as parties, jurisdiction, and applicable schemas. Drafts MUST be editable and versioned.

[DCS-FR-CWE-14] Contract Submission for Review
Once created, contracts MUST be submitted to assigned reviewers for validation. The system MUST support approval routing, tracking of reviewer input, and status change upon submission.

[DCS-FR-CWE-15] Contract Review and Approval
Designated reviewers MUST be able to approve or reject contracts. All actions MUST be logged with comments, digital credentials, and time of action. Rejection MUST return the contract to draft status.

[DCS-FR-CWE-16] Contract Initiation
Upon approval, the contract MUST be marked as ready for execution. The system MUST transition it into the signing phase or deploy it to external systems, depending on configuration.

[DCS-FR-CWE-17] Contract Review
The system MUST support detailed contract review, including side-by-side comparison of versions, redlining, and automated checks for missing fields or inconsistencies.

[DCS-FR-CWE-18] Contract Negotiation
The system MUST enable parties to engage in structured negotiation workflows. This includes comment threads, redline proposals, approvals of changes, and full negotiation audit logs.

[DCS-FR-CWE-19] Contract Signing
Once approved, the system MUST initiate the signing process. Each party MUST be guided through their required steps, with signature validity and completion tracked in real time.

[DCS-FR-CWE-20] Store Contract in Archive
Signed contracts MUST be stored in the contract archive, along with signature metadata, version history, and credential hashes. Archived contracts MUST be immutable and auditable.

[DCS-FR-CWE-21] Retrieve Contract from Archive
Authorized users MUST be able to retrieve archived contracts based on filters such as contract ID, status, or participant. Retrieval MUST maintain audit trail and access control.

[DCS-FR-CWE-22] Contract Renewal Management
The system MUST provide a dedicated renewal workflow, including template reuse, automatic metadata carryover, and notification to involved parties about renewal deadlines.

[DCS-FR-CWE-23] Contract Termination
Contract Managers MUST be able to formally terminate contracts using a termination interface or API. Terminated contracts MUST be marked accordingly and removed from active workflows.

[DCS-FR-CWE-24] Contract Management Dashboard
A visual dashboard MUST be available to manage and track contract progress, status, responsibilities, and deadlines. The dashboard MUST support filtering, bulk actions, and live updates.

[DCS-FR-CWE-25] Contract Review and Approval Interface
A dedicated interface MUST allow reviewers to access, validate, and comment on contracts awaiting approval. The interface MUST support highlighting, approval workflows, and role attribution.

[DCS-FR-CWE-26] Contract Signing Interface
The system MUST provide a secure signing interface where authorized signers can apply legally valid signatures. The interface MUST support wallet integration and display signer tasks.

[DCS-FR-CWE-27] Contract Tracking and Status Overview
The system MUST display real-time status of contracts, including which stage they are in (draft, review, signing, active, expired, etc.), with timestamps and action history.

[DCS-FR-CWE-28] Automated Contract Interaction via API
The system MUST offer API endpoints for external systems to create, update, or query contracts. API interactions MUST enforce authentication, rate limits, and action validation.

[DCS-FR-CWE-29] Multi-Contract Visualization
The system MUST enable visual composition and management of complex contract packages, including subcontracts, annexes, and related documents. The visualization MUST show hierarchy and dependency links.

[DCS-FR-CWE-30] Contract Package Bundling
The system MUST allow bundling of multiple related contracts into a single distributable package. Each package MUST maintain internal references, shared metadata, and signature states.

[DCS-FR-CWE-31] Contract Performance Tracking
The system MUST track key performance indicators defined in the contract such as delivery timelines, milestones, and financial terms. Alerts MUST be raised for underperformance or missed targets.

3.2.3 Signature Management

[DCS-FR-SM-01] Level of Assurance Flexibility for Simple Electronic Signature, Advanced Electronic Signature, and Qualified Electronic Signature
The system MUST support flexible signature levels in accordance with the eIDAS Regulation, including SES, AES, and QES. Each level MUST be selectable based on contract requirements and risk profiles. This ensures compatibility with diverse signing needs while maintaining compliance.

[DCS-FR-SM-02] Support for PAdES, JAdES, and CAdES Signatures
The system MUST support multiple advanced digital signature formats, including PAdES for PDFs, JAdES for JSON-based data, and CAdES for CMS structures. These formats MUST be available for respective contract representations to support cross-system interoperability and legal recognition.

[DCS-FR-SM-03] Signing Identity and PoA Authorization Credentials
Each Contract Signer MUST present a valid identity credential and (if applicable) a PoA credential. These credentials MUST be verifiable and issued by recognized authorities. The system MUST validate both the signer's identity and their authorization to act on behalf of an organization.

[DCS-FR-SM-04] Counterparty Authorization and PoA Credential Chain Verification
The system MUST verify that counterparties possess valid PoA credentials and that the delegation chain is valid and traceable. Credential chains MUST be anchored in trusted registries or via verifiable credentials to ensure authenticity.

[DCS-FR-SM-05] Integration with Signing Identity and PoA Verifiable Credentials
The system MUST integrate with Verifiable Credential frameworks to validate and consume identity and PoA credentials. Credentials MUST be issued, presented, and verified in compliance with W3C and eIDAS-compatible data models.

[DCS-FR-SM-06] Wallet for Identity, PoA Credential Management, and Signing
Contract Signers MUST be able to manage their identity and PoA credentials within a secure digital wallet (e.g., XFSC OCM, Cloud PCM). The wallet MUST support credential presentation and digital signature operations.

[DCS-FR-SM-07] Multi-Signature and Role-Based Signing Flows
The Signature Management system MUST support contracts requiring multiple signatures, where each signatory has a distinct role. The system MUST enforce the correct order, dependencies, and role-specific conditions within the signing workflow.

[DCS-FR-SM-08] Persisted Contract Signing Summary with Verifiable Credential and PDF/A-3 Embedding
Each signed contract MUST include a persisted signing summary. The summary MUST be available as a VC and embedded within the PDF/A-3 document to ensure independent verifiability and auditability.

[DCS-FR-SM-09] Secure Human-Readable Contract Viewer
The system MUST provide a secure, tamper-proof viewer for human-readable contract content. This viewer MUST be used by signers to inspect contract terms before signing and MUST guarantee that no modifications can be made during the viewing session.

[DCS-FR-SM-10] Proof of Contract Execution
After all required signatures are collected, the system MUST generate cryptographic proof of contract

execution. This proof MUST include hash references, timestamps, signer identities, and status confirmations.

[DCS-FR-SM-11] Linked Machine-Readable and Human-Readable Signatures
Each digital signature MUST be linked to both the machine-readable and human-readable versions of the contract. The system MUST ensure that both representations reflect the same content hash to guarantee consistency and prevent tampering.

[DCS-FR-SM-12] Contract Deployment Trigger
Upon completion of the signing process, the system MUST automatically trigger contract deployment to connected target systems. The trigger MUST include the signed contract and relevant metadata such as hash, version, and timestamp.

[DCS-FR-SM-13] Signature Workflow Process
The system MUST orchestrate a structured signature workflow, managing signatory assignment, status tracking, retries, and completion validation. The workflow MUST enforce order, deadlines, and dependencies defined in the contract.

[DCS-FR-SM-14] Signature Request from Signer
The system MUST allow designated signers to request a signature step via wallet, email, or integration interfaces. Requests MUST only be valid if the signer's role and authorization are verified.

[DCS-FR-SM-15] Contract Retrieval for Signing
The system MUST allow authorized signers to securely retrieve the contract they are asked to sign. The retrieval MUST be cryptographically validated and logged with timestamp, signer ID, and contract ID.

[DCS-FR-SM-16] Apply Digital Signature (via Cloud PCM or OCM Signer API Endpoint)
The system MUST allow digital signatures to be applied via an integrated signing service, such as Cloud PCM or OCM Signer API. The system MUST ensure secure key usage and enforce signature integrity validation upon signing.

[DCS-FR-SM-17] Multi-Signer Support
The SM module MUST allow multiple users to sign the same contract, either sequentially or in parallel, based on the workflow configuration. Each signature MUST be independently verifiable.

[DCS-FR-SM-18] Signature Validation
The system MUST provide tools to validate digital signatures applied to a contract, including credential status checks, cryptographic integrity, and timestamp verification. Validation results MUST be exportable for compliance purposes.

[DCS-FR-SM-19] Audit Log for Signatures
All signature actions MUST be logged in an immutable audit log, capturing signer ID, timestamp, credential used, and outcome (success/failure). The log MUST be available to auditors and compliance officers.

[DCS-FR-SM-20] Signature Revocation
The system MUST support revocation of digital signatures in case of credential invalidation or organizational revocation policies. Revocation MUST be logged and MUST invalidate the associated contract until re-signing occurs.

[DCS-FR-SM-21] Signature Compliance Verification
The system MUST assess each signature’s compliance with legal and organizational signature policies, including signature type (QES, AES), credential status, and associated roles. The system MUST flag any policy violations.

[DCS-FR-SM-22] Signature Dashboard for Contract Signers
A dashboard MUST be available to Contract Signers showing the status of pending, completed, and revoked signatures. The dashboard MUST also display associated credentials, timestamps, and validation results.

[DCS-FR-SM-23] Signing Interface
The system MUST provide a secure and user-friendly interface for applying digital signatures. The interface MUST support wallet-based signing, biometric confirmation (if applicable), and real-time validation feedback.

[DCS-FR-SM-24] Signature Status Tracking
The system MUST allow Contract Managers and Signers to track the real-time status of signature progress, including pending actions, completion timestamps, and signer acknowledgements.

[DCS-FR-SM-25] Automated Signature Processing API
The system MUST provide an API for triggering automated signature operations, allowing external systems to initiate and complete digital signatures based on pre-authorized credentials.

[DCS-FR-SM-26] Signature Compliance Viewer
The system MUST offer a viewer that displays signature metadata and compliance status, including signer identity, role, credential chain, timestamp, and cryptographic integrity proof.

[DCS-FR-SM-27] Support for PDF/A Format
The system MUST ensure that all signed contracts are exportable in PDF/A format with embedded metadata and signature containers. This format MUST comply with long-term archival and regulatory requirements.

3.2.4 Contract Storage and Archive

[DCS-FR-CSA-01] Tamper-Proof Contract Storage
The system MUST ensure contracts are stored in a tamper-evident format, using cryptographic mechanisms (e.g., hashing) to detect any unauthorized changes after storage. All modifications MUST be prohibited or logged with full traceability.

[DCS-FR-CSA-02] Role-Based Access Control
Access to stored contracts MUST be controlled based on assigned roles. Only authorized roles (e.g., Contract Manager, Legal Officer) MAY retrieve, archive, or delete stored contracts. Access attempts MUST be audited.

[DCS-FR-CSA-03] Proof-of-Existence
The system MUST generate a verifiable proof-of-existence for each archived contract. This proof MAY include a cryptographic hash, timestamp, and optional anchoring on a distributed ledger for independent verification.

[DCS-FR-CSA-04] Contract Expiry & Renewal Tracking
The system MUST monitor contract expiration timelines and support tracking of renewal status. Alerts MUST be generated as contracts approach expiration based on configurable thresholds.

[DCS-FR-CSA-05] Hierarchical Contract Storage
Contracts MUST be stored in a structured hierarchy, supporting nesting of frame agreements, sub-contracts, and appendices. Relationships between contract components MUST be preserved in metadata.

[DCS-FR-CSA-06] Machine-Readable Contract Storage
Machine-readable versions of contracts (e.g., JSON-LD, XML) MUST be stored alongside human-readable documents. The system MUST validate synchronization between both formats before archival.

[DCS-FR-CSA-07] Automated Compliance Checks
Before contracts are archived, the system MUST perform automated compliance checks based on configured business rules or regulations. Non-compliant contracts MUST be flagged for review or prevented from storage.

[DCS-FR-CSA-08] Store Signed Contract in Archive
Upon completion of the signature workflow, the system MUST automatically store the finalized contract and all signature data in the archive, ensuring document integrity and preservation of all verifiable metadata.

