

Technical Development Requirements

for

Federation Architecture for Composed Infrastructure Services (FACIS) Zero Trust Demonstrator

FACIS.ZTD

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
 - Builder Node orchestration
 - Deployment workflows
 - Low-code integration
- **Deployment on Kubernetes**
 - Deployment on Kubernetes clusters such as IONOS Cloud and T-Systems Open Sovereign Cloud (OSC).
- **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 Keycloak for**
 - Realm and client provisioning
 - Service credential management
- **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.
 - The client must confirm API and FOSS component versions after contractor's in-depth assessment
- No redundant functions in the XFSC repository.

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

- 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 on IONOS Cloud or T-Systems OSC.
- Helm for deployments
- ORCE runtime for Builder execution
- No Docker Compose permitted

Programming Languages

- Node.js for Builder module logic inside ORCE
- HTML + JavaScript + JSON schema for ORCE UI nodes
- Bash for automation scripts under /scripts
- YAML for Helm templates and Kubernetes manifests
- Golang for all other services

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:
 - Builder Node UI and parameter documentation
 - Deployment and teardown instructions
 - Integration notes for Keycloak
 - Helm chart documentation
 - BDD acceptance specifications
 - 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
 - BDD
 - 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/>
- API definitions and examples for all REST endpoints; Node-RED node specs for orchestration hooks.
- JSON-LD contexts and SHACL shapes for templates and data.
- Template Examples are available from the Contractor.
- The final repository will be communicated after the award of the tender.
- **README.md**

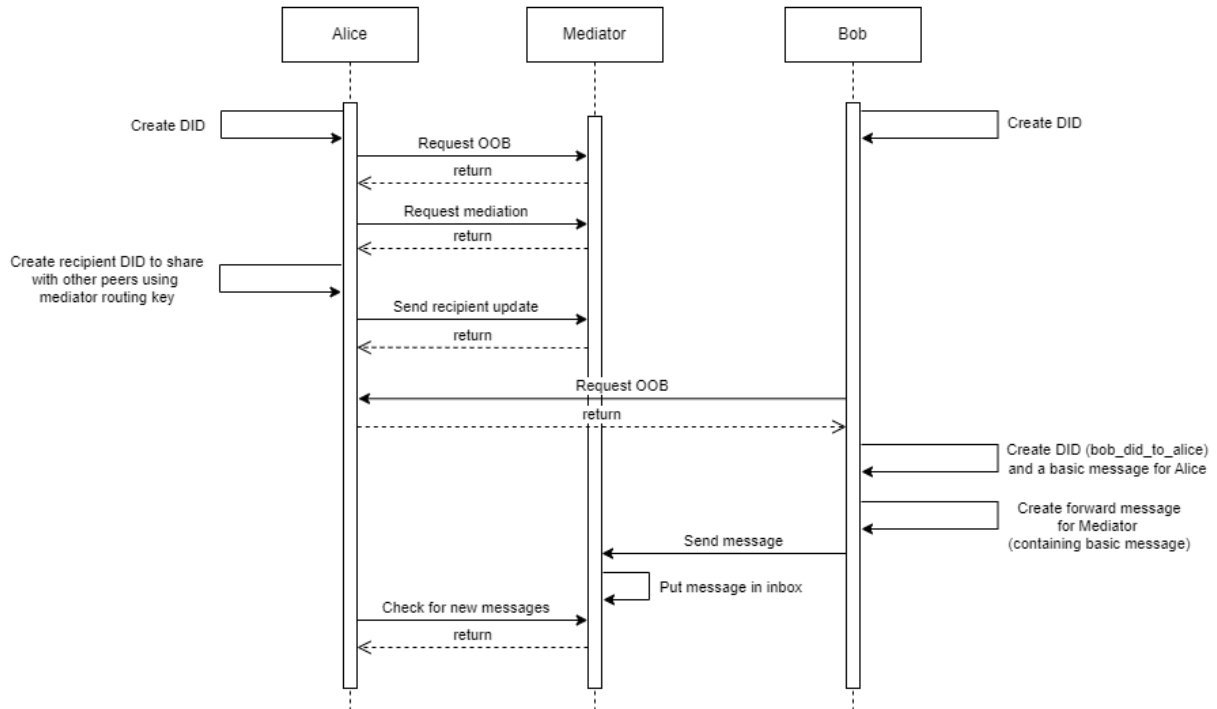
This markdown file should include the

 - 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)

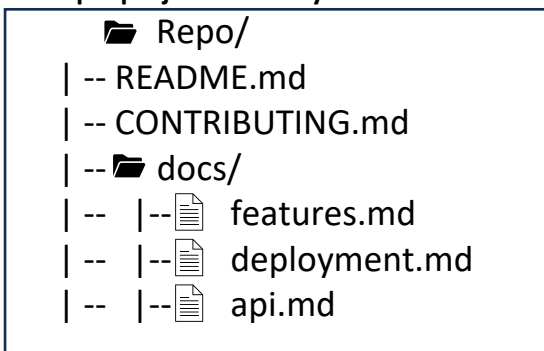
² <https://eclipse-xfsc.github.io/landingpage/developer-guide/on-boarding>

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 and docker deployments.	
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: namespace, ingress, service, deployment
- External dependencies (Keycloak, trust endpoints) reachable

BDD Test Suite

Must include executable scenarios for:

- Successful deployment
- Invalid parameters
- Idempotent redeployment
- Uninstall
- Integration tests (Keycloak, policies, tokens, etc.)

BDD Packs

- Deliver BDD packs with each release.

Target hosting environments for QA

- Reference cluster: IONOS Cloud and OSC.

- 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, BDD reports.

Architecture Decision Records (ADR)

ADR 001 – Deployment Engine

Decision: Helm is the mandatory deployment engine for all ESB modules³.

Consequence: Repeatable Kubernetes-native deployments.

ADR 002 – Orchestration Layer

Decision: ORCE is the mandatory runtime for Builder execution.

Consequence: Unified low-code deployment flow and JSON output.

ADR 003 – Automation Stack

Decision: Node.js + Bash + YAML for UI, backend logic, and automation.

Consequence: Stable and consistent execution environment.

ADR 004 – Identity Integration

Decision: Keycloak mandatory for realm and client configuration.

Consequence: Single consistent identity model across ESB modules.

ADR 005 – Security Baseline

Decision: TLS 1.3, RBAC, Secrets for credential storage, 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.

³ [https://github.com/eclipse-xfsc/smartdeployment/tree/main/Easy%20Stack%20Builder%20\(ESB\)](https://github.com/eclipse-xfsc/smartdeployment/tree/main/Easy%20Stack%20Builder%20(ESB))