mojaloop foundation

Reference Architecture and V2 Update

Jan 2022

Ref Arch and vNext Team

Agenda

- Reference Architecture Documentation Update
- vNext Build Update

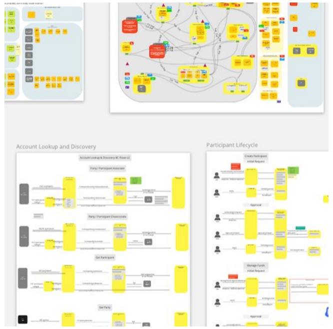
mojaloop
foundation

Initial Version Released!

Ref. Architecture Documentation

Ref. Architecture Documentation

- Working Docs



Miroboard

PI 14 Reference Architecture Working Doc

Overall plan

- Phase 1 - Complete Ref Arch and Deliver Implementation Architecture
- Phase 2 - Build the foundation for V2
- Phase 3 - Consolidate and Release

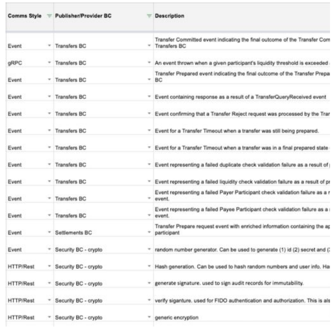
Objectives of this PI:

- Agreed Reference Architecture
- Agreed Implementation Architecture
- Build all the breaking changes starting this PI, to release a beta with the final structure ASAP
- Build Plan for next PI

Relevant docs:

- This doc has all notes and will later serve as the main documentation source
- Miro board is: [illegible]
- Google spreadsheet with build plans and interface matrix: [illegible]
- PI14 Convening presentation slides: [illegible]
- Michael Wagener proper version of the Ref Arch Documentation (to be replaced by VuePress doc site) - [illegible]
- VuePress documentation repository (Later we'll decide if we keep this separate from the main Mojaloop doc repo or merge this with it): https://github.com/mojaloop/reference-architecture-doc
  - This is being automatically published here: https://ref-arch-docs.moja-lab.live/ (use this if that doesn't work [illegible]