[DCS-FR-CSA-09] Generate and Assign Contract Identifier
Each archived contract MUST be assigned a globally unique identifier (UUID or DID) for referencing across workflows and systems. This ID MUST be persistent and immutable.

[DCS-FR-CSA-10] Index Contract Metadata
Archived contracts MUST be indexed with metadata fields such as parties, contract type, status, jurisdiction, and validity period. Metadata indexing MUST support efficient searching and filtering.

[DCS-FR-CSA-11] Create Contract Summary and Tags
The system MUST allow automatic or manual generation of a summary for each archived contract. Users MUST also be able to assign tags for thematic categorization and discovery.

[DCS-FR-CSA-12] Retrieve Contract from Archive
Authorized users MUST be able to retrieve contracts using metadata filters, contract ID, or associated tags. Retrieval operations MUST be audited and follow access control policies.

[DCS-FR-CSA-13] Search Contracts
The system MUST include a full-text and metadata-based search function. Users MUST be able to search by content, participants, dates, tags, and custom fields across all archived contracts.

[DCS-FR-CSA-14] Contract Expiration Handling
Expired contracts MUST be flagged in the system and removed from active workflows. The system MUST support retention according to configured retention policies and prevent expired contract usage.

[DCS-FR-CSA-15] Contract Renewal and Extension
The system MUST support creation of renewal or extension contracts linked to archived originals. Renewals MUST retain references to the prior contract’s version, ID, and signatures.

[DCS-FR-CSA-16] Contract Termination
Terminated contracts MUST be marked as such in the archive with a recorded termination reason, effective date, and initiating role. Terminated contracts MUST remain accessible in read-only mode.

[DCS-FR-CSA-17] Contract Deletion
If permitted by policy, authorized users MUST be able to delete archived contracts. Deletion operations MUST require justification and MUST be logged with timestamp and user identity.

[DCS-FR-CSA-18] Audit Log for Contract Storage and Retrieval
Every archival and retrieval operation MUST be recorded in an immutable audit log. Each log entry MUST include actor, timestamp, operation type, contract ID, and success/failure outcome.

[DCS-FR-CSA-19] Compliance Verification for Archived Contracts
The system MUST provide tools to verify whether archived contracts meet predefined compliance requirements (e.g., retention time, signature policies, metadata completeness). Non-compliant entries MUST be flagged.

[DCS-FR-CSA-20] Automated Contract Monitoring and Alerts
The archive MUST include rules-based monitoring for contract status and metadata (e.g., expired, renewal due). Alert notifications MUST be configurable and delivered via UI, email, or API.

[DCS-FR-CSA-21] Contract Archive Dashboard
A dashboard MUST provide an overview of archived contract statistics, recent actions, storage volume, expiring contracts, and compliance status. The dashboard MUST support drill-down into contract details.

[DCS-FR-CSA-22] Contract Search Interface
The system MUST include a dedicated search interface for archived contracts. It MUST support advanced filtering, saved queries, and export of search results for compliance or legal use.

[DCS-FR-CSA-23] Contract Expiration and Renewal Management UI
The system MUST provide a visual interface for monitoring and managing contract expirations and renewal tasks. It MUST allow bulk actions and notification management for contract owners.

[DCS-FR-CSA-24] Contract Compliance and Audit Viewer
A viewer MUST allow auditors to inspect contracts, associated metadata, compliance status, and audit logs from a single interface. This viewer MUST support export to standard formats for external audits.

[DCS-FR-CSA-25] Contract Processing API
The system MUST expose APIs for contract archival, metadata updates, tagging, and retrieval. APIs MUST require authorization and include audit trail generation for each interaction.

[DCS-FR-CSA-26] Archive Multi-Party Contract Component Assignments
For contracts involving multiple parties, each party's assigned sections MUST be individually archived and linked to the overall contract package. The system MUST allow per-party access restrictions to their respective sections.

3.2.5 Process Audit & Compliance

[DCS-FR-PACM-01] Tamper-Proof Audit Trail for Contract Lifecycle
The system MUST maintain a tamper-proof audit trail capturing all contract lifecycle events – including creation, editing, submission, review, signing, renewal, and termination. Each log entry MUST include a timestamp, actor identity, action taken, and affected contract component. Audit logs MUST be immutable and exportable for forensic review.

[DCS-FR-PACM-02] Compliance Monitoring and Risk Detection
The system MUST continuously monitor contract lifecycle activities for violations of defined compliance rules (e.g., missing approvals, expired credentials, unauthorized access). Detected risks MUST be flagged and reported in real-time via dashboards and alerts.

[DCS-FR-PACM-03] Automated Regulatory and Policy Compliance Checks
The system MUST perform automated checks during contract workflows to ensure compliance with regulatory frameworks (e.g., eIDAS, GDPR, ISO) and internal policies. Contracts failing these checks MUST be blocked from execution or flagged for manual review.

[DCS-FR-PACM-04] Role-Based Access Control for Audit Logs
Access to audit logs MUST be restricted based on roles such as Compliance Officer, Auditor, or Admin. Each access MUST be logged and include a justification for traceability. Unauthorized access attempts MUST be blocked and logged.

[DCS-FR-PACM-05] Contract Non-Compliance Investigation and Reporting
The system MUST include tools for investigating non-compliance events, such as incomplete workflows, late signatures, or invalid credentials. Investigators MUST be able to generate detailed compliance reports and export case files for regulatory review.

[DCS-FR-PACM-06] Structural Integrity Validation for Multi-Contract Packages
The system MUST validate the structural integrity of multi-contract packages to ensure completeness, logical correctness, and proper linkage between components (e.g., main contract, annexes, sub-agreements). Missing or misconfigured components MUST be flagged before execution.

[DCS-FR-PACM-07] Compliance Reporting by Contract Component and Party
The system MUST generate compliance reports segmented by individual contract components (e.g., clauses, appendices) and by involved parties. Each report MUST include compliance status, timestamp, non-conformance indicators, and relevant credential metadata.

3.3 Other Nonfunctional Requirements

3.3.1 Performance Requirements

[DCS-NFR-PER-01] Performance by Design
Every component SHOULD be designed and implemented with performance in mind. They MUST particularly be implemented in a non-blocking way.
- Measurement: Response time under load, throughput metrics.
- Verification Method: Review of documentation and testing.

[DCS-NFR-PER-02] Scalability
Every component MUST be scalable and able to handle increased load and users without performance degradation. This allows the system to handle growing demand and multi-requests.
- Measurement: System scalability test results, number of supported users & connections.
- Verification Method: Review of documentation and testing.

[DCS-NFR-PER-03] Availability & Resilience
It MUST be ensured the DCS system is always available and can recover from failures.
- Measurement: Uptime percentage, MTTR (Mean Time to Recovery).
- Verification Method: Review of Documentation, Testing.

3.3.2 Safety Requirements

[DCS-NFR-SF-01] Reset Possibility
In case of errors, it MUST be possible to reset the component and continue execution as specified in this document. The component SHOULD be stateless and need no recovery in case of a reset to maintain system stability and reliability during errors.
- Measurement: System recovery success rate, mean time to recover (MTTR).
- Verification Method: Review of Documentation and Testing.

[DCS-NFR-SF-02] Remote Administration
If the component can be remotely administrated by the Federator, the communication MUST utilize a secure communication channel such as SSH or VPN.

- Measurement: Remote access security test results, penetration test findings.
- Verification Method: Review of Documentation, Testing.

[DCS-NFR-SF-03] Business Continuity & Disaster Recovery
To prevent business disruptions and ensure high availability, The DCS MUST ensure data resilience and rapid recovery in case of failures or cyber incidents.
- Measurement: Recovery Time Objective (RTO) and Recovery Point Objective (RPO) metrics.
- Verification Method: Review of Documentation, Testing.

3.3.3 Security Requirements

[DCS-NFR-SEC-01] Transport Layer Security
To ensure secure communication and data integrity, each communication with an interface of DCS MUST utilize TLS 1.3. It MUST NOT use SSL 3.0, TLS 1.0 and 1.1.
- Measurement: TLS version compliance, security audit results.
- Verification Method: Review of Documentation and Testing.

[DCS-NFR-SEC-02] State-of-the-art Cryptography
Cryptography Cryptographic algorithms and cipher suites MUST be state-of-the-art and chosen in accordance with official recommendations. Those recommendations MAY be those of the German Federal Office for Information Security (BSI) or SOG-IS.
- Measurement: Encryption compliance with standards, cryptographic algorithm strength.
- Verification Method: Review of Documentation and Testing.

[DCS-NFR-SEC-03] Authentication and Authorization
DCS MUST grant access to its services only to authenticated and authorized users. It MUST implement RBAC and enforce least privilege principles for human and machine users of the service. RBAC MUST be managed via role assertions in the form of a verifiable credential stored in the wallets of users and presented during the authentication and authorization process of a user.
- Measurement: Authentication success rate, number of unauthorized access attempts.
- Verification Method: Review of Documentation and Testing.

[DCS-NFR-SEC-04] Integrity Protection for Configuration
Where the functionality of the DCS is based on configuration files, those files MUST be authenticated, and integrity protected.
- Measurement: Configuration integrity verification logs.
- Verification Method: Review of Documentation and Testing.

[DCS-NFR-SEC-05] Integrity Protection for Service
The Federator MUST utilize security measures to ensure the integrity of DCS. It MAY support proof of the integrity of remote parties using an additional interface (Remote Attestation).
- Measurement: Integrity verification test results, number of detected tampering attempts.
- Verification Method: Review of Documentation, Testing.

[DCS-NFR-SEC-06] Storage of Secrets
Secrets such as keys and other cryptographic material MUST be stored in a secure and protected environment, e.g., a TPM, HSM, or TEE to ensure their confidentiality and integrity.
- Measurement: Secure storage compliance tests, key access logs.
- Verification Method: Review of Documentation and Testing.

[DCS-NFR-SEC-07] Testing
The development of DCS MUST include functional and security testing, source code audits, and penetration testing.

- Measurement: Test coverage reports, number of security vulnerabilities found.
- Verification Method: Review of Documentation.

[DCS-NFR-SEC-08] Confidentiality
The data in the DCS MUST be protected from unauthorized access during storage and transmission.
- Measurement: Number of data breaches.
- Verification Method: Review of Documentation and Testing.

[DCS-NFR-SEC-09] Monitoring, Logging & Auditability
Logs MUST be securely stored and accessible for audits to allow for proactive issue detection, performance optimization, and forensic analysis in case of security incidents.
- Measurement: Availability of logs, number of detected anomalies, and system uptime.
- Verification Method: Review of Documentation and Testing.

[DCS-NFR-SEC-10] Data Integrity
The DCS MUST ensure that data remains unchanged and verifiable over its lifecycle and the service MUST be protected against unauthorized data modifications.
- Measurement: Integrity hash checks, number of detected modifications.
- Verification Method: Review of Documentation, Testing.

[DCS-NFR-SEC-11] Monitoring & Incident Response
The DCS MUST ensure continuous monitoring and automated incident response mechanisms to enable proactive detection of anomalies and security incidents.
- Measurement: Number of detected anomalies, mean time to detect (MTTD).
- Verification Method: Review of Documentation, Testing.

[DCS-NFR-SEC-12] Secure Configuration Management
The DCS MUST ensure that system configurations are stored securely and protected from unauthorized modifications to prevent security misconfigurations.
- Measurement: Configuration compliance score.
- Verification Method: Review of Documentation, Source Code Review.

[DCS-NFR-SEC-13] Secure Data Disposal
The DCS MUST prevent data leaks and compliance violations and it MUST ensure that sensitive data is properly deleted when no longer needed.
- Measurement: Number of completed secure deletion tasks.
- Verification Method: Review of Documentation, Testing.

[DCS-NFR-SEC-14] Data Encryption at Rest & In Transit
The system MUST ensure all sensitive data is encrypted both in storage and during transmission to protect data from unauthorized access.
- Measurement: Encryption coverage percentage.
- Verification Method: Review of Documentation, Testing.

[DCS-NFR-SEC-15] Secure Software Development Lifecycle (SDLC)
To reduce vulnerabilities in the codebase, the DCS SHOULD enforce secure coding practices and security testing throughout the development lifecycle.
- Measurement: Number of security issues detected in code reviews.
- Verification Method: Review of Documentation, Source Code Review.

