

Technical Development Requirements

for

**Federation Architecture for
Composed Infrastructure
Services (FACIS)
Convoy Demonstrator (CD)**

FACIS.CD

Table of Contents

Open-Source Software	1
Runtime Environments.....	2
Programming Languages	2
Documentation.....	2
QA Requirements	5
DevOps Tools.....	6
Architecture Decision Records (ADR)	6

Open-Source Software

- **Mandatory use of XFSC ORCE¹ for**
 - Service orchestration
 - All CD components as ORCE flows and custom nodes, in separate ORCE instances as defined in SRS Section 8.1 (NFR-DEL-002)
 - Low-code integration
- **Deployment on Kubernetes**
 - Deployment into the Kubernetes cloud environment provided by FACIS (IONOS and T-Systems Open Sovereign Cloud).
- **Mandatory use of Kubernetes and Helm for**
 - Automated deployment
 - Update and uninstall processes
 - Resource and ingress management
- **Helm charts**
 - Deliver Helm charts with each release
- **Mandatory integration with XFSC OCM W-Stack and PCM Cloud for**
 - Credential issuance, presentation verification and status list via the OCM W-Stack, consumed as provided (SRS FR-GW-001, FR-GW-003, FR-GW-005, FR-GW-013)
 - Credential holders in PCM Cloud, one holder per vehicle (SRS FR-GW-014)
- **Mandatory use of the FACIS UI Builder package for**
 - Dashboard and console (SRS Section 2.1, NFR-DEL-002)
- **Repository rules**
 - All code, Node.js logic, UI schemas, Helm charts and scripts must reside in the designated FACIS repository, that will be communicated after the award of the tender
 - CI runs on GitHub Actions with security checks
 - No parallel private repositories allowed
 - Daily commits are mandatory. Meaning, the individual code must be stored in the Client's software repository at the end of each day of development
 - All code and charts live in the client-designated repository.
- **Compliance with Apache License, Version 2.0:**
 - Text version: <https://www.apache.org/licenses/LICENSE-2.0.txt>
 - SPDX short identifier: [Apache-2.0](https://spdx.org/licenses/Apache-2.0)
 - OSI Approved License: <https://opensource.org/licenses/Apache-2.0>
- **Only FOSS code must be used and must be compatible with Apache License 2.0**
 - The project should consume third-party content under one of the approved licenses listed here: <https://www.eclipse.org/legal/licenses.php#approved>. The client must be informed in written form when exceptions are planned to be included.
 - The client must confirm any existing FOSS code that is planned to be included in the deliverables.

¹ <https://github.com/eclipse-xfsc/orchestration-engine>

- The client must confirm API and FOSS component versions after contractor's in-depth assessment
- No redundant functions in the XFSC repository.
- A project git repository will be made available, and all contractors are obliged to use this repository as master on a daily base. A parallel contractor repository is not allowed.
- All work must be done in compliance with Eclipse Contributor Agreement 3.1.0 or higher: <https://www.eclipse.org/legal/ECA.php>

Runtime Environments

- Kubernetes v1.29 or higher in the FACIS cloud environment.
- Helm for deployments
- ORCE runtime for all CD components (SRS Section 8.1)
- No Docker Compose permitted

Programming Languages

- Node.js (JavaScript/TypeScript) for the CD component logic inside ORCE; logic that needs unit tests as plain modules wrapped by thin ORCE custom nodes (SRS Section 8.1)
- HTML + JavaScript + JSON schema for ORCE custom nodes; dashboard and console with the FACIS UI Builder package (SRS Section 2.1)
- Bash for automation scripts under /scripts
- YAML for Helm templates and Kubernetes manifests
- No further service languages: all CD components run as ORCE instances (SRS NFR-DEL-002, Section 8.1); the MQTT broker product is named in the Integration Note (SRS Section 8.3)

Documentation

- All documentation must follow Eclipse Project Handbook guidelines. <https://www.eclipse.org/projects/handbook/>
- FAP Partner Onboarding must be used for reference
 - Specifications: [https://github.com/eclipse-xfsc/facis/tree/main/FAP/Partner%20Onboarding%20\(Reference%20FAP\)/specification](https://github.com/eclipse-xfsc/facis/tree/main/FAP/Partner%20Onboarding%20(Reference%20FAP)/specification)
 - Implementation: <https://github.com/eclipse-xfsc/facis-fap-partner-onboarding/tree/17ed22c49dcc8c8c8749e4f402c8fe8d9bbe267e>
- Deliverables:
 - Dashboard, console and configuration-profile parameter documentation
 - Deployment and teardown instructions
 - Integration Note (SRS NFR-DEL-002) and credential provisioning instructions via OCM W-Stack/PCM Cloud (SRS NFR-DEL-001)
 - Helm chart documentation
 - Demonstration script and demo runbook covering M1–M5 and R (SRS NFR-DEL-001, NFR-DEL-005)
 - Troubleshooting guide
- All documentation must be included in docs/ or README.md in the project repository.
- Project repository:

- High-level description
- Packaging and container
- Deployment and preconditions
- Changes and additions to source specifications
- Functional breakdown for orchestrated flows
- Tests and demonstration script (SRS Section 10)
- Licenses
- OSS dependencies and external OSS references

General Principles

- **GitHub uses a monorepo-style approach**, where multiple sub-projects are organized in folders within a single repository. Each repository has one-version history, so nested independent Git projects are not supported by default.
- **Single Source of Truth:** Each repository is the authoritative source for its respective development activity.
- **Unified Content:** All relevant concepts, specifications, and implementation materials must reside in the same repository.

