ONAP API Fabric (API GW) Proposal

Proposed by：NetCracker Technology
Supported by : Vodafone, Swisscom, Verizon

12th June 2019

Agenda

- Problem Statement
- Proposal
- Usage Scenarios
- Discussion Summary So far & Suggested Next Steps

What we improved from previous presentations

1. Renamed the project to avoid confusion with individual views on API GW or Service Mesh or Anything Else which is planned

2. Focus on the concept/solution scope than where/how it is implemented

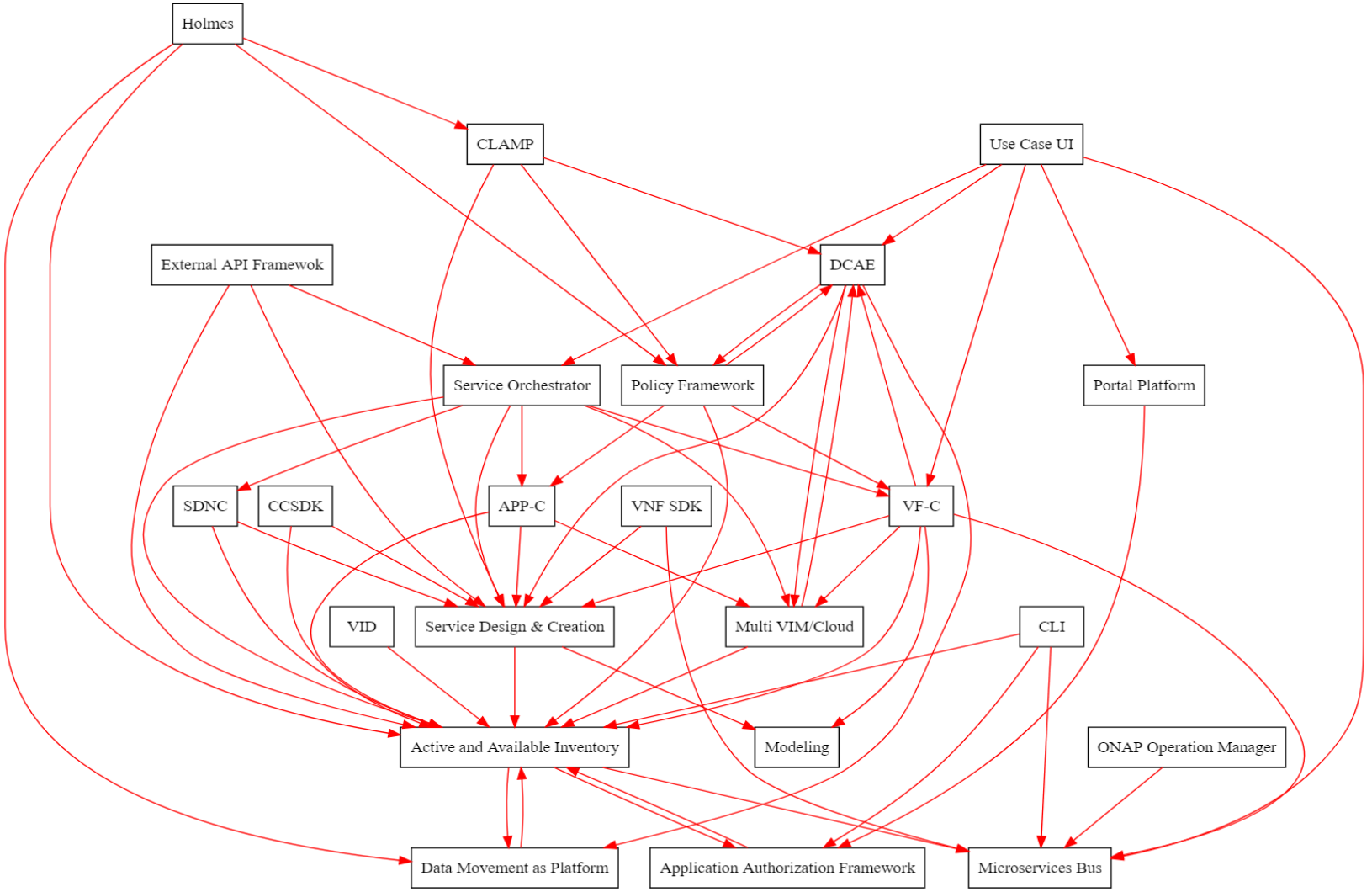
3. Added Use Cases to explain the possible scenarios where the proposed solution is useful

4. Added suggested next step - Discussions can continue in parallel

Section 1

Problem Statement

ONAP Component Dependency Matrix



ONAP Projects API Dependencies

Do we need to expose this dependency to end user, use case developer, component developer ?