[DCS-NFR-SEC-16] Identity Federation
DCS MUST enable seamless user authentication across multiple systems and support interoperability with third-party identity providers and authentication frameworks for access.

- Measurement: Number of successful federated logins.
- Verification Method: Review of Documentation and Testing.

[DCS-NFR-SEC-17] Secure Boot & Hardware Security
To prevent unauthorized firmware or OS modifications, the DCS MUST ensure that the system only runs trusted software through secure boot mechanisms.

- Measurement: Number of successful secure boot validations.
- Verification Method: Review of Documentation and Testing.

[DCS-NFR-SEC-18] Selective Disclosure for Privacy
By default, DCS MUST enforce selective disclosure of attributes, sharing only the minimum necessary information. Machine-to-Machine interactions in the name of a user MUST require explicit and verifiable user consent prior to execution.

- Measurement: Number of data disclosures aligned with selective disclosure policy, number of M2M transactions executed with recorded consent, audit results of consent management.
- Verification Method: Review of Documentation, Testing, and Consent Audit Logs.

3.3.4 Software Quality Attributes

[DCS-NFR-SQ-01] Programming Style
The implementation SHOULD follow best practices and a consistent style for coding, e.g., the source code SHOULD be clearly structured and modularized; there SHOULD be no dead code; function and variables SHOULD be clear and self-explaining. The code MUST be well documented to support adaptability, maintainability, and usability of the component.

- Measurement: Code review and audit results, documentation completeness.
- Verification Method: Review of Documentation, Source Code Review.

[DCS-NFR-SQ-02] Build Scripts
The repository for the DCS MUST include build scripts to build and run the Service from the repository.

- Measurement: Automated build success rates, deployment consistency metrics.
- Verification Method: Review of Documentation, Testing**.**

[DCS-NFR-SQ-03] Containerized Deployment
The software MUST be containerized using industry-standard containerization technologies (e.g., Docker, Podman) to ensure seamless deployment, portability, and runtime consistency across container orchestration platforms such as Kubernetes and OpenShift.

- Measurement: Successful deployment and execution in containerized environments, compatibility with Kubernetes and container orchestration tools, and automated CI/CD integration.
- Verification Method: Review of Documentation, Deployment Testing.

[DCS-NFR-SQ-04] Privacy by Design
To ensure compliance with privacy regulations, the DCS SHOULD embed privacy-enhancing technologies into system architecture and data processing.

- Measurement: Privacy impact assessment results.
- Verification Method: Review of Documentation**.**

[DCS-NFR-SQ-05] Non-Repudiation
The system MUST ensure that digital signatures and logs can be used as legal evidence.

- Measurement: Number of legally recognized signed transactions.
- Verification Method: Review of Documentation, Testing.

[DCS-NFR-SQ-06] System Interoperability
To facilitates seamless data exchange and integration, the system MUST ensure compatibility with external platforms, cloud providers, and enterprise systems.

- Measurement: Number of successfully integrated third-party systems.
- Verification Method: Review of Documentation, Testing.

[DCS-NFR-SQ-07] Usability & Accessibility

The DCS MUST ensure that the system is easy to use and accessible.

- Measurement: Compliance with WCAG and usability testing results.
- Verification Method: Review of Documentation, Testing.

[DCS-NFR-SQ-08] Orchestration Layer
To ease the integration of XFSC services and components, the DCS MUST be integrated with the FACIS Orchestration Engine which provides access to XFSC Cloud PCM, OCM, Catalogue, and Trust Module.

- Measurement: Working integration of FACIS orchestration engine and underlying XFSC services.
- Verification Method: Review of Documentation, Testing.

3.4 Business Rules

[DCS-NFR-BR-01] Strong Authentication & Role Binding
Access to the DCS MUST only be possible with two-factor authentication and VCs from recognized organizational/legal-person wallets. Functions MAY only be executed by users or system roles that have been explicitly authorized for that action.

[DCS-NFR-BR-02] Participant Eligibility
Only recognized ecosystem participants proven via organizational/legal-person wallets MAY interact with the DCS. Unverified entities MUST NOT obtain access or interact with contract workflows.

[DCS-NFR-BR-03] Legally Valid Signatures
Contracts requiring legal enforceability MUST be signed with AES by default and with QES or seals where mandated. Contracts lacking the required signature level MUST NOT proceed to deployment or execution states.

[DCS-NFR-BR-04] Template Governance
Every contract MUST originate from an approved, versioned template in the Template Repository. Unapproved templates MUST NOT be usable, and template changes MUST be versioned and traceable.

[DCS-NFR-BR-05] Immutable Auditability
All contract actions (creation, negotiation, review, approval, signing, archival, revocation) MUST produce immutable, tamper-evident audit records with timestamps and actor identity. Access to audit trails MUST be restricted to authorized roles (e.g., Auditor, Compliance Officer).

[DCS-NFR-BR-06] Revocation & Termination Propagation
Revocation of credentials, signatures, or contracts MUST take immediate effect and be propagated across dependent systems; terminated contracts MUST be archived with full evidence preserved.

[DCS-NFR-BR-07] Token & API Control
Integration tokens (including any logging or “log” tokens) and API credentials MUST only be issued to verified participants and MUST authorize only the minimum necessary scopes for the intended workflow.

[DCS-NFR-BR-08] DCS-to-DCS Interoperability Safeguards
DCS-to-DCS exchanges (create offers, status updates, revocations) MUST occur only over authenticated/authorized APIs between verified parties, with full traceability and audit logs.

[DCS-NFR-BR-09] Catalogue-Aligned Publishing
Publishing or requesting templates via a federated catalogue MUST follow the catalogue's governance and discovery rules; only approved artifacts may be published or consumed.

3.5 Compliance

[DCS-NFR-COMP-01] Legal Compliance
The DCS MUST ensure compliance with national and European laws for system operations. The DCS MUST adhere to international regulations.

[DCS-NFR- COMP-02] EUCS/ENISA Compliance
The DCS SHOULD fulfill the cybersecurity control set of the EUCS Annex A according to its assigned Assurance Level to meet cybersecurity compliance standards for assurance levels.

[DCS-NFR- COMP-03] GDPR Compliance
The DCS MUST ensure audit logs and compliance against eIDAS/EUDI logging regulations, the eIDAS Regulation (910/2014 & upcoming eIDAS 2.0), the General Data Protection Regulation (GDPR, Regulation 2016/679), and relevant ETSI and ISO standards (e.g., ETSI EN 319 401, ISO/IEC 27001).

3.6 Design and Implementation

3.6.1 Installation

The DCS MUST provide build and deployment scripts to ensure reproducible installation on the target platform. The repository MUST include container definitions (e.g., Dockerfiles, Helm charts) that enable automated setup of the DCS components. All installation procedures MUST ensure secure integration with external identity providers, wallets, and catalogues. Installation verification MUST be possible through automated test suites and documentation review.

3.6.2 Distribution

The DCS MUST support deployment in geographically distributed environments, including multi-cloud and hybrid infrastructures. Contract data and audit logs MUST be securely replicated across nodes while preserving confidentiality and integrity. Distribution MUST respect regulatory requirements (e.g., data localization) and MUST ensure that contract state synchronization between DCS instances is consistent and verifiable.

3.6.3 Maintainability

The DCS architecture MUST be modular, with each core component (TR, CWE, SM, CSA, PACM) implemented as a separate service with well-defined interfaces. Each service MUST expose standardized APIs (REST/JSON-LD, gRPC optional) to allow independent updates and maintenance. The codebase SHOULD follow clean code principles, and documentation MUST be maintained to describe component responsibilities, dependencies, and upgrade procedures.

3.6.4 Reusability

The DCS MUST maximize reusability of its components across different use cases and ecosystems. Contract templates, compliance policies, and audit modules MUST be designed for reuse across multiple domains (e.g., supply chain, pharmaceuticals, legal). Interfaces MUST allow integration into existing orchestration workflows without reimplementation.

3.6.5 Portability

The DCS MUST be portable across major operating systems (Linux, Windows) and cloud platforms (Kubernetes, Docker Swarm). No component MUST rely on vendor-specific features without providing a fallback. The system SHOULD support deployment in both centralized and federated environments, with minimal configuration changes.

3.6.6 Cost

The DCS MUST be designed for cost efficiency in deployment and operation. Open-source components SHOULD be used where possible, provided they meet compliance requirements. Licensing costs MUST be documented, and scaling strategies (e.g., autoscaling services, serverless functions for verification) SHOULD minimize cloud infrastructure costs.

3.6.7 Deadline

Delivery of the DCS software MUST follow project schedule constraints. Alpha prototypes MUST be delivered within the proof-of-concept phase, with a stable beta including all five core components before production readiness. Milestones MUST align with consortium deliverables and external regulatory deadlines (e.g., ESPR, eIDAS 2.0 compliance dates). All items listed in Appendix BSHOULD be resolved before the proof-of-concept phase with documented evidence of resolution stored in the project repository.

3.6.8 Proof of Concept

The DCS MUST provide a proof-of-concept deployment that demonstrates end-to-end contract lifecycle management, including template creation, negotiation, QES signing, policy validation, and archival. The PoC MUST validate interoperability between at least two DCS instances and integration with external components (e.g., ERP, catalogue, organizational wallets). Successful execution of the PoC MUST serve as the baseline for final product implementation.

4 System Features

This chapter outlines DCS system features, which correspond to a specific set of use cases. Each system feature is described with its associated use cases, priority, stimulus/response sequences, and functional requirements.

4.1 UC-01 User Authentication & Authorization

4.1.1 Description and Priority

Priority: High
This feature ensures secure authentication and role-based access to the DCS system. User authentication and authorization is required for all human and machine users interacting with the system.

4.1.2 Stimulus/Response Sequences

Stimulus: A user (all human or system user roles as outlined in Chapter 2.4) attempts to access the DCS system by presenting a verifiable credential that includes role-based authorization data.

Response: The system verifies the authenticity of the credential, confirms the user's role, and grants access to the appropriate resources within the DCS system according to the role-based permissions. If the credential is invalid (expired, signature broken, revoked), access policies are not fulfilled or the user's access has been revoked, the system denies access and logs the attempt for audit and security monitoring.

4.1.3 Functional Requirements

[DCS-FR-UC-01-1] Authentication & Credential Handling

FR-UC-01-1 is linked to the following functional requirements listed in Section 3.2:

- FR-SM-03 – Signing Identity & PoA Authorization Credentials
- FR-SM-05 – Integration with Signing Identity and PoA VCs
- FR-SM-06 – Wallet for Identity, PoA Credential Management, and Signing
- FR-SM-08 – Persisted Contract Signing Summary with VC and PDF/A-3 Embedding
- FR-SM-16 – Apply Digital Signature (for “role-based signing” using PoA authorisation credentials)

[DCS-FR-UC-01-2] Authorization & Access Control

FR-UC-01-2 is linked to the following functional requirements listed in Section 3.2:

- FR-CSA-02 – Role-based Access Control for Contract Storage and Archive
- FR-CWE-07 – Role-Based Access Control for Contract Workflow Engine
- FR-PACM-04 – Role-Based Access Control for Audit Logs
- FR-TR-06 – Role-Based Access Control for Template Repository
- Role Based Access Control for EUDI Identity Holders and Validation

[DCS-FR-UC-01-3] Role Enforcement

FR-UC-01-3 is linked to the following functional requirements listed in Section 3.2:

- FR-TR-02 – Multi-Tiered Contract Template Management
- FR-TR-09 – Template Provenance and Versioning
- FR-SM-13 – Signature Workflow Process

[DCS-FR-UC-01-4] Error Handling & Invalid Input Responses

- The system MUST reject expired, revoked, or malformed credentials and return an error message: "Credential invalid or access revoked."
- Unauthorized access attempts MUST be logged with user identifier (if available), timestamp, and targeted resource.
- If wallet integration fails (e.g., signer unavailable), the system MUST notify the user and allow retry or fallback (TBD).
- Multiple invalid Credential Attempts MUST block out the user.