Google Docs

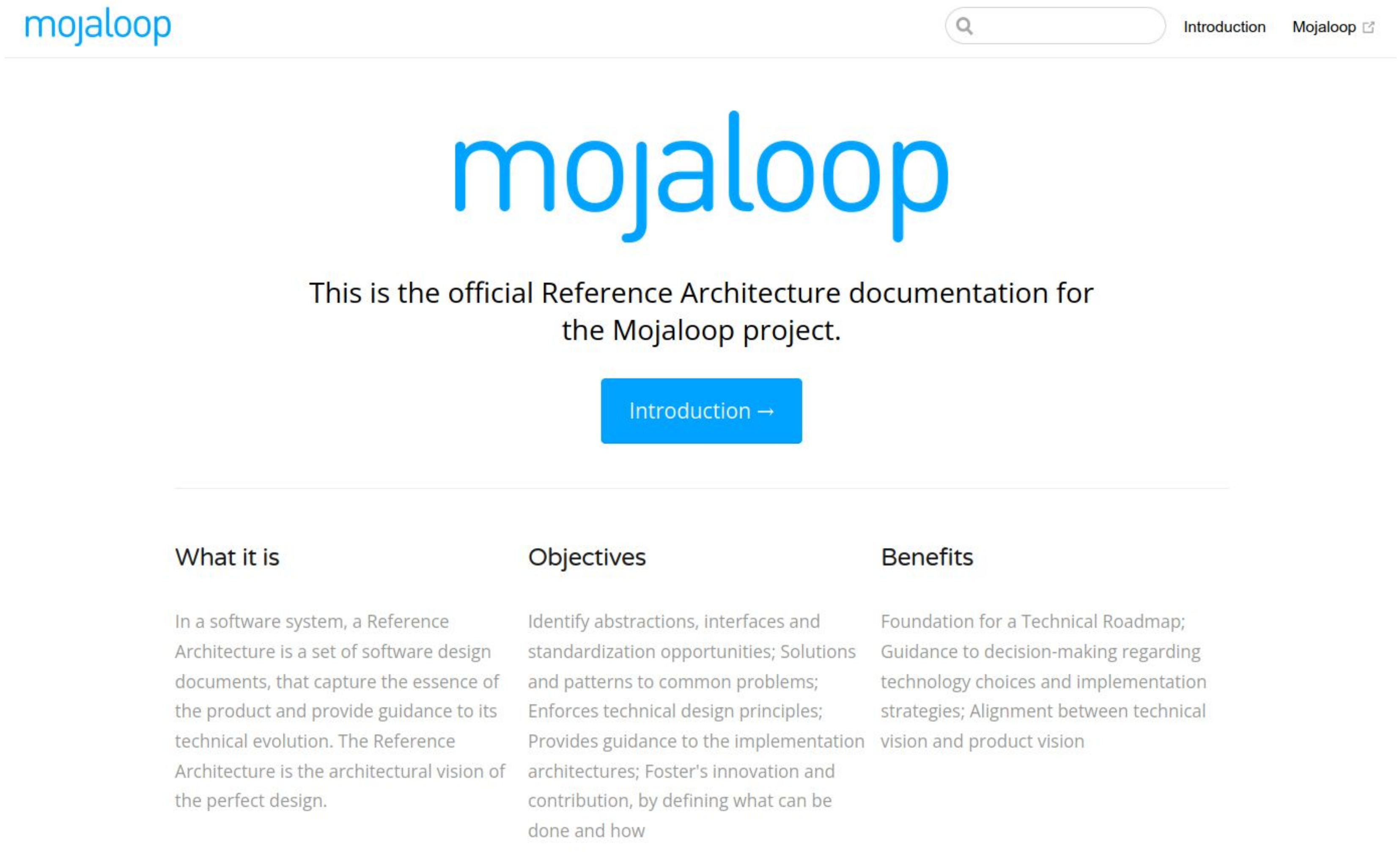


Comms Style	Publisher/Provider BC	Description
Event	Transfers BC	Transfer Committed event indicating the final outcome of the Transfer Com Transfers BC
gRPC	Transfers BC	An event thrown when a given participant's liquidity threshold is exceeded
Event	Transfers BC	Transfer Prepared event indicating the final outcome of the Transfer Prepa BC
Event	Transfers BC	Event containing response as a result of a TransferQueryReceived event
Event	Transfers BC	Event confirming that a Transfer Reject request was processed by the Trar
Event	Transfers BC	Event for a Transfer Timeout when a transfer was still being prepared.
Event	Transfers BC	Event for a Transfer Timeout when a transfer was in a final prepared state
Event	Transfers BC	Event representing a failed duplicate check validation failure as a result of
Event	Transfers BC	Event representing a failed liquidity check validation failure as a result of pr
Event	Transfers BC	Event representing a failed Payer Participant check validation failure as a r event.
Event	Transfers BC	Event representing a failed Payee Participant check validation failure as a event.
Event	Settlements BC	Transfer Prepare request event with enriched information containing the ap participant
Event	Security BC - crypto	random number generator. Can be used to generate (1) id (2) secret and (
HTTP/Rest	Security BC - crypto	Hash generation. Can be used to hash random numbers and user info. Ha
HTTP/Rest	Security BC - crypto	generate signature. used to sign audit records for immutability.
HTTP/Rest	Security BC - crypto	verify siganture. used for FIDO authentication and authorization. This is als
HTTP/Rest	Security BC - crypto	generic encryption

Google Sheets

- Published Version

https://docs.mojaloop.io/reference-architecture-doc/



mojaloop

This is the official Reference Architecture documentation for the Mojaloop project.

What it is

In a software system, a Reference Architecture is a set of software design documents, that capture the essence of the product and provide guidance to its technical evolution. The Reference Architecture is the architectural vision of the perfect design.

Objectives

Identify abstractions, interfaces and standardization opportunities; Solutions and patterns to common problems; Enforces technical design principles; Provides guidance to the implementation architectures; Foster's innovation and contribution, by defining what can be done and how

Benefits

Foundation for a Technical Roadmap; Guidance to decision-making regarding technology choices and implementation strategies; Alignment between technical vision and product vision

Introduction
What is a Reference Architecture?
Objectives
Benefits of having a Reference Architecture
Process of creating and maintaining the Reference Architecture
How to keep it up-to-date
Principles guiding this architecture
Reference Architecture Overview
Bounded Contexts
Common Terms & Conventions
Account Lookup & Discovery
Accounts & Balances
Quoting/Agreements
Auditing
FSP Interop API
Logging
Notifications And Alerts
Participant Lifecycle Management
Platform Monitoring
Reporting
Scheduling
Security
Settlements
Third Party API
Transfers
Common Interface List
How to Implement
Glossary
Further Reading

Introduction

What is a Reference Architecture?