Can we hide it behind an abstraction layer ?

Problem Statement

Complexity of APIs : Fine-Grained APIs exposed for consumption

- Each component exposes fine grained capabilities, not all will be necessary always and might confuse the business consumer.
- API consumer need to depend on the fine-grained component level API intricacies, Entity model rather than what is necessary and sufficient (hiding the complexity)
- Cross dependency between components require enrichment through multiple API calls

Standard Alignment is a priority in ONAP

- Enhancing multiple components for standard API alignment is time consuming
- Redundant API adaptation logic across different components that cannot be reused – e.g. SOL003 adaptor in SO , VFC and SDNC
- Overhead on project teams to manage standard adaptation/abstraction than core functionality

Production deployments might require interoperability with legacy and 3rd Party components

- Need for an API abstraction /façade layer rather than point to point integration with each component
- Capability to compose APIs exposed by different components at different levels of abstraction and integration with 3rd party , External/Internal Components

Evolution of Platform functional capability vs. Use Case capability:

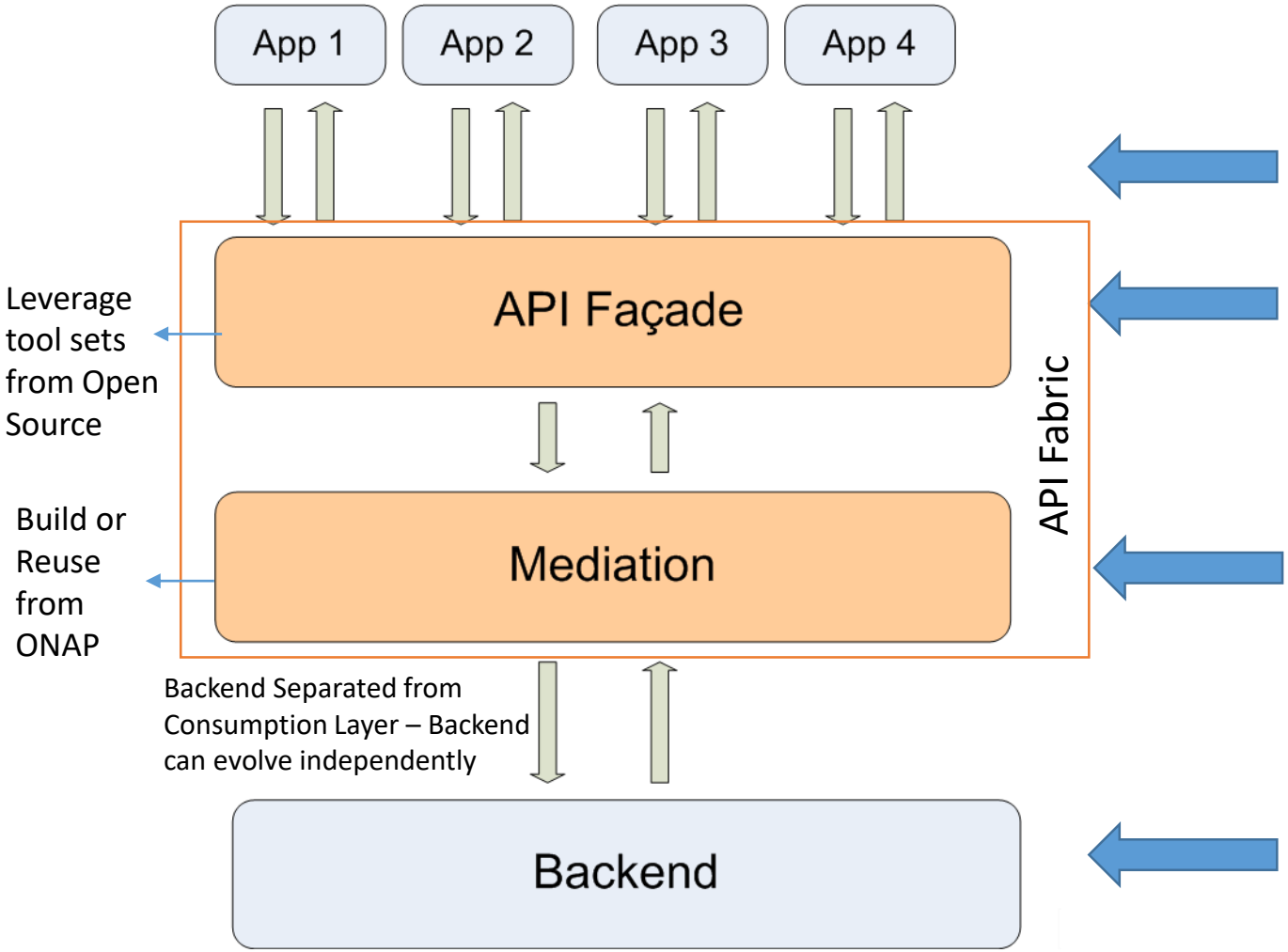
Platform need to evolve independently, not strictly based on use cases:

- Missing an appropriate facade layer to isolate these two needs
- Use cases typically expect standard/composite APIs for wider acceptance and adoption, project specific API alignment roadmap not completely in sync with use cases and delay the use case development.

Section 1

Proposal

API Fabric : Solution Guideline



Use Case Specific Simplified APIs exposed , different abstraction levels

Consistent API Management ,Security, Identity, Composition , Policy control

Transformation , Adaptation, Enrichment – Used for Standard API adoption

Existing ONAP Component Level Fine-grained APIs

Handle common API transactional requirements at Façade, Focus on integration of adaptation logic in Mediation

What is being proposed ?

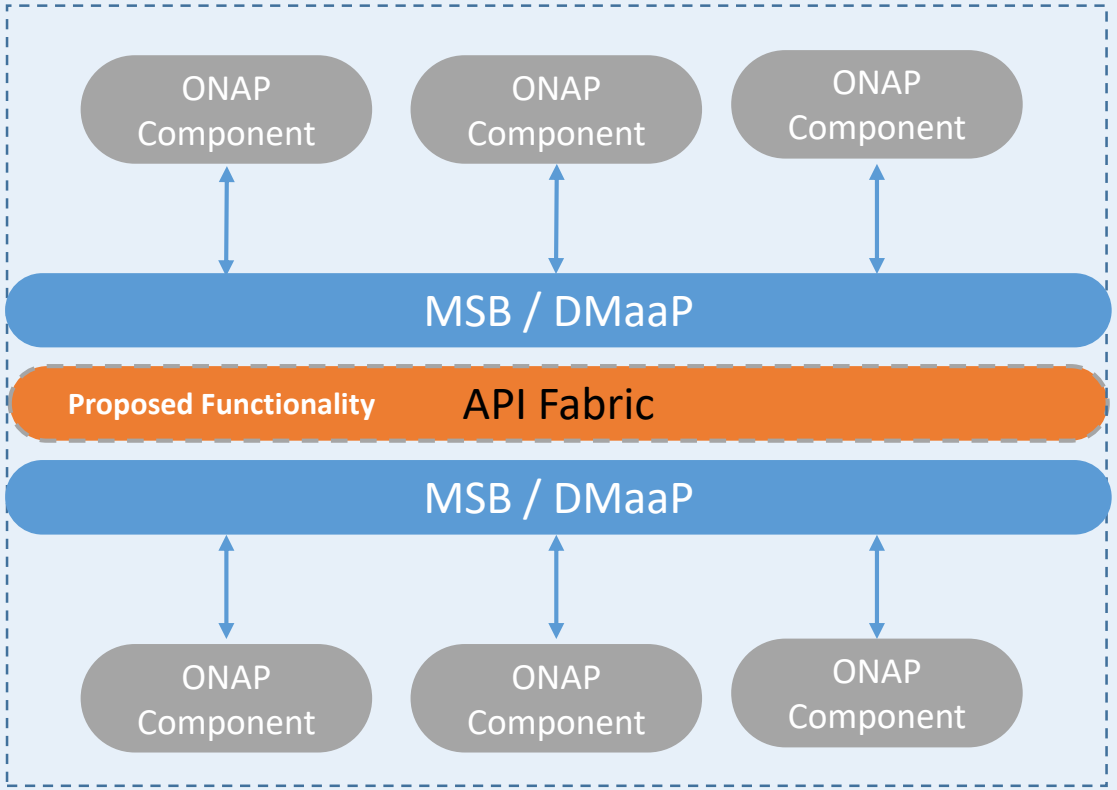
- **Create level of abstraction over component level APIs (façade) so that ONAP capabilities can be consumed easily without knowing how it is implemented internally**
  - Analogy : ONAP CLI (user need not know the complexities of ONAP Component level APIs), Ext-API (user just need to know the standard TMF APIs to work with ONAP)
- **Mediate and Adapt the APIs to desired format**
  - Patterns for adaptation are more or less same. Ease Standard API Adaptor development, Integration with third party solutions, composition of ONAP APIs to a more consumable format etc.
- **Enable ease of ONAP capability consumption**
  - Does not differentiate internal or external consumption – objective is to ease any type of consumption
- **Set of reusable tool set to manage APIs, Develop Adaptors, Manage Consumption**