4.2 UC-02 Contract Template Management

4.2.1 Description and Priority

Priority: High

This feature handles the lifecycle of single and multi-contract templates. The associated use cases of this feature are listed as follows:

- UC-02-01 – Create Contract Template: Allows Template Managers or Approvers to create reusable contract templates.
- UC-02-02 – Search and Retrieve Contract Templates: Enables users to search and access existing contract templates.
- UC-02-03 – Generate Contract from Template: Automatically fills a contract template with relevant data.
- UC-02-04 – Update Contract Template: Modifies existing contract templates for updates or improvements.
- UC-02-05 – Deprecate Contract Template: Marks outdated contract templates as deprecated.
- UC-02-06 – Add Template Provenance Information: Ensures traceability by adding metadata and origin details to contract templates.
- UC-02-07 – Verify Template & Provenance: Validates template correctness, meta data, semantics (JSON-LD context, SHACL) and the authenticity and source of contract templates before use.
- UC-02-08 – Create and Maintain Semantic Schemas: Develops and manages semantic structures for contract templates.
- UC-02-09 – Check Template Management Dashboard for Status: Allows users to track template progress, approvals, and execution status.

4.2.2 Stimulus/Response Sequences

1. Stimulus: A Template Creator submits a request to create a new contract template.
   Response: The system initiates the template creation workflow, stores the draft, and assigns appropriate roles for further review and approval.
2. Stimulus: A Template Reviewer performs a search query to locate an existing contract template.
   Response: The system filters and returns templates matching the search criteria, based on the user's role and access rights.
3. Stimulus: A Template Approver selects a template to generate a contract instance.
   Response: The system generates a contract by populating the selected template with context-specific data and metadata.
4. Stimulus: A Template Creator initiates a request to update an existing contract template.
   Response: The system validates edit permissions, enables editing, and records the version history for audit and rollback.
5. Stimulus: A Template Reviewer marks a template as deprecated.
   Response: The system changes the template status to "Deprecated", prevents new contract generation from it, and logs the action with timestamp and user ID.
6. Stimulus: A Template Approver submits provenance metadata for a contract template.
   Response: The system records the origin details, contributors, timestamp, and unique identifiers as part of the template's metadata.
7. Stimulus: A Template Manager requests verification of a template's provenance.
   Response: The system verifies the authenticity and integrity of the template's origin data using digital signatures or verifiable credentials.
8. Stimulus: A Template Manager creates or updates semantic schemas for use in contract templates.
   Response: The system validates and stores the schema, linking it to compatible templates and enforcing schema conformity during template creation.
9. Stimulus: A Template Manager opens the template management dashboard.
   Response: The system displays real-time status of templates, including approval stage, usage metrics, and historical changes.

4.2.3 Functional Requirements

[DCS-FR-UC-02-1] Contract Template Management

FR-UC-02-1 is linked to the following functional requirements listed in Section 3.2:

- FR-TR-02 – Multi-Tiered Contract Template Management
- FR-TR-13 – Template Creation
- FR-TR-14 – Template Submission for Approval
- FR-TR-15 – Template Approval Process
- FR-TR-16 – Template Update Management
- FR-TR-17 – Template Retirement and Deprecation
- FR-TR-18 – Template Deletion

4.3 UC-03 Contract Creation

4.3.1 Description and Priority

Priority: High

This feature covers the end-to-end contract generation process for both single contracts and multi contracts. The associated use cases of this feature are listed as follows:

- UC-03-01 – Create Contract: Enables Contract Managers to create contracts based on predefined templates.
- UC-03-02 – Negotiate Contract Terms: Facilitates discussions and modifications to contract clauses before finalization.
- UC-03-03 – Adjust Contract Terms: Allows specific contract clauses to be edited without regenerating the entire contract.
- UC-03-04 – Approve Contract: Gathers required approvals from designated contract approvers before signing.
- UC-03-05 – Review Machine-Readable and Human-Readable Contract Correctness and Versions: Ensures both machine-readable and human-readable versions are accurate, validated and consistent.
- UC-03-06 – Manage Contract Signing Process: Oversees and coordinates the structured signing process for all parties.
- UC-03-07 – Check Contract Management Dashboard for Status and Search Contracts: Allows users to track contract progress, approvals, and execution status, as well as search for individual contracts.

4.3.2 Stimulus/Response Sequences

1. Stimulus: A Contract Creator submits a request to create a new contract.
   Response: The system generates a draft contract from a selected template, assigns necessary metadata, and allows further editing or collaboration.
2. Stimulus: A Contract Manager or Contract Reviewer opens a draft contract to negotiate specific clauses.
   Response: The system enables commenting, version tracking, and proposed edits with a negotiation log.
3. Stimulus: A Contract Manager or Contract Reviewer requests to adjust specific terms in the contract.
   Response: The system permits granular editing of clauses while maintaining document integrity and audit history.
4. Stimulus: A Contract Approver or Contract Manager initiates the approval process for a finalized contract.
   Response: The system routes the contract to required approvers, logs approvals with timestamps, and locks the content upon completion.
5. Stimulus: A Contract Creator, Contract Reviewer, or Contract Manager requests a view of the contract in both machine-readable and human-readable formats.
   Response: The system renders synchronized representations of both views, highlighting any inconsistencies or formatting errors.

6. Stimulus: A Contract Manager initiates the contract signing workflow.
   Response: The system schedules and tracks signing steps, assigns signatories, and integrates identity checks where needed.
7. Stimulus: A Contract Manager or Contract Observer checks the contract dashboard.
   Response: The system displays the real-time contract lifecycle status, approval steps, and searchable logs for completed and pending contracts.

4.3.3 Functional Requirements

[DCS-FR-UC-03-1] Contract Drafting & Creation

FR-UC-03-1 is linked to the following functional requirements listed in Section 3.2:
- FR-CWE-13 – Contract Creation
- FR-CWE-03 – Contract Assembling
- FR-CWE-30 – Contract Package Bundling

[DCS-FR-UC-03-2] Negotiation, Editing & Adjustment

FR-UC-03-2 is linked to the following functional requirements listed in Section 3.2:
- FR-CWE-08 – Version Control
- FR-CWE-14 – Contract Submission for Review
- FR-CWE-15 – Contract Review and Approval
- FR-CWE-17 – Contract Review
- FR-CWE-18 – Contract Negotiation

[DCS-FR-UC-03-3] Approval & Compliance

FR-UC-03-3 is linked to the following functional requirements listed in Section 3.2:
- FR-CWE-16 – Contract Initiation
- FR-CWE-25 – Contract Review and Approval Interface
- FR-PACM-03 – Automated Regulatory and Policy Compliance Checks
- FR-PACM-02 – Compliance Monitoring and Risk Detection

4.4 UC-04 Contract Signing

4.4.1 Description and Priority

Priority: High
This feature manages the contract signing workflow of both single contracts and multi contracts. The associated use cases of this feature are listed as follows:
- UC-04-01 – Review and Sign Contract Electronically: Reviews contract in secure contract viewer (See Section 3.1.1.3). Enables secure and legally binding digital contract signing. Adds identity and PoA credentials to the signature.
- UC-04-02 – Verify Counterparty Authorization: Ensures the counterparty has the legal authority to sign the contract.
- UC-04-03 – Verify Counterparty Contract Signature: Validates the authenticity and integrity of the counterparty's signature.

4.4.2 Stimulus/Response Sequences

1. Stimulus: A Contract Signer opens the contract in a secure document viewer to initiate signing.
   Response: The system enables secure, legally binding digital signing, including integration of identity and PoA credentials. The action is logged with timestamp and signer ID.

2. Stimulus: A Contract Signer or Contract Manager checks the signing credentials of the counterparty.
   Response: The system verifies the counterparty's legal authority using stored credentials or third-party trust anchors and notifies of any discrepancies.
3. Stimulus: A Contract Signer or Contract Manager initiates the verification of the counterparty's digital signature.
   Response: The system validates the cryptographic integrity of the signature, confirms it matches the registered signer, and confirms the document was not altered.

4.4.3 Functional Requirements

[DCS-FR-UC-04-1] Contract Signing
FR-UC-04-1 is linked to the following functional requirements listed in Section 3.2:

- FR-CWE-19 – Contract Signing
- FR-CWE-26 – Contract Signing Interface
- FR-SM-13 – Signature Workflow Process
- FR-SM-16 – Apply Digital Signature

[DCS-FR-UC-04-2] Signature Validation
FR-UC-04-2 is linked to the following functional requirements listed in Section 3.2:

- FR-SM-18 – Signature Validation
- FR-SM-21 – Signature Compliance Verification

4.5 UC-05 Contract Deployment

4.5.1 Description and Priority

Priority: Low
This feature ensures that contracts are properly deployed for execution. The associated use case of this feature is listed as follows:

- UC-05-01 – Deploy Signed Contract for Execution by a Target System: Makes the signed contract accessible for implementation. Facilitates the deployment of finalized contracts into the appropriate target system.

4.5.2 Stimulus/Response Sequences

Stimulus: A Contract Manager submits a signed contract for deployment.
Response: The system transfers the finalized contract to the designated execution environment, ensuring proper handover to the target system and confirming receipt.

4.5.3 Functional Requirements

[DCS-FR-UC-05-1] Contract Deployment
FR-UC-05-1 is linked to the following functional requirements listed in Section 3.2:

- FR-SM-12 – Contract Deployment Trigger
- FR-CWE-06 – Event-Driven Contract Execution

4.6 UC-06 Contract Lifecycle Management

4.6.1 Description and Priority

Priority: High
This feature covers the monitoring and management of single- and multi-contract execution. The associated use cases of this feature are listed as follows:

- UC-06-01 – Monitor Contract Performance: Ensures that contractual obligations are met and enforced.
- UC-06-02 – Manage Contract Renewal or Termination: Handles contract renewal and structured termination processes including the revocation of contract meta data VCs.

4.6.2 Stimulus/Response Sequences

Stimulus: A Contract Manager or Contract Observer opens a contract lifecycle dashboard to monitor performance.
Response: The system displays real-time status, flags upcoming milestones or missed deadlines, and records fulfillment of contractual terms.

Stimulus: A Contract Manager initiates a renewal or termination request.
Response: The system evaluates the contract status and triggers appropriate workflows for extension, renegotiation, or termination with logging and versioning.

4.6.3 Functional Requirements

[DCS-FR-UC-06-1] Monitoring Contract Performance
FR-UC-06-1 is linked to the following functional requirements listed in Section 3.2:

- FR-CWE-24 – Contract Management Dashboard
- FR-CWE-27 – Contract Tracking and Status Overview
- FR-CWE-31 – Contract Performance Tracking
- FR-CWE-09 – SLA & Compliance Monitoring
- FR-CSA-20 – Automated Contract Monitoring and Alerts
- FR-PACM-01 – Tamper-Proof Audit Trail for Contract Lifecycle
- FR-PACM-02 – Compliance Monitoring and Risk Detection
- FR-PACM-05 – Contract Non-Compliance Investigation and Reporting

[DCS-FR-UC-06-2] Renewal and Termination Management
FR-UC-06-2 is linked to the following functional requirements listed in Section 3.2:

- FR-CWE-11 – Contract Renewal
- FR-CWE-12 – Termination Handling
- FR-CWE-22 – Contract Renewal Management
- FR-CWE-23 – Contract Termination
- FR-CSA-04 – Contract Expiry & Renewal Tracking
- FR-CSA-14 – Contract Expiration Handling
- FR-CSA-15 – Contract Renewal and Extension
- FR-CSA-16 – Contract Termination
- FR-CSA-23 – Contract Expiration and Renewal Management UI

4.7 UC-07 Contract Storage and Security

4.7.1 Description and Priority

Priority: High
This feature covers contract archiving and security for both single contracts and multi contracts. The associated use cases of this feature are listed as follows:

- UC-07-01 – Store Contract in Secure Archive: Ensures long-term, tamper-proof storage of signed contracts.
- UC-07-02 – Manage Contract Permissions & Access: Controls user access rights and security settings for contracts.
- UC-07-03 – Check Contract Storage & Security Dashboard: Allows the archive manager to track archive progress, coverage, and status.