Repository Creation & Contributor Onboarding

- **Repository Creation:** Eclipse XFSC Project Leads are responsible for creating the main project repositories and applying the correct naming conventions. Communication with Eclipse XFSC Project Leads via the Client.
- **Contributor Onboarding:** The implementation team must complete the Contributor Onboarding² to be onboarded to the XFSC Eclipse Project.

Documentation Guide

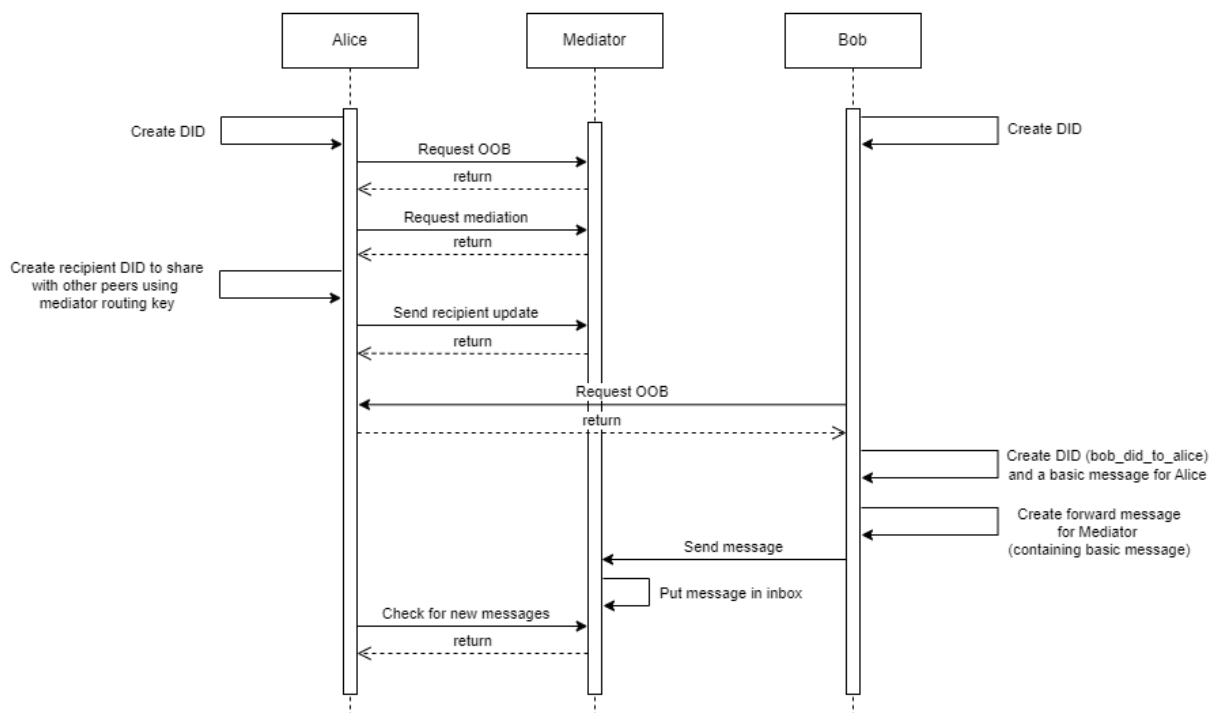
- Generator
 1. readthedoc
 2. mkdocs
- Product Documentation
 1. AsciiDoc
 2. RTS
- Architecture Modelling
 1. Archimate
 2. UML
 3. <https://www.drawio.com/>
- Interface description with versioned schemas, MQTT topics and example messages (SRS FR-IF-001); API definitions and examples for all REST endpoints; Node-RED node specs for the ORCE custom nodes.
- Versioned schemas for all messages (SRS FR-IF-001) and the JSON schema of the configuration profile (SRS FR-CFG-001).
- Template Examples are available from the Contractor.