Proposal: API Fabric

A centralized function that acts as an API Fabric that consists of toolsets to create Facades, and enablers/plugins to mediate API across different logical layers .

FEATURES:

- Facilitate development of Façade APIs
- Consolidates Façade API Management in a single logical function
- Augments Integration Layer capabilities in ONAP
- Reuse API Routing Functions available in MSB
- Supports Plugin model to attach request and response transformation logic
- Offload common API tasks from other ONAP Components



API Fabric Solution Scope

API Façade

- Model Driven API Management – Swagger import, LCM management (version, canary, artifacts/plugin association)
- API Composition toolsets – HTTP Callout and aggregation
- Integration with ONAP specific or external Auth Provider
- API Marketplace , Subscription Management, Plan Management
- API Policy Management – RBAC, Tenancy, Rate Limit, Quota, Circuit Break
- Documentation Tools
- Input Validation

API Mediation

- Script insertion – Groovy, Python or Custom
- Business logic insertion – Plugin SDK
- Transformation templates – JOLT , Velocity
- Expression Language – Query strings , Regular expression support
- Alert Generation and Control loop support
- Runtime Mediation Control
- Flexibility to support API variance – SOAP, REST, GraphQL, gRPC, XML, JSON

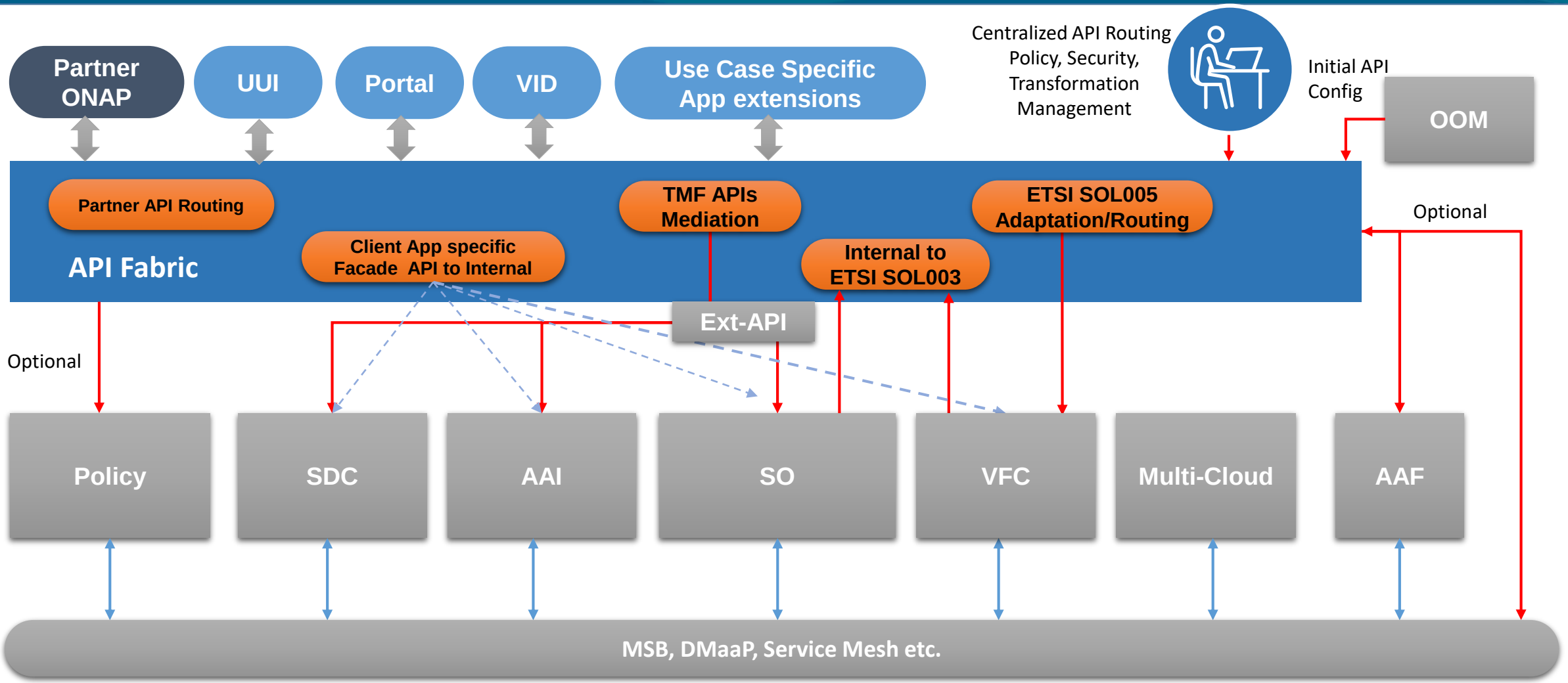
Common (including non-functional)

- API Monitoring, Metrics Collection, Analytics
- Cloud native friendly - Distributed, Microservice based, Scalable
- API Sharding , Canary Support
- Low Maintenance overhead
- Developer friendly toolsets , Low effort