4.7.2 Stimulus/Response Sequences

1. Stimulus: A Contract Manager stores a signed contract in the secure archive.
   Response: The system validates the contract, timestamps the archive entry, ensures tamper-proof sealing, and stores the contract in long-term encrypted storage.
2. Stimulus: A Contract Manager configures or modifies access and permission settings for a contract.
   Response: The system updates access control rules, logs the change, and immediately enforces restrictions for all relevant users or systems.
3. Stimulus: An Archive Manager opens the contract storage and security dashboard.
   Response: The system displays contract archive status, data integrity checks, access logs, and alerts related to coverage or anomalies.

4.7.3 Functional Requirements

[DCS-FR-UC-07-1] Store Contract in Secure Archive
FR-UC-07-1 is linked to the following functional requirements listed in Section 3.2:

- FR-CSA-01 – Tamper-Proof Contract Storage
- FR-CSA-08 – Store Signed Contract in Archive
- FR-CWE-20 – Store Contract in Archive
- FR-CSA-05 – Hierarchical Contract Storage
- FR-CSA-06 – Machine-Readable Contract Storage
- FR-CSA-26 – Archive Multi-Party Contract Component Assignments

[DCS-FR-UC-07-2] Manage Contract Permissions and Access
FR-UC-07-2 is linked to the following functional requirements listed in Section 3.2:

- FR-CSA-02 – RBAC
- FR-PACM-04 – Role-Based Access Control for Audit Logs

[DCS-FR-UC-07-3] Check Contract Storage & Security Dashboard
FR-UC-07-3 is linked to the following functional requirements listed in Section 3.2:

- FR-CSA-21 – Contract Archive Dashboard

4.8 UC-08 Contract Compliance & Auditing

4.8.1 Description and Priority

Priority: High
This feature ensures compliance with regulations and standards. The associated use cases of this feature are listed as follows:

- UC-08-01 – Generate Report about Contract Activity Logs & Timestamps: Reports and records contract-related activities for auditing purposes.
- UC-08-02 – Audit Contract Compliance: Conducts compliance checks against legal and organizational frameworks.
- UC-08-03 – Audit Logs/Compliance against eIDAS/EUDI logging regulations
  - eIDAS Regulation (910/2014 & upcoming eIDAS 2.0)
  - General Data Protection Regulation (GDPR, Regulation 2016/679)
  - Relevant ETSI and ISO standards (e.g., ETSI EN 319 401, ISO/IEC 27001)

4.8.2 Stimulus/Response Sequences

1. Stimulus: An Auditor or Compliance Officer requests a report of contract activity logs.
   Response: The system compiles logs and timestamps of contract creation, edits, approvals, and signatures into an auditable report with export options.

2. Stimulus: An Auditor or Compliance Officer initiates a compliance audit.
   Response: The system evaluates contract contents and lifecycle history against predefined legal and policy criteria, flags issues, and generates a compliance summary.

4.8.3 Functional Requirements

[DCS-FR-UC-08-1] Generate Report about Contract Activity Logs & Timestamps
FR-UC-08-1 is linked to the following functional requirements listed in Section 3.2:
- FR-PACM-01 – Tamper-Proof Audit Trail for Contract Lifecycle
- FR-PACM-07 – Compliance Reporting by Contract Component and Party

[DCS-FR-UC-08-2] Audit Contract Compliance
FR-UC-08-2 is linked to the following functional requirements listed in Section 3.2:
- FR-PACM-03 – Automated Regulatory and Policy Compliance Checks
- FR-PACM-02 – Compliance Monitoring and Risk Detection
- FR-PACM-05 – Contract Non-Compliance Investigation and Reporting

4.9 UC-09 DCS Administration

4.9.1 Description and Priority

Priority: Normal
This feature covers system-wide administrative functions. The associated use cases of this feature are listed as follows:
- UC-09-01 – System Configuration & User Management: Handles role-based access, security, and system configurations.
- UC-09-02 – System Monitoring & Logging: Ensures operational monitoring and maintains security logs.

4.9.2 Stimulus/Response Sequences

1. Stimulus: A System Administrator updates role-based access permissions or modifies system settings.
   Response: The system validates administrative privileges, applies changes to user roles and configurations, and logs the update.
2. Stimulus: A System Administrator initiates monitoring or views the system logs.
   Response: The system presents operational health metrics and a searchable log of system events, security warnings, and usage statistics.

4.9.3 Functional Requirements

[DCS-FR-UC-09-1] Role-Based Access Configuration
The system MUST allow authorized system administrators to configure RBAC settings including de-actiation or re-activation of accounts. This includes the ability to add, edit, or remove user roles, assign permissions, and define and change access scopes for each role.

[DCS-FR-UC-09-2] System Logging and Monitoring
The system MUST monitor critical system activities and log all security-related events (e.g., access attempts, configuration changes, failures) with accurate timestamps and actor identification.

- Each user authentication and authorization will be logged
- Deactivation of an Account
- Template & Contract Life-Cycle Events
- Each Admin action must be logged for Audit Trails

4.10 UC-10 Contract Automation & Integration

4.10.1 Description and Priority

Priority: High
This feature ensures seamless automation and integration with external systems. The associated use cases of this feature are listed as follows:

- UC-10-01 – Automate Contract Workflow Processes: Integrates contract workflows into AI/ERP systems.
- UC-10-02 – Validate Contract Integrity & Compliance: Performs automated contract validation before execution.

4.10.2 Stimulus/Response Sequences

1. Stimulus: A Process Orchestrator initiates the integration of a contract workflow with an external system.
   Response: The system connects with the configured AI/ERP platform, translates contract milestones into actionable triggers, and starts the synchronized execution.
2. Stimulus: A Validator triggers a compliance check before contract deployment.
   Response: The system automatically reviews contract content, structure, and metadata for consistency with legal rules and internal policies, then returns a validation report.

4.10.3 Functional Requirements

[DCS-FR-UC-10-1] Automate Contract Workflow Processes
FR-UC-10-1 is linked to the following functional requirements listed in Section 3.2:

- FR-CWE-28 – Automated Contract Interaction via API
- FR-CSA-25 – Contract Processing API

[DCS-FR-UC-10-2] Validate Contract Integrity & Compliance
FR-UC-10-2 is linked to the following functional requirements listed in Section 3.2:

- FR-PACM-03 – Automated Regulatory and Policy Compliance Checks

4.11 UC-11 API & System Integrations

4.11.1 Description and Priority

Priority: High
This feature handles external system communication. The associated use case of this feature is listed as follows:

- UC-11-01 – Manage API-Based Contract Workflows: Ensures seamless contract automation via API integrations.

4.11.2 Stimulus/Response Sequences

1. Stimulus: An Integration Manager configures or invokes an API to trigger a contract-related event.
   Response: The system authenticates the request, initiates the associated workflow (e.g., contract creation, signing, validation), and logs the interaction for traceability.

4.11.3 Functional Requirements

[DCS-FR-UC-11-1] API & System Integration
FR-UC-11-1 is linked to the following functional requirements listed in Section 3.2:

- FR-CSA-25 – Contract Processing API
- FR-CWE-28 – Automated Contract Interaction via API
- FR-SM-25 – Automated Signature Processing API

4.12 UC-12 System-Based Contract Management

4.12.1 Description and Priority

Priority: High
This feature covers automated contract execution through system roles. The associated use cases of this feature are listed as follows:

- UC-12-01 – Create Contract via API: Enables automated contract creation through system integration.
- UC-12-02 – Review Contract via API: Supports contract validation through system-based checks.
- UC-12-03 – Approve Contract via API: Handles automated contract approvals.
- UC-12-04 – Manage Contracts via API: Integrates contract management into AI/ERP systems.
- UC-12-05 – Sign Contract via API: Supports automated and AI-driven contract signing.

4.12.2 Stimulus/Response Sequences

1. Stimulus: A System Contract Creator service triggers contract generation through API.
   Response: The system retrieves the relevant template, populates it with data from integrated systems, and saves the contract for processing.
2. Stimulus: A System Contract Reviewer initiates a review workflow via API.
   Response: The system checks the contract content against predefined rules, flags inconsistencies, and reports results for automated correction or routing.
3. Stimulus: A System Contract Approver service submits an approval request.
   Response: The system validates the request origin, marks the contract as approved, and logs the decision with system metadata.
4. Stimulus: A System Contract Manager component requests access to manage a contract's lifecycle.
   Response: The system exposes APIs for querying, updating, and tracking contract metadata, supporting AI-driven contract lifecycle automation.
5. Stimulus: A System Contract Signer initiates a signature operation via API.
   Response: The system generates a digital signature, binds it to the contract using verifiable credentials, and updates the contract status.

4.12.3 Functional Requirements

[DCS-FR-UC-12-1] Create Contract via API
FR-UC-12-1 is linked to the following functional requirements listed in Section 3.2:

- FR-CWE-13 – Contract Creation
- FR-CWE-28 – Automated Contract Interaction via API

[DCS-FR-UC-12-2] Review Contract via API
FR-UC-12-2 is linked to the following functional requirements listed in Section 3.2:

- FR-CWE-17 – Contract Review
- FR-CWE-15 – Contract Review and Approval

[DCS-FR-UC-12-3] Approve Contract via API
FR-UC-12-3 is linked to the following functional requirements listed in Section 3.2:

- FR-CWE-25 – Contract Review and Approval Interface

[DCS-FR-UC-12-4] Manage Contracts via API
FR-UC-12-4 is linked to the following functional requirements listed in Section 3.2:

- FR-CWE-24 – Contract Management Dashboard
- FR-CWE-31 – Contract Performance Tracking

[DCS-FR-UC-12-5] Sign Contracts via API
FR-UC-12-5 is linked to the following functional requirements listed in Section 3.2:
- FR-SM-25 – Automated Signature Processing API

4.13 UC-13 External System Contract Execution

4.13.1 Description and Priority

Priority: Low
This feature ensures contract enforcement in target systems. The associated use case is:
- UC-13-01 – Deploy Contract to Target System: Enables contract execution in ERP or similar systems.

4.13.2 Stimulus/Response Sequences

1. Stimulus: The DCS-TRG-SYS (Target System) requests or receives a signed and validated contract deployment payload.
   Response: The system transfers the contract content, verifies delivery, and confirms contract activation/execution in the target environment.

4.13.3 Functional Requirements

[DCS-FR-UC-13-1] External Contract Deployment
FR-UC-13-1 is linked to the following functional requirements listed in Section 3.2:
- FR-SM-10 – Proof of Contract Execution
- FR-SM-12 – Contract Deployment Trigger
- FR-CWE-06 – Event-Driven Contract Execution

4.14 UC-14 Identity and PoA Credential Acquisition

4.14.1 Description and Priority

Priority: High
This feature ensures the retrieval and verification of identity credentials and PoA credentials to authorize contract signing and execution. The associated use case is:
- UC-14-01 – Retrieve Identity and PoA credentials: Ensures the acquisition of verified identity and PoA credentials required before any contract can be signed or executed.

4.14.2 Stimulus/Response Sequences

1. Stimulus: The DCS-CTR-SGN (Contract Signatory) or DCS-SYS-SGN (System Signatory) initiates a signing process for a contract.
   Response: The system verifies whether the required identity and PoA credentials are already present. If not, it queries trusted external sources for credential data, validates them, associates them with the user/session, and then authorizes the signing operation.

4.14.3 Functional Requirements

[DCS-FR-UC-14-1] Identity & PoA Credential Verification
FR-UC-14-1 is linked to the following functional requirements listed in Section 3.2:
- FR-SM-03 – Signing Identity & PoA Authorization Credentials
- FR-SM-04 – Counterparty Authorization & PoA Credential Chain Verification
- FR-SM-05 – Integration with Signing Identity and PoA Verifiable Credentials
- FR-SM-26 – Signature Compliance Viewer

4.15 UC-15 Access Rights Revocation

4.15.1 Description and Priority

Priority: Normal

This feature ensures the revocation of signatures or credentials to invalidate access rights when organizational policies, credential invalidation, or regulatory requirements demand it. The associated use case is:

- UC-15-01 – Revoke Access Rights and Signatures: Ensures that contracts signed with invalid or revoked credentials are marked as non-compliant and access rights are removed.