² <https://eclipse-xfsc.github.io/landingpage/developer-guide/on-boarding>

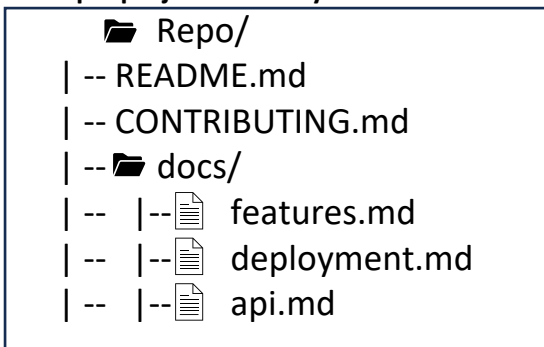
- The final repository will be communicated after the award of the tender.
- **README.md**
This markdown file should include the
 - Project Overview (brief description of the microservice),
 - Installation & setup (Dependencies, environment variables and configuration)
 - Technical Documentation Links
 - use case diagram (if applicable to understand the service functionality)

Example use case diagram (referenced from <https://gitlab.eclipse.org/eclipse/xfsc/common-services/didcomm-connector>)

The diagram shows the basic didcomm v2 flow between two peers (Alice and Bob):



Example project directory structure:



The docs/ directory should contain detailed documentation to help developers, contributors and users to understand the project effectively.

A template for the **contribution.md** "<https://github.com/auth0/open-source-template/blob/master/GENERAL-CONTRIBUTING.md>"

OSS Microservice Repository

File/ Folder	Purpose	Status (✓/✗)
.github/workflows	The eclipse dash license scanners and build workflows must be referenced in the repo from the standard org: https://github.com/eclipse-xfsc/dev-ops/tree/main/.github/workflows Example: https://github.com/eclipse-xfsc/oid4-vci-vp-library/tree/main/.github	
deployment/helm deployment/docker	Folders for Helm deployments and Docker image builds.	
README.md	Project overview, installation, setup, architecture, links to documentation folder, contributors responsible for contact info/e-mail address, concept and scope	
CONTRIBUTING.md	Guidelines for contributors, process release and issue reporting.	
LICENSE	License terms for the project	
CODE_OF_CONDUCT.md	Rules and behavior expected for the community	
docs/	Contains detailed documentation about the project/ Specifications or models. This doc section should be hosted on GitHub pages.	
docs/ci-cd.md	CI/ CD pipeline configuration, GitHub Actions and deployment.	
Docs/api-docs.md	API documentation (swagger, OpenAI)	

QA Requirements

ORCE Automation QA

- Deploy, redeploy and uninstall must run with zero manual intervention
- Logs must use structured JSON
- Error output must be machine-readable and exposed via ORCE context

Runtime Validation

- TLS 1.3 enforced for all endpoints
- Secrets stored in Kubernetes Secrets; no plaintext in logs
- Required Kubernetes objects created correctly in the client-provided namespace: ingress, service, deployment
- External dependencies (OCM W-Stack, PCM Cloud, trust endpoints) reachable

Test Suite

Must include the following tests (SRS Sections 10.1, 10.3 and 10.4):

- Unit tests in CI as required by SRS NFR-CFG-004, including the edge checks T-04, T-05 and T-10; each test that implements a step, row, vector or edge check of the SRS names it and asserts the result stated there
- Successful deployment
- Invalid parameters
- Idempotent redeployment
- Uninstall
- Integration tests of SRS Section 10.1, run once in the FACIS cloud environment (status-list block and release, FR-GW-012; agent stop of a rolling vehicle, FR-VEH-004)

Test Reports

- Deliver the CI test report with each tagged release (SRS NFR-CFG-004).

Target hosting environments for QA

- Reference cluster: FACIS cloud Kubernetes environment (SRS NFR-DEP-001, Section 10.1).
- Helm charts must pass lint and dry-run checks as part of QA.

DevOps Tools

- Git
- GitFlow
- GitHub Actions (CI/CD)
- Release artifacts: Helm repo, OCI images, CI test reports.

Architecture Decision Records (ADR)

ADR 001 – Deployment Engine

Decision: Helm is the mandatory deployment engine for all CD components.

Consequence: Repeatable Kubernetes-native deployments.

ADR 002 – Orchestration Layer

Decision: ORCE is the mandatory runtime of all CD components (SRS NFR-DEL-002, Section 8.1).

Consequence: One runtime for all components; logic in unit-testable modules behind thin custom nodes (SRS Section 8.2).

ADR 003 – Automation Stack

Decision: Node.js + Bash + YAML for backend logic and automation; UI built with the FACIS UI Builder package (SRS Section 2.1, NFR-DEL-002).

Consequence: Stable and consistent execution environment.

ADR 004 – Identity Integration

Decision: Vehicles and the traffic light are identified with DID-based verifiable credentials issued via the OCM W-Stack and held in PCM Cloud; convoy-layer services and console operators hold separate identities; no driver or personal credentials (SRS Section 3, FR-GW-015, NFR-SEC-003).

Consequence: Consistent identity model across all CD components, with vehicle, service and administrative identities kept separate.

ADR 005 – Security Baseline

Decision: TLS 1.3, RBAC, Kubernetes Secrets for service secrets (verifiable credentials remain in PCM Cloud, SRS FR-GW-001), log masking.

Consequence: Secure, auditable deployment operations.

ADR 006 – Logging and Observability

Decision: Structured JSON logs exposed via ORCE context outputs.

Consequence: Machine-readable logs for testing and monitoring.