In a software system, a Reference Architecture is a set of software design documents, that capture the essence of the product and provide guidance to its technical evolution.

This concept can be further simplified:
The Reference Architecture is the architectural vision of the perfect design.

In normal conditions that perfect design is never achieved, partly because there is neither enough time nor resources to fully implement it, partly because that design can iterate and improve faster than it can be implemented.
It is the nature of a Reference Architecture to be a living document that is continuously updated and enhanced.

Objectives

Its main objectives are to:

- Identify abstractions, interfaces and standardization opportunities
- Propose solutions and patterns to common problems
- Help enforcing technical design principles
- Provide guidance to the implementation architectures
- Foster innovation and contribution, by defining what can be done and how

Benefits of having a Reference Architecture

The first benefit is that it is the perfect foundation for a Technical Roadmap. By having the future vision in perspective, a phased Technical Roadmap can easily be created from it, ensuring resources and attention are focused on the long term value.

Another important benefit is the guidance it provides to decision-making regarding technology choices and implementation strategies.
With the Reference Architecture in mind we can frame any development decision as tactical or strategic:

- Tactical - something that is required right now which may have exceptions to the Reference Architecture for the sake of a high-value urgent requirement. These exceptions should be documented as technical debt which can be addressed in future.
- Strategic - something that is long-lasting and should be implemented in accordance to the Reference Architecture - should take the implementation closer to it.

Last but not less important, it ensures alignment between the technical vision and the more important product vision (see below regarding the process and ways of working).

Process of creating and maintaining the Reference Architecture

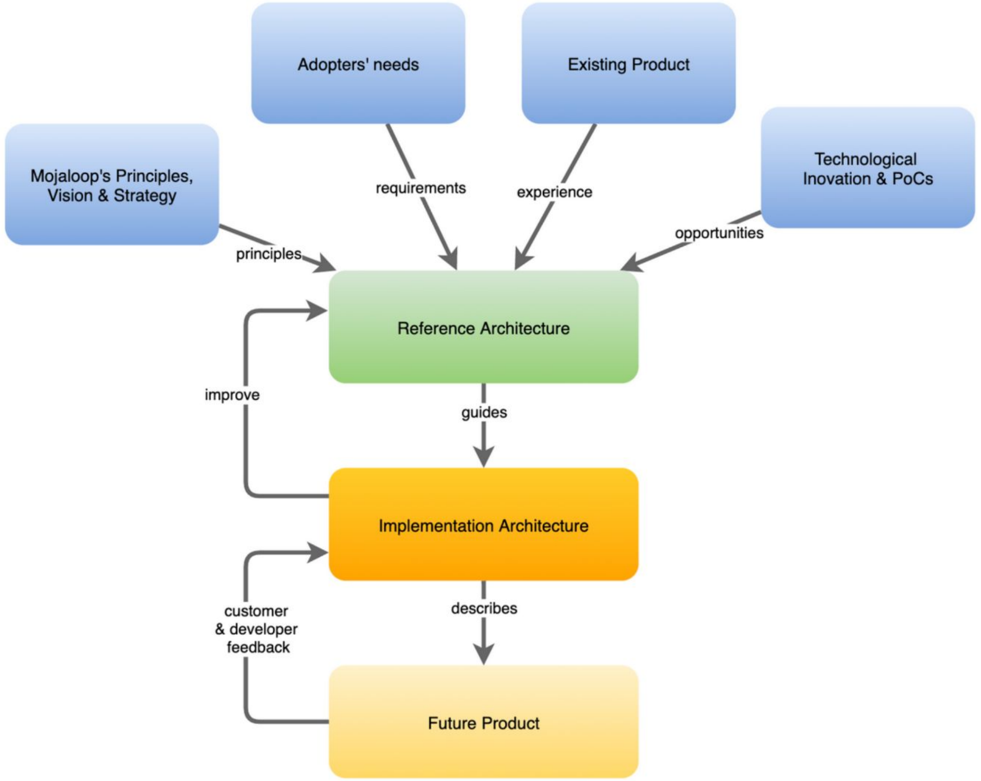
While creating the initial version of the Reference Architecture, the team followed these steps:

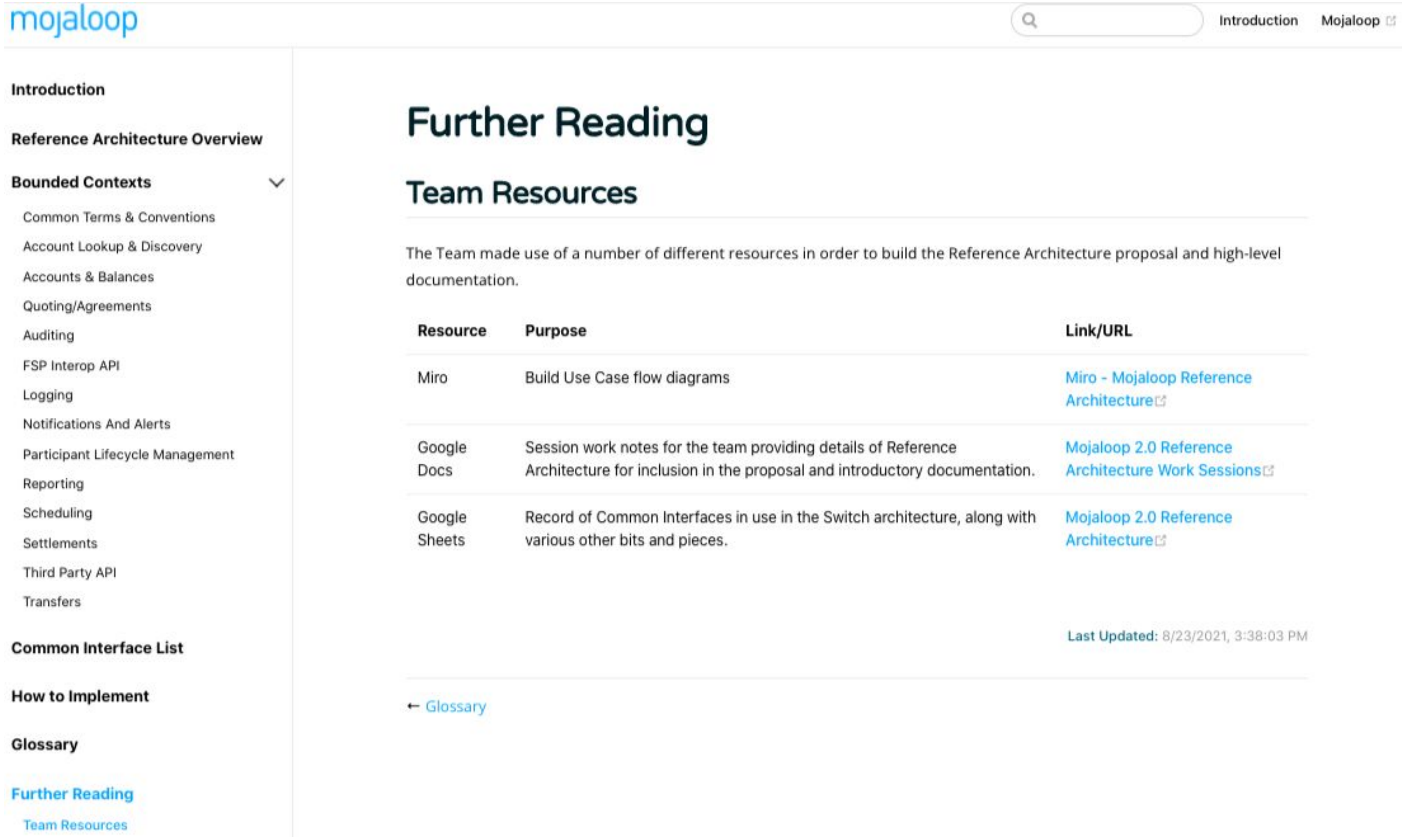
Introduction
What is a Reference Architecture?
Objectives
Benefits of having a Reference Architecture
Process of creating and maintaining the Reference Architecture
How to keep it up-to-date
Principles guiding this architecture
Reference Architecture Overview
Bounded Contexts
Common Terms & Conventions
Account Lookup & Discovery
Accounts & Balances
Quoting/Agreements
Auditing
FSP Interop API
Logging
Notifications And Alerts
Participant Lifecycle Management
Platform Monitoring
Reporting
Scheduling
Security
Settlements
Third Party API
Transfers
Common Interface List
How to Implement
Glossary
Further Reading

2. Solution Space Context mapping - Group similar problems together based on their purpose and context
3. Individual use case mapping - Discuss and document use cases in detail taking into consideration the entire solution

How to keep it up-to-date

The diagram below shows where a reference architecture exists in reference to other processes of the Mojaloop Platform; what it must incorporate and understand, not only the vision and the principles, but also the requirements, previous experience and even forster technical innovation.





Introduction

Reference Architecture Overview

Bounded Contexts

Common Terms & Conventions

Account Lookup & Discovery

Accounts & Balances

Quoting/Agreements

Auditing

FSP Interop API

Logging

Notifications And Alerts

Participant Lifecycle Management

Reporting

Scheduling

Settlements

Third Party API

Transfers

Common Interface List

How to Implement

Glossary

Further Reading

Team Resources

Further Reading

Team Resources

The Team made use of a number of different resources in order to build the Reference Architecture proposal and high-level documentation.

Resource	Purpose	Link/URL
Miro	Build Use Case flow diagrams	Miro - Mojaloop Reference Architecture
Google Docs	Session work notes for the team providing details of Reference Architecture for inclusion in the proposal and introductory documentation.	Mojaloop 2.0 Reference Architecture Work Sessions
Google Sheets	Record of Common Interfaces in use in the Switch architecture, along with various other bits and pieces.	Mojaloop 2.0 Reference Architecture

Last Updated: 8/23/2021, 3:38:03 PM

← Glossary

mojaloop foundation

vNext Build

Alpha -> Beta -> Production!

vNext Alpha Release

Internal release for developers

Stable interfaces

Happy path only

Fully testable

Requires load tests, but no optimisation

vNext Alpha Release

Design

Build of the foundation & Cross Cutting Concerns

<- We're here!

Happy Path Features

vNext Alpha Release

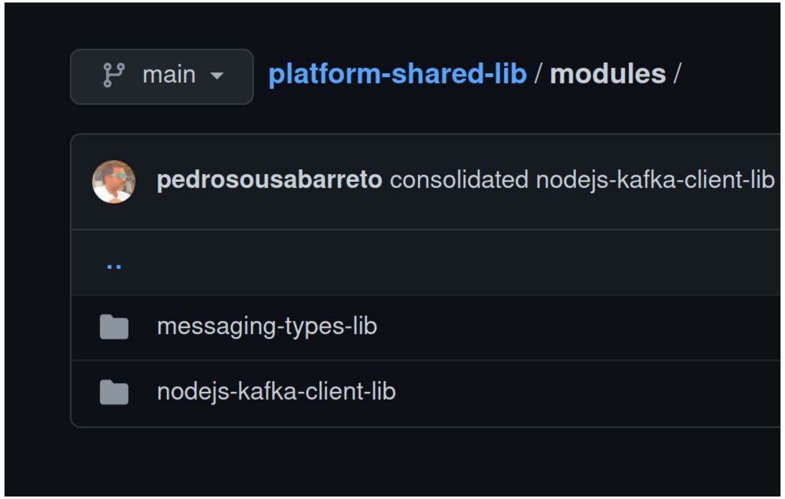
Alpha Release scheduled for May

After alpha, comes…

mojaloop foundation

vNext Alpha Release

Objective - decouple from infra when possible


main

platform-shared-lib / modules /

pedrosousabarreto consolidated nodejs-kafka-client-lib

..

messaging-types-lib

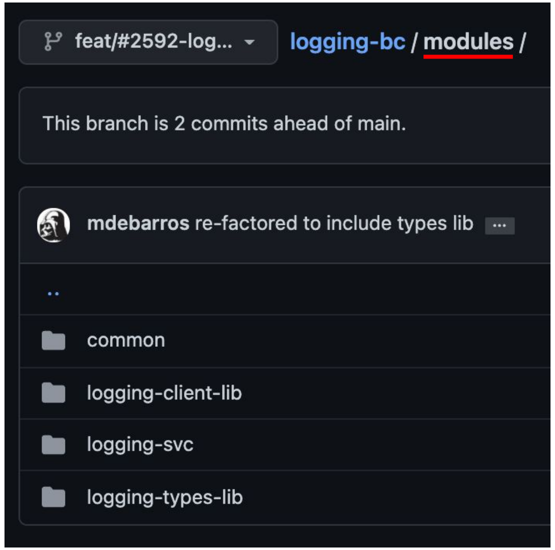
nodejs-kafka-client-lib

- Separate general types from actual implementations
- Design for extensibility and replaceability
- We should even be able to replace Kafka

mojaloop foundation

vNext Alpha Release

Objective - modularity through abstractions & interfaces


feat/#2592-log... logging-bc / modules /

This branch is 2 commits ahead of main.

mdebarros re-factored to include types lib ...

..

common

logging-client-lib

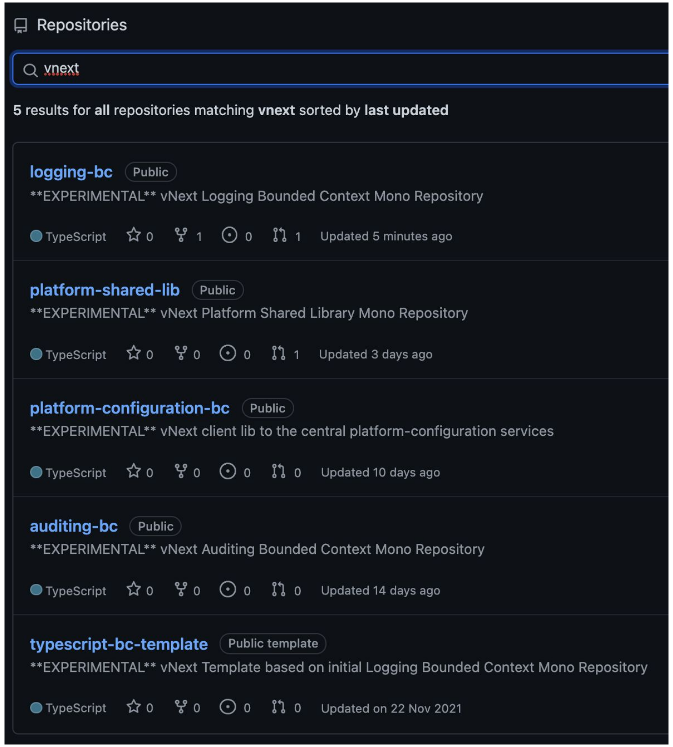
logging-svc

logging-types-lib

- De-coupling pattern through Interfaces (***-types-lib**)
- Dependency injection through concrete implementations using interfaces on both Server (***-svc**) and Client (***-client-lib**) side
- Allows for multiple different concrete implementations (i.e. different backend, and infrastructure)
- Example:
  - Single Client produces logs to Messaging infra (e.g. Kafka)
  - Multiple Service implementation consume log messages from Messaging infra and indexes them into:
    - Databases (MySQL, PostgreSQL)
    - Loki (Grafana)
    - ElasticSearch (EFK/ELK)
    - FluentD / FluentBit, etc
- Common internal lib which is specific to the BC's context & usage

mojaloop foundation

vNext Alpha Repositories



Repositories

vnext

5 results for **all** repositories matching **vnext** sorted by **last updated**

logging-bc Public
EXPERIMENTAL vNext Logging Bounded Context Mono Repository
TypeScript ☆ 0 1 0 1 Updated 5 minutes ago

platform-shared-lib Public
EXPERIMENTAL vNext Platform Shared Library Mono Repository
TypeScript ☆ 0 0 0 1 Updated 3 days ago

platform-configuration-bc Public
EXPERIMENTAL vNext client lib to the central platform-configuration services
TypeScript ☆ 0 0 0 0 Updated 10 days ago

auditing-bc Public
EXPERIMENTAL vNext Auditing Bounded Context Mono Repository
TypeScript ☆ 0 0 0 0 Updated 14 days ago

typescript-bc-template Public template
EXPERIMENTAL vNext Template based on initial Logging Bounded Context Mono Repository
TypeScript ☆ 0 0 0 0 Updated on 22 Nov 2021

https://bit.ly/32rFTVR

mojaloop foundation

Logging BC

Logging libs, clients and services

Platform Shared Lib

Common lib across all BC contexts (e.g. Messaging types/client-libs)

Platform Configuration BC

Common Configuration client /svc functions

Auditing BC

Auditing libs, clients and services

Typescript BC Template

Template to bootstrap BCs with mono repo tooling, basic structure and standards

We really need help!!

We STILL need developers willing to work on this effort

Questions?

Reading / References?

Official Ref Arch Doc Page: here

Ref Arch Working Doc: here

Ref Arch Miro Board: here

Next version build plan working doc: here

PI-13 Jan 21 - Mojaloop Performance, Scalability and Architecture Update: here

PI-14 Apr 2021 - Opening and Mojaloop Reference Architecture and TigerBeetle: here

PI-15 July 2021 - Reference Architecture v1.0: here, videos

PI-16 Oct 2021 - Reference Architecture and V2 update: here, videos

mojaloop foundation