4.15.2 Stimulus/Response Sequences

1. Stimulus: The Auditor or Compliance Officer identifies that a signer's credentials have been revoked in the XFSC Revocation List via organizational policy.
   Response: The system checks the revocation list, verifies credential validity, and if revoked:
   - Updates the contract record to "revoked" state.
   - Logs the revocation event.
   - **Invalidates associated rights until re-signing occurs.**

4.15.3 Functional Requirements

UC-15-1 is linked to the following functional requirements listed in Section 3.2:
- FR-SM-20 – Signature Revocation
- FR-SM-26 – Signature Compliance Viewer

5 Other Requirements

C2PA Content & Life Cycle Credentials for PDF Contracts

C2PA (Coalition for Content Provenance and Authenticity)[1] is an open standard for tamper-evident provenance metadata called "Content Credentials." It records who created a file, which tools were used, and what changed. Adobe, Microsoft, and others are standardising the format and integrating it into media tools, cameras, platforms, and PDF workflows.

This is important because provenance can be verified across systems for audit and compliance. In digital contract signing services, we add a C2PA manifest to the signed PDF (or host a remote manifest). It binds the contract ID and file hash, tracks lifecycle states (active, amended, terminated), and can link to a verifiable credential and status list. The legal e-signature (or PCM, OCM signature) stays as is; C2PA adds verifiable context and status. C2PA document life-cycle assertions can be added incrementally with every life-cycle-event.

[DCS-OR-C2PA-001] Use of C2PA for Contract Provenance
The system MUST use the C2PA standard to record origin and edits of contract PDFs. C2PA is an open standard that adds tamper-evident Content Credentials. It helps detect manipulation and supports audit and compliance.
- Measurement: Share of contract PDFs with a valid C2PA manifest (target 100%).
- Verification Method: Tool-based C2PA validation; documentation review.

[DCS-OR-C2PA-002] PDF Embedding and Incremental Updates
The system MUST embed a C2PA manifest in each signed contract PDF or link a remote manifest. It MUST use PDF incremental updates so existing legal signatures remain valid.

[1] www.c2pa.org

- Measurement: PDFs that pass both PDF signature checks and C2PA verification after an update (target 100%).
- Verification Method: Sign→append→verify test; PDF/C2PA tool checks.

[DCS-OR-C2PA-003] Contract Lifecycle Assertions
The system MUST model lifecycle states as C2PA assertions: draft, active, amended, suspended, terminated, expired, replaced. Each assertion MUST include: contract_id, file_hash, status, reason, effective_at, authority, vc_id, prev_manifest_hash (if any).

- Measurement: Coverage of all states and fields in test assets (target 100%).
- Verification Method: Schema validation; unit tests on sample manifests.

[DCS-OR-C2PA-004] Verifiable Credential Binding
The system MUST issue a W3C VC for contract status. The VC MUST bind to contract_id and file_hash and repeat status, reason, effective_at. The C2PA manifest MUST carry a link to the VC (vc_id) or an embedded copy.

- Measurement: Lifecycle events that have a matching VC (target 100%).
- Verification Method: VC signature verification; cross-check VC ↔ C2PA fields.

[DCS-OR-C2PA-005] Status Publication and Revocation
The system MUST publish current contract status in a verifiable list (e.g., Status List 2021/2023 or equivalent). It MUST support real-time suspension and termination.

- Measurement: Time from status approval to list update (target ≤ 5 minutes).
- Verification Method: Integration tests; timestamp comparison.

[DCS-OR-C2PA-006] Verifier Behavior and UI
The verifier MUST check PDF signatures, C2PA manifests, the VC signature, and the status list. It MUST show a clear banner: Active, Suspended, Terminated, Replaced, Expired, or Draft.

- Measurement: Correct banner shown for all test cases (target 100%).
- Verification Method: Automated verifier tests; manual UX check.

[DCS-OR-C2PA-007] Trust Anchors and Delegation
Issuer keys MUST be anchored to the organization (e.g., org DID with LPID/eIDAS data or Qualified eSeal/QES). Delegation for status changes MUST be proven by a PoA credential. Keys MUST support rotation and revocation.

- Measurement: Existence of key and PoA policy; successful rotation drills (2/year).
- Verification Method: Policy review; drill reports; audit of key IDs in VC/C2PA.

[DCS-OR-C2PA-008] Resilience to Metadata Stripping
A remote C2PA manifest MUST exist for every contract. The verifier MUST fetch it if the embedded manifest is missing or stripped.

- Measurement: Successful verification after stripping in test copies (target 100%).
- Verification Method: Strip-then-verify tests; remote fetch logs.

[DCS-OR-C2PA-009] Audit and Trusted Time
Each lifecycle assertion and VC issuance SHOULD be time-stamped (RFC 3161 or equivalent). An append-only audit log SHOULD record who changed status and why.

- Measurement: Time-stamp and audit entries for all events (target 100%).
- Verification Method: Log review; TSA signature checks.

[DCS-OR-C2PA-010] Backward Compatibility with Legal Signatures
Adding or updating C2PA manifests MUST NOT break existing PDF legal signatures.

- Measurement: PDFs failing signature validation after C2PA updates (target 0).
- Verification Method: PDF signature validation before and after updates.

6 Verification

This verification plan qualifies the DCS by demonstrating end-to-end acceptance scenarios that parallel the Use Cases in Section 4. Rather than verifying every requirement individually, each scenario exercises the functional path with realistic inputs and records objective evidence (logs, signatures, artifacts, reports) that the system fulfills its specified behavior. Non-functional observations MAY be recorded during the runs but are not formal pass/fail criteria unless explicitly noted. Table 6 captures the acceptance criteria for the use cases defined in Section 4, whereas Table 7 depicts the acceptance criteria for sub use cases in Section 4. In addition to the acceptance criteria in the tables, the following requirements apply as acceptance criteria to all scenarios in Table 6 and Table 7:

1. All acceptance scenarios SHOULD be specified and executed as executables in Behaviour-driven development (BDD) framework using Gherkin and Behave step definitions, run via the XFSC bdd-executor. The acceptance presentation SHOULD demonstrate these tests live, and the acceptor SHOULD be able to run the same pack locally. A Rancher Desktop Setup would be preferred for easy adoption during verification, instead of Docker Compose.
2. Implementations SHOULD conform to the XFSC OCM W Default Toolstack (service mesh, databases, messaging, CI/CD, Helm, Kubernetes, API design, etc.). The preference is to use Golang and provide adapters to Cassandra (for data persistence) or NATS (for messaging) for external sources, and eventing/messaging SHOULD be demonstrable with NATS.
3. All deployments SHOULD be scripted via Helm and installable in Kubernetes; charts SHOULD support configuration profiles (DEV, Acceptance, Prod) and use secretRefs (no clear-text secrets). Charts SHOULD be suitable for Argo CD and published to a Helm repository.

Table 6 – Acceptance criteria defined for Section 4 use cases

Use Case	Description	Acceptance Criteria
UC-01 User Authentication & Authorization	Authenticate; access a role-protected page/API; attempt an unauthorized action	Authorized access succeeds; unauthorized action denied; audit entry shows actor, role, decision
UC-02 Template Management	Create template; submit for review; approve; search & open	Template stored with version/provenance; status transitions logged; search returns approved item
UC-03 Contract Creation & Approval	Instantiate from template; edit metadata; route for approval; lock content	Contract ID issued; “Approved” status; immutable hash/provenance recorded; audit trail complete
UC-04 Contract Signing	Open secure viewer; present identity/PoA; execute signature; verify result	Valid signature attached; signer & PoA bound; verification OK; timestamp and evidence stored
UC-05 Contract Deployment	Trigger deployment; observe outbound call; confirm receipt	Target acknowledges deployment; DCS status “Deployed;” correlation ID and receipt archived
UC-06 Lifecycle Management	View KPIs/alerts; initiate renew or terminate flow; confirm state change	KPIs visible; new term or terminated state recorded; notifications and logs captured
UC-07 Storage & Archive	Store in archive; search; retrieve artifact	Entry stored as PDF/A-3 (or configured format); search finds it; retrieval returns intact file; audit event written
UC-08 Audit & Compliance	Request activity report; run policy audit; export results	Report lists actors/timestamps/actions; violations (if any) flagged with reasons; exports available
UC-09 Administration	Create/modify roles; assign to user; open monitoring/logs	Role takes effect immediately; changes and admin actions logged; health metrics visible

UC-10 Automation/Orchestration	Invoke HTTP node to start process; receive webhooks/callbacks; complete flow	End-to-end completes with 2xx responses; callback received; trace shows ordered events
UC-11 API & Integrations	Call API to create, sign, and validate; inspect responses	Auth succeeds; APIs return HTTP 2xx; validation result returned; request/response logs with correlation IDs
UC-12 System-Based Contract Mgmt	Create → review → approve → sign → archive via API	Each step updates status; signature evidence stored; archive entry created; full audit chain present
UC-13 External Execution	Submit execution payload; verify activation in target	Target confirms activation; DCS stores proof (receipt/hash/tx-id); status reflects "Executed"
UC-14 Identity & PoA	Fetch credentials; validate; bind to session	Credentials verified (valid, unrevoked); authorization check passes or blocks with reason; event logged
UC-15 Access Rights Revocation	Revoke role/credential; attempt prior action (access/sign)	Access/signing denied; affected items flagged (if configured); revocation visible in logs/status lists
UC-EUDI	EUDI PID verification	Verify Person Identification Data (PID) presented from an EUDI-compliant wallet (e.g., Lissi EUDI Wallet Connector or EU EUDI Wallet Reference Implementation).

Table 7 – Acceptance Criteria defined for Section 4 sub use cases

Sub Use Case	**Description**	**Acceptance Criteria**
UC-02-01 – Create Contract Template	Create reusable template by Template Manager/Approver.	Show a template is created with required metadata/provenance; repository returns a template ID/version; entry appears in search; action is audit-logged.
UC-02-02 – Search & Retrieve Templates	Find and access existing templates.	Execute search with filters; results respect RBAC; opening a template displays current version & provenance; access events logged.
UC-02-03 – Generate Contract from Template	Populate template with context data to create a contract.	Given inputs, system produces a draft with linked template ID; both machine- and human-readable versions render; creation logged.
UC-02-04 – Update Contract Template	Edit existing template with versioning.	Update creates a new immutable version; previous remains readable; diff and author shown; change logged.
UC-02-05 – Deprecate Contract Template	Mark a template deprecated.	Deprecation prevents new contract generation; banner shows status; event logged with timestamp and user.
UC-02-06 – Add Template Provenance	Capture origin, contributors, identifiers.	Add provenance fields; system validates/records; provenance visible in UI/API and included in exports.
UC-02-07 – Verify Template & Provenance	Validate correctness, semantics (JSON-LD/SHACL) and authenticity.	Run verification; success report lists schema checks and signature/VC validation; failures block generation.

UC-02-08 – Create & Maintain Semantic Schemas	Manage schemas used by templates.	Create/update schema; link to templates; validation enforces conformity; schema versioning and rollback demonstrated.
UC-02-09 – Template Management Dashboard	Track status, approvals, usage.	Dashboard shows per-template lifecycle, usage metrics, last changes; supports filtering and export; access controlled.
UC-03-01 – Create Contract	Generate contract from predefined templates.	Create a draft; receive contract ID; both views render; creation logged and traceable to template version.
UC-03-02 – Negotiate Contract Terms	Collaboratively adjust clauses before finalization.	Add comments/edits; see tracked changes and negotiation log; version history preserved.
UC-03-03 – Adjust Contract Terms	Granular clause edits without regenerating.	Edit a clause; integrity checks pass; only targeted sections change; audit trail updated.
UC-03-04 – Approve Contract	Route to required approvers before signing.	Route shows pending/approved states; all required approvals recorded with timestamps; content locked on completion.
UC-03-05 – Review MR/HR Correctness & Versions	Validate machine- and human-readable consistency.	Open both renderings; system highlights inconsistencies (none expected after fix); export both with same version/tag.
UC-03-06 – Manage Contract Signing Process	Coordinate structured signing steps.	Configure signers/sequence; system schedules, reminds, and tracks status; changes logged.
UC-03-07 – Contract Dashboard & Search	Track progress and search contracts.	Dashboard shows lifecycle states; full-text and metadata search returns expected contracts respecting RBAC.
UC-04-01 – Review & Sign Contract Electronically	Secure viewer; legally binding e-signature incl. identity/PoA.	Signer authenticates; signs in viewer; system produces signed artifact (e.g., PAdES/JAdES) and updates status; event logged.
UC-04-02 – Verify Counterparty Authorization	Check legal authority to sign (identity/PoA).	Present counterparty VC/PoA; system validates chains/status lists; unauthorized cases block signing with error.
UC-04-03 – Verify Counterparty Signature	Validate authenticity/integrity of signature.	Run crypto validation and policy checks; report shows certificate/VC status and document hash match.
UC-05-01 – Deploy Signed Contract to Target System	Make signed contract available for execution.	Push deployment payload to target; receive ack/callback; target reads content; DCS logs proof of delivery.
UC-06-01 – Monitor Contract Performance	SLA/compliance monitoring & tracking.	Dashboard displays KPIs/milestones; alerts fire for violations; history shows fulfilled terms.
UC-06-02 – Renewal or Termination	Manage renewal/termination incl. VC revocation.	Trigger renewal/termination; system updates state, issues/updates revocation where applicable; notifications/logs produced.
UC-07-01 – Store Contract in Secure Archive	Tamper-proof, long-term storage of signed contracts.	Archive a signed contract; system seals, timestamps, encrypts, and returns archive ID; retrieval confirms integrity.

UC-07-02 – Manage Contract Permissions & Access	Control RBAC for stored contracts.	Change access policy; unauthorized access is denied; permitted roles can retrieve; changes and access attempts logged.
UC-07-03 – Storage & Security Dashboard	Track archive status, integrity, access logs.	Dashboard shows coverage/integrity checks, alerts, and recent access; export of logs available.
UC-08-01 – Report Contract Activity Logs & Timestamps	Produce auditable reports.	Generate report with creation/approval/signature events; includes timestamps/actors; export to CSV/PDF.
UC-08-02 – Audit Contract Compliance	Check against legal/organizational policies.	Run compliance scan; issues listed with rule references; pass/fail summary produced and archived.
UC-09-01 – System Configuration & User Management	Administer roles, security, settings.	Admin changes a role/setting; effect visible immediately; all admin actions logged with actor/time.
UC-09-02 – System Monitoring & Logging	Operational monitoring and security logs.	Show health metrics & searchable logs; filters by severity/time; export supports incident review.
UC-10-01 – Automate Contract Workflow Processes	Integrate workflows with AI/ERP (orchestration).	Orchestrator triggers external action from a contract milestone; target system receives and executes; trace visible end-to-end.
UC-10-02 – Validate Contract Integrity & Compliance	Pre-execution automated validation.	Run validation; violations block deployment; detailed report stored with contract.
UC-11-01 – Manage API-Based Contract Workflows	Ensure automation via API integrations.	Invoke API to create/sign/validate; request is authenticated; workflow executes; interaction logged for traceability.
UC-12-01 – Create Contract via API	Automated contract creation through system integration.	POST creates contract; returns ID & status; data pulled from integrated system; audit log records requester/system.
UC-12-02 – Review Contract via API	System-driven validation checks.	API call runs rule checks; response lists issues; failing contracts cannot proceed to approval.
UC-12-03 – Approve Contract via API	Automated approvals.	API marks contract approved; requires authN/authZ; status changes and approver identity logged.
UC-12-04 – Manage Contracts via API	Query/update/track lifecycle.	Use APIs to list/update metadata and read history; RBAC enforced; changes versioned and logged.
UC-12-05 – Sign Contract via API	Automated/AI-driven signing.	Sign endpoint produces valid signature artifact; binds signer VC/PoA; status → “signed”; verification succeeds.
UC-13-01 – Deploy Contract to Target System	Execute in ERP targets.	Deliver deployment payload; target confirms activation/execution; DCS records proof-of-execution reference.
UC-14-01 – Retrieve Identity & PoA Credentials	Acquire verified identity/PoA before signing/execution.	Missing credentials trigger retrieval; chain and status verified; authorization granted only on success; events logged.

UC-15-01 – Revoke Access Rights & Signatures	Invalidate access when credentials/signatures are revoked.	Detect revocation via status list; mark contract state "revoked"; rights withdrawn; audit entry created; requires re-sign to restore.

7 Appendix

Appendix A: Glossary

Access Rights Revocation: Withdrawing rights when credentials or signatures are revoked; affected items are flagged and require re-signing to restore.

Audit Log Filtering: The ability to search and filter audit logs based on specific criteria, such as date range, user, or event type.

Automated Alerts: System-generated notifications for key contract events, such as renewals, expirations, or required actions.

Automated Breach Notification: A system-triggered alert to relevant parties when a contract-related compliance breach is detected.

Automated Compliance Checks: System-driven validation processes that compare contracts against legal, regulatory, or organizational requirements before execution or storage.

Automated Workflows: Automates contract generation, execution, and deployment to ensure legal consistency and efficiency.

Clause Library: A curated collection of pre-approved contract clauses that can be reused across different templates to ensure legal and operational consistency.

Compliance Dashboard: A user interface that provides real-time visibility into the compliance status of contracts, workflows, and related activities.

Counterparty Authorization: The verification process ensuring the other contracting party has proper authority to sign on behalf of their organization.

Counterparty Contract Signature Verification: The process of confirming the authenticity and integrity of a counterparty's digital signature.

Cross-Format Signature Binding: Ensuring that a signature is valid across multiple contract representations (e.g., machine-readable and human-readable versions).

Cryptographic Proof of Contract Execution: A verifiable record containing hashes, timestamps, and signer identity data that confirms a contract was executed as intended.

Delegated Signing: The process where an authorized individual signs a contract on behalf of another person or organization based on a verifiable power of attorney or delegation credential.

Event-Driven Contract Execution: Progressing execution steps when specified lifecycle events occur.

Immutable Storage: A storage system where data, once written, cannot be altered or deleted, ensuring integrity for legal and compliance purposes.

Linked Signatures: Digital signatures bound to both machine-readable and human-readable versions of a contract, ensuring content consistency across formats.

Machine-Readable Contract Template: A contract template encoded in a structured, machine-processable format such as JSON-LD for interoperability and automation.

Machine Signing: Supports automated signing for high-volume or routine transactions.

Multi-Contract Signing: Enables multi-party contract execution within a single integrated workflow.

Multi-Factor Signing Authentication: A signing process that requires more than one authentication factor (e.g., password + biometric) to verify the signer's identity.

Negotiation History: A record of all changes, comments, and proposals made during the contract negotiation process.

Non-Compliance Investigation: A process for identifying, analyzing, and reporting contract-related violations or missed obligations.

Organizational Seal: A digital seal that ensures the origin and integrity of electronic documents issued by a legal entity.

Parallel Signing: A signing mode where multiple parties can apply their signatures simultaneously rather than sequentially.

Regulatory Reporting Export: A feature that generates reports in standardized formats for submission to regulators or oversight bodies.

Remote QES Signing Service: An external service that enables the remote application of a qualified electronic signature, typically via a secure API.

Retention Schedule: A documented policy specifying how long contracts and related records must be retained before being archived or deleted.

Role Assertion: A verifiable-credential claim of a user's role (e.g., Template Manager/Approver) presented for authentication and authorization.

Role-Based Access Control for Audit Logs: A permissions model that restricts access to audit logs based on predefined user roles to protect sensitive compliance data.

Role-Based Signature Routing: The automated assignment of signing tasks to the correct parties based on their predefined roles and authority.

Secure Contract Viewer: A protected interface for viewing contract content in a secure, tamper-proof environment before signing, preventing unauthorized changes.

Secure Retrieval: A controlled process for accessing stored contracts that enforces authentication, authorization, and audit logging.

Semantic Hub: A centralized service or repository that provides standardized vocabularies, taxonomies, and ontologies to ensure semantic consistency across contract templates and workflows.

Signature Management: The process of linking contract signatures to verifiable digital identities to maintain legal validity and trust.

Signature Process: The structured set of steps for applying electronic signatures to a contract, ensuring role-based compliance and validation.

Signature Revocation: The act of invalidating a previously applied electronic signature, typically in cases of compromise, error, or contract withdrawal.

Signature Verification Service: A service that checks the validity, authenticity, and integrity of an electronic signature against predefined legal and technical criteria.

Signed Contract Archive: A secure repository for finalized contracts, maintained in both machine-readable and human-readable formats to ensure long-term accessibility and integrity.

Signer Authorization: The verification process ensuring a person has the authority to sign a contract on their own behalf or for an organization.

Status List: A revocation/status registry used to check whether credentials or signatures are valid.

Structural Dependency Mapping: The process of identifying and linking related sections, clauses, or data points within or across contract templates to maintain logical and legal coherence.

Tamper-Proof Audit Trail: An immutable record of all contract-related activities throughout the lifecycle, ensuring that no entries can be altered without detection.

Template Identifier Generator: A generator that assigns globally UUIDs or DIDs to templates, ensuring they are universally identifiable and traceable across contract workflows.

Versioned Archive: An archive that maintains multiple versions of a contract or document, preserving a full historical record of changes.

Workflow State Transition: The change of a contract's status within the workflow lifecycle, triggered by specific events or actions.

Appendix B: To Be Determined List

TBD-A: Use of Qualified Electronic Seals in Digital Contracts

Status: Open
Definition: The legal validity and contractual effect of applying a qualified electronic seal (QSeal) generated with a qualified seal creation device (QSCD) to contractual documents within the [System/Project] workflow has not yet been determined.
Resolution Criteria:

1. Receipt of written legal advice confirming whether QSeals (as distinct from qualified electronic signatures) are sufficient or complementary for contract formation and/or evidentiary purposes under applicable law, and any constraints on their use.
2. Confirmation of the relevant technical and certification properties of the selected D-Trust electronic seal solution (e.g., qualification status, device protection/QSCD, certificate profile, validation/long-term validation capabilities) and their alignment with the legal advice.

Dependencies: External legal counsel; D-Trust product/technical documentation (see vendor information: d-trust electronic seals)

TBD-B: Use of XFSC PCM as Personal Identity Wallet

Status: Open
Definition: The selection and use of the XFSC Personal Credential Manager (PCM) as the personal identity wallet for natural-person users in DCS (covering authentication, credential presentation, Power-of-Attorney (PoA) chain verification, and signing flows) is not yet finalized. While the SRS foresees integration with identity wallets and explicitly references XFSC's PCM via the FACIS orchestration layer and EUDI-conformant infrastructures, the actual availability, maturity, and compatibility of PCM for the required DCS use cases remain to be confirmed.

Resolution Criteria:

1. Status & availability report from ECO Verband and XFSC wallet experts: A written statement on current status, roadmap, and deployment availability (demo/test/prod) of XFSC PCM (incl. Cloud PCM) and support/hosting model. Note: the SRS anticipates orchestration that "can be hosted by ECO."