Section 3

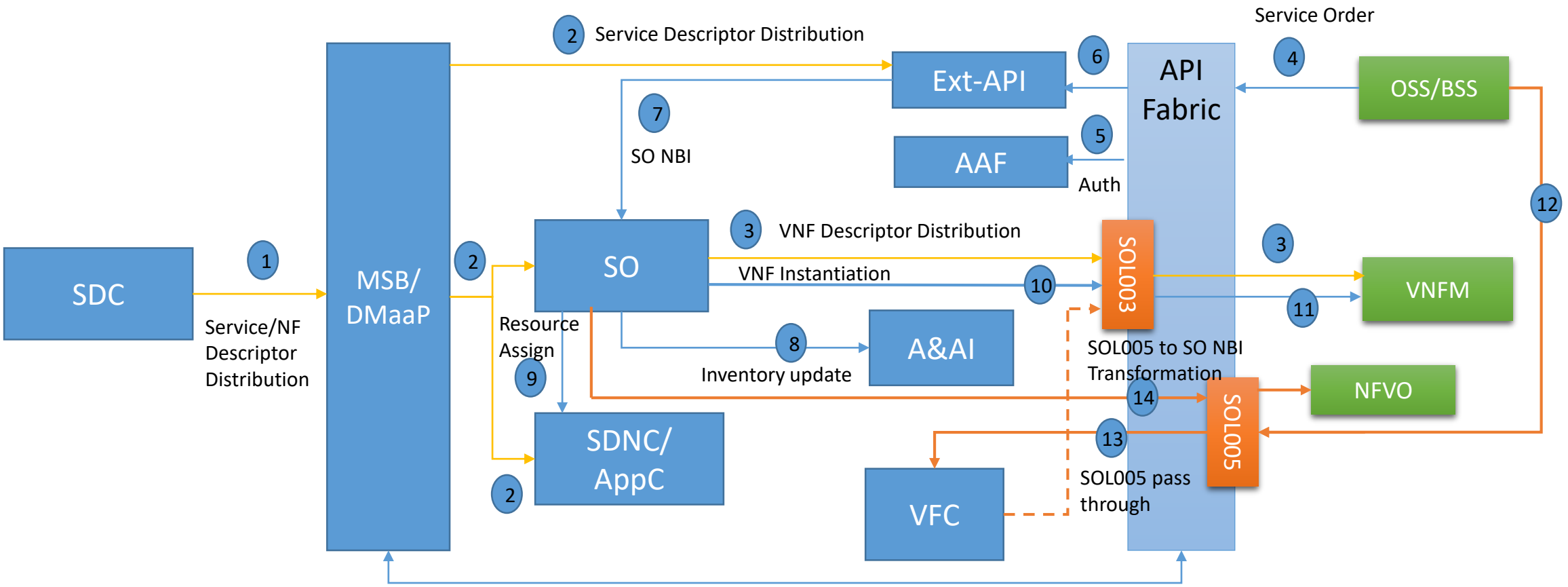
Usage Scenarios

ONAP API Fabric : As an API Consumption Layer

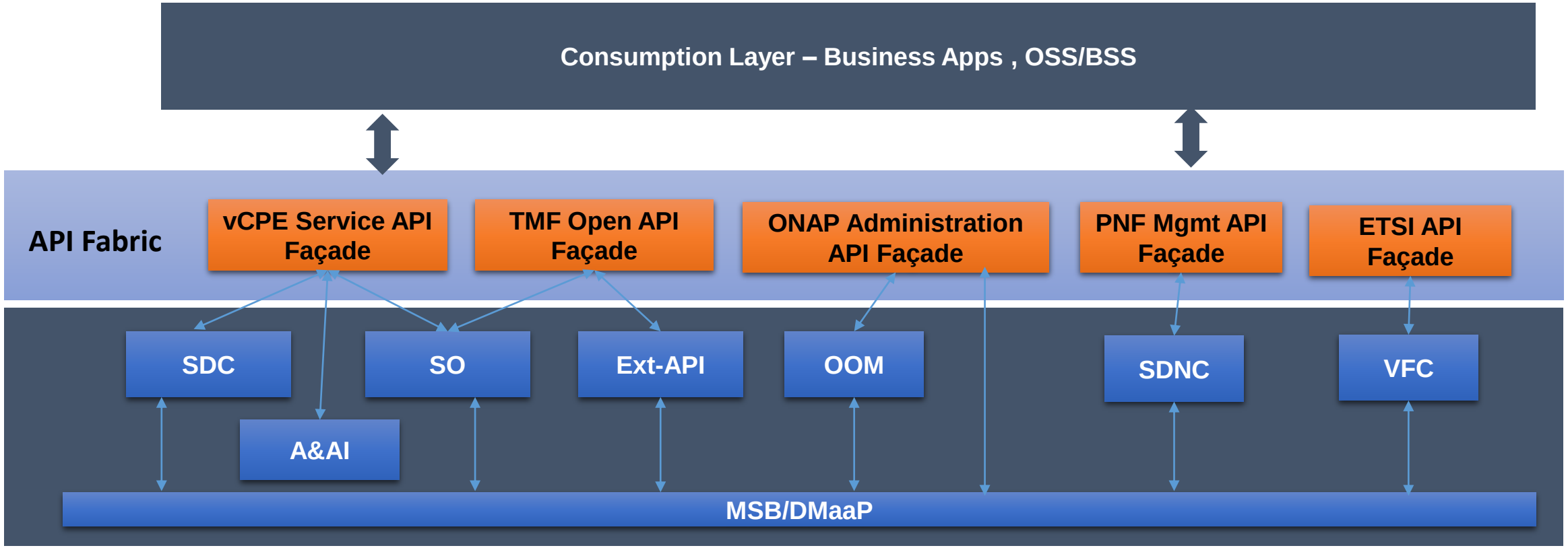


"Use the façade pattern when you want to provide a simple interface to a complex subsystem. Subsystems often get more complex as they evolve." - Design Patterns – Elements of Reusable Object-Oriented Software

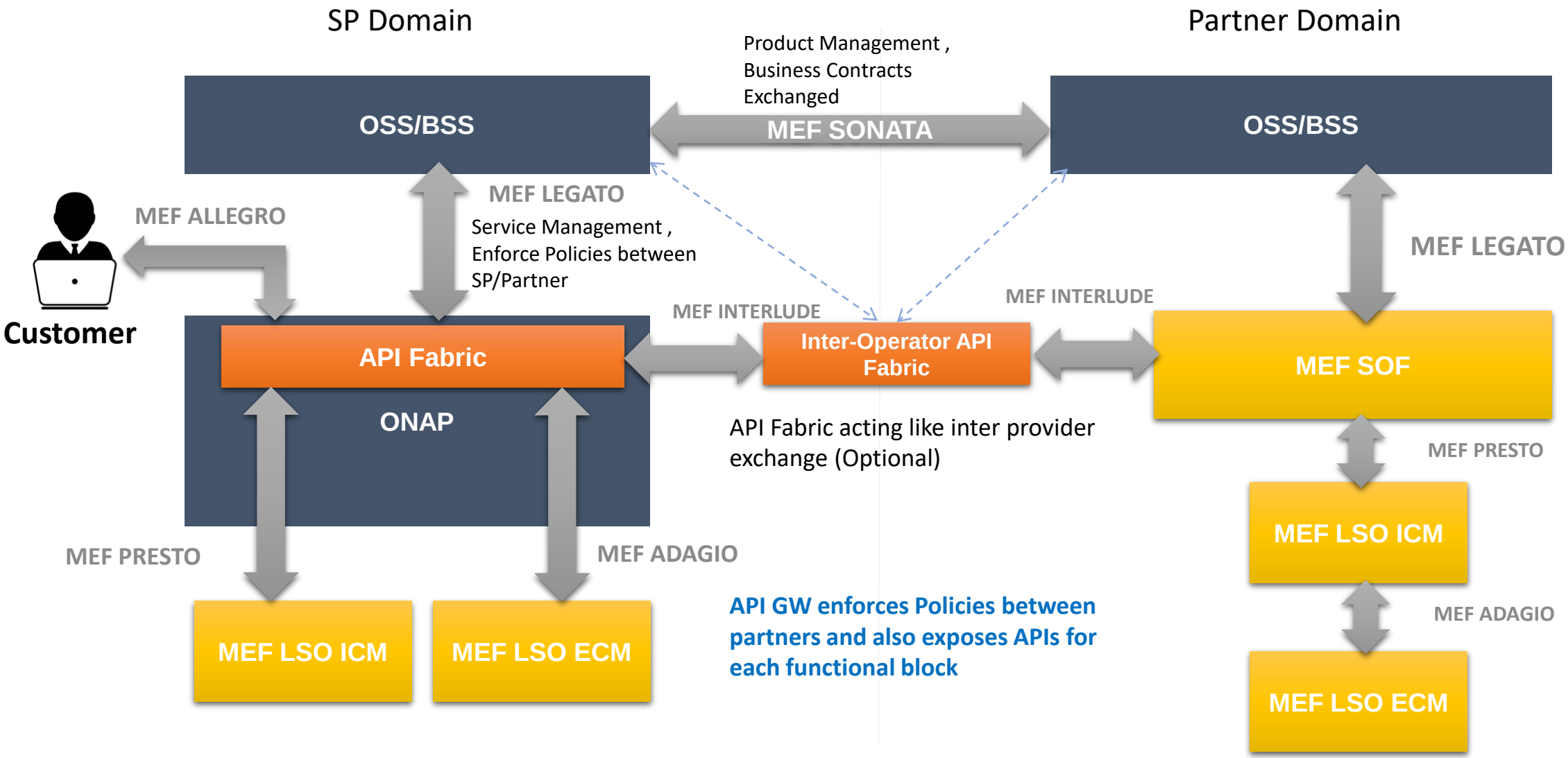
API Fabric as a Standard API Abstraction Layer



