

Technical Development Requirements

for

Federation Architecture for Composed Infrastructure Services (FACIS)

Federation Architecture Pattern Principal Credential Issuance (PCI)

FACIS.FAP_PCI

Scope Management

For a proper scope and requirement management, this specification has several side conditions which must be considered during the implementation:

- Preferred development mode is Kanban or Scrum
- All software within the scope must be released under Apache 2.0
- UX concept must be presented before implementation
- Repository structure must be part of the project plan for proper preparation
- Bi-weekly presentation of progress
- Milestone presentations must consist of BDD tests.

Open-Source Software

- **Mandatory use of XFSC Orchestration Engine (ORCE)¹ for**
 - Service Orchestration
 - Builder Node orchestration
 - Deployment workflows (if applicable)
 - Low-code integration
- **Deployment on Kubernetes**
 - Deployment on Kubernetes clusters on 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. The final repository 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. 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 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>

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

- **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.
- A project git repository will be made available, and the contractor is 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 Open Sovereign Cloud (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 backend services

Dependencies to mandated XFSC components

Any issues with mandated XFSC components must be reported promptly in writing to the client, with optional suggestions for bug fixes or enhancements. Any resulting additional work may require a formal change request (please refer to contract template in the tender documents) and client approval.

Documentation

- All activities according to Eclipse Handbook. <https://www.eclipse.org/projects/handbook/>
- FAP Partner Onboarding must be using for reference:
 - Specifications: [https://github.com/eclipse-xfsc/facis-fap/tree/main/Partner%20Onboarding%20\(Reference%20FAP\)/specification](https://github.com/eclipse-xfsc/facis-fap/tree/main/Partner%20Onboarding%20(Reference%20FAP)/specification)
 - Implementation: [GitHub – Partner Onboarding \(Reference FAP\)](#)
- Submit BDD documentation as Milestone (no later than one month after project plan approval).
- Local Repo:
 - 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 of 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 onboard into XFSC Eclipse Project.

General Documentation Guide

- Generator
 - readthedoc
 - mkdocs
- Product documentation
 - AsciiDoc
 - RTS
- Architecture modelling
 - Archimate
 - UML
 - <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 will be made available by the client.
- Project repository: <https://github.com/eclipse-xfsc>

README.md

This markdown file should include the

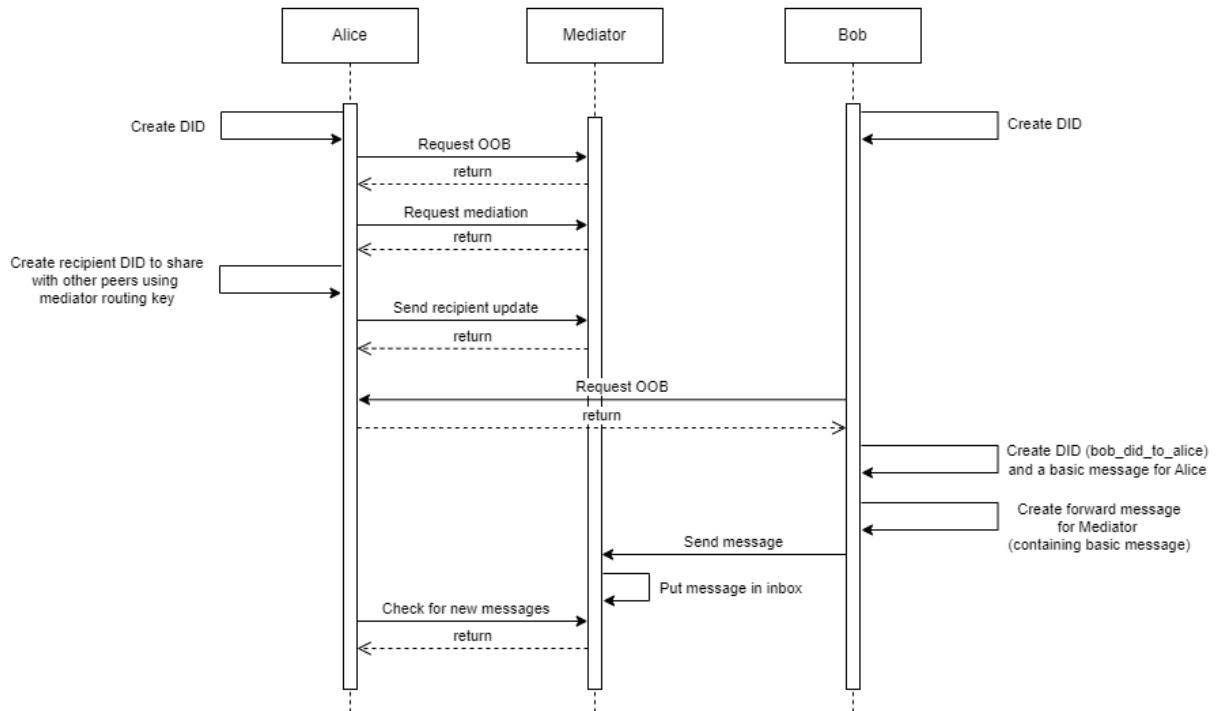
- Project overview (brief description of the microservice),
- Installation & setup (dependencies, environment variables and configuration)
- Technical documentation Links

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

- use case diagram in markdown compatible language e.g. mermaid (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:

```

Repo/
|-- README.md
|-- CONTRIBUTING.md
|-- docs/
|-- |-- features.md
|-- |-- deployment.md
|-- |-- api.md
    
```

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>”

FOSS Microservice Repository

File/ Folder	Purpose	Status (✓/✗)
.github/workflows	The eclipse dash liscense 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	
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 expectation for the community – to be finalized with Principal	
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)	

Note: some of the documents are already stored by Eclipse Foundation.

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 Suite³

- Must include executable scenarios for:
 - Successful deployment
 - Invalid parameters

³ <https://github.com/eclipse-xfsc/bdd-executor>

- Idempotent redeployment
- Uninstall
- Integration/Validation tests (Keycloak, policies, tokens, etc.)

BDD packs

Deliver BDD packs with each release.

Target hosting environments for QA

- The Reference hosting environment for the cloud trust zone is setup in the IONOS cloud and T-System Open Sovereign Cloud (OSC).
- XFSC Identity and Trust services
 - Keycloak
 - OCM
 - PCM Mobile
 - TSA
 - OpenBao (former HashiCorp Vault)
- ORCE

DevOps tools

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