Appendix C: JSON-LD Contract Policy Example

This chapter provides a JSON-LD based example of a machine-readable contract policy, illustrating how contractual conditions and obligations can be expressed in a standardized format for validation and enforcement within the Digital Contracting Service. This policy is expressed in the W3C Open Digital Rights Language (ODRL).


```
[ { "@id": ":b0", "@type": [ "http://www.w3.org/ns/odrl/2/Permission" ],
"http://www.w3.org/ns/odrl/2/action": [ { "@id":
"http://www.w3.org/ns/odrl/2/use" } ], "http://www.w3.org/ns/odrl/2/assignee":
[ { "@id": "https://w3id.org/drk/resource/RegionalNewspaper" }, { "@id":
"https://w3id.org/drk/resource/CultureResearchInstitute" } ],
"http://www.w3.org/ns/odrl/2/assigner": [ { "@id":
"https://w3id.org/drk/resource/DE_Staatstheater_Augsburg" } ],
"http://www.w3.org/ns/odrl/2/constraint": [ { "@id": ":b1" }, { "@id": ":b2" }
], "http://www.w3.org/ns/odrl/2/target": [ { "@id":
"https://w3id.org/drk/resource/AugsburgStaatstheaterShowtimesAPI" } ] }, {
"@id": ":b1", "@type": [ "http://www.w3.org/ns/odrl/2/Constraint" ],
"http://www.w3.org/ns/odrl/2/leftOperand": [ { "@id":
"http://www.w3.org/ns/odrl/2/spatial" } ],
"http://www.w3.org/ns/odrl/2/operator": [ { "@id":
"http://www.w3.org/ns/odrl/2/eq" } ],
"http://www.w3.org/ns/odrl/2/rightOperand": [ { "@value": "DE" } ] }, { "@id":
":b2", "@type": [ "http://www.w3.org/ns/odrl/2/Constraint" ],
"http://www.w3.org/ns/odrl/2/leftOperand": [ { "@id":
"http://www.w3.org/ns/odrl/2/dateTime" } ],
"http://www.w3.org/ns/odrl/2/operator": [ { "@id":
"http://www.w3.org/ns/odrl/2/lteq" } ],
"http://www.w3.org/ns/odrl/2/rightOperand": [ { "@type":
"http://www.w3.org/2001/XMLSchema#dateTime", "@value": "2025-05-10T23:59:59" }
] }, { "@id":
"https://w3id.org/drk/resource/AugsburgStaatstheaterShowtimesAPI", "@type": [
```

```
"https://w3id.org/drk/ontology/ShowTimesAPI",
"http://www.w3.org/ns/odrl/2/Asset" ], "http://purl.org/dc/terms/title": [ {
"@language": "de", "@value": "AugsburgStaatstheaterShowtimesAPI" } ],
"http://www.w3.org/ns/odrl/2/hasPolicy": [ { "@id":
"https://w3id.org/drk/ontology/TempoSpatialAccess" } ],
"http://www.w3.org/ns/odrl/2/uid": [ { "@value":
"AugsburgStaatstheaterShowtimesAPI" } ] }, { "@id":
"https://w3id.org/drk/resource/CulturalPlatformBavaria", "@type": [
"http://www.w3.org/ns/odrl/2/Party" ], "http://www.w3.org/2000/01/rdf-
schema#label": [ { "@language": "en", "@value": "Cultural Platform Bavaria" }
], "http://www.w3.org/2004/02/skos/core#definition": [ { "@language": "en",
"@value": "Represents a Cultural Platform subscriber" } ],
"http://www.w3.org/ns/odrl/2/uid": [ { "@value": "CulturalPlatformBavaria" } ]
}, { "@id": "https://w3id.org/drk/resource/CultureResearchInstitute", "@type":
[ "http://www.w3.org/ns/odrl/2/Party" ], "http://www.w3.org/2000/01/rdf-
schema#label": [ { "@language": "en", "@value": "Culture Research Institute" }
], "http://www.w3.org/2004/02/skos/core#definition": [ { "@language": "en",
"@value": "Represents a culture research institute subscriber" } ],
"http://www.w3.org/ns/odrl/2/uid": [ { "@value": "cultureresearchinstitute" }
] }, { "@id": "https://w3id.org/drk/resource/DE_Staatstheater_Augsburg",
"@type": [ "http://www.w3.org/ns/odrl/2/Party",
"http://xmlns.com/foaf/0.1/Organization" ], "http://www.w3.org/2000/01/rdf-
schema#label": [ { "@language": "en", "@value": "DE_Staatstheater_Augsburg" }
], "http://www.w3.org/ns/odrl/2/uid": [ { "@value":
"DE_Staatstheater_Augsburg" } ] }, { "@id":
"https://w3id.org/drk/resource/RegionalNewspaper", "@type": [
"http://www.w3.org/ns/odrl/2/Party" ], "http://www.w3.org/2000/01/rdf-
schema#label": [ { "@language": "en", "@value": "Regional Newspaper" } ],
"http://www.w3.org/2004/02/skos/core#definition": [ { "@language": "en",
"@value": "Represents a regional newspaper subscriber" } ],
"http://www.w3.org/ns/odrl/2/uid": [ { "@value": "regionalnewspaper" } ] }, {
"@id": "https://w3id.org/drk/resource/TempoSpatialAccess", "@type": [
"http://www.w3.org/ns/odrl/2/Policy" ],
"http://www.w3.org/ns/odrl/2/permission": [ { "@id": ":b0" } ],
"http://www.w3.org/ns/odrl/2/uid": [ { "@value": "TempoSpatialAccess" } ] } ]
```


Appendix D: OCM W-Stack Deployment

The OCM W-Stack Deployment contains several modules for the OID4VC/VP functionality described in the xfsc-documentation[2]. The deployment scripts are defined in deployment section[3] of xfsc. The stack itself splits the protocol in various parts[4]: Retrieval, Issuance, Verification and Well Known. Essential crypto parts are defined in the Crypto Provider Service (formerly TSA Signer Service). This service can be used for operations like LDP VC/SD JWT Creation or Verification, creating DID Documents etc. over the REST API[5].

[2] https://github.com/eclipse-xfsc/docs/tree/main/ocm-w-stack
[3] https://github.com/eclipse-xfsc/deployment/tree/main/OCM%20W-Stack
[4] https://github.com/eclipse-xfsc/crypto-provider-service
[5] https://github.com/eclipse-xfsc/crypto-provider-service/blob/main/design/design.go

Appendix E: Status Lists

The status list service[6] implements multiple formats of revocation lists. This service allows it to create a link plus database entry, which can be used for embedding within credentials or other purposes. The service can be deployed standalone or together with the Crypto Provider Service. The list entries can be requested over nats via a small snippet (golang, but works in other languages as well)[7]

Appendix F: Federated Catalogue

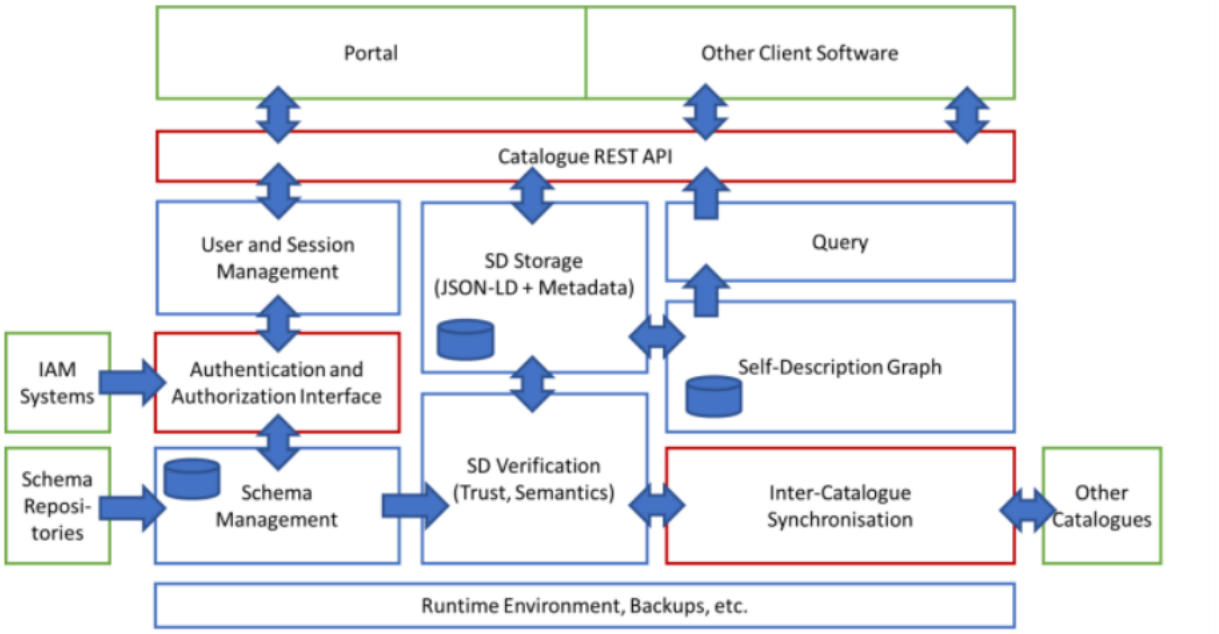


Fig.10 – High-level architecture of the XFSC Catalogue

A more detailed overview of the XFSC Federated Catalogue can be found in the Architecture Document: https://gaia-x.gitlab.io/data-infrastructure-federation-services/cat/architecture-document/architecture/catalogue-architecture.html

[6] https://github.com/eclipse-xfsc/statuslist-service
[7] https://github.com/eclipse-xfsc/statuslist-service/blob/main/example/main.go

Appendix G: DCS Component Diagram

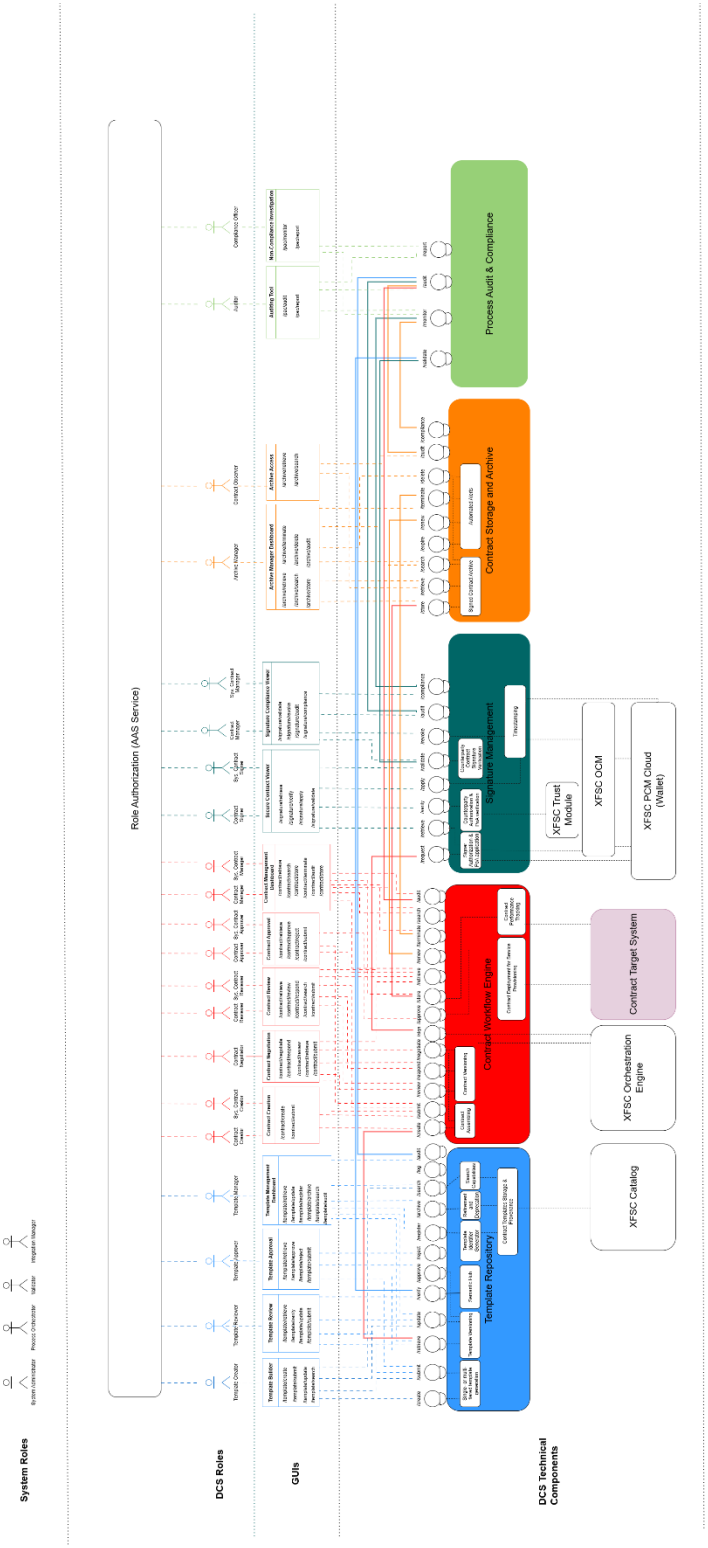


Fig.11 – DCS component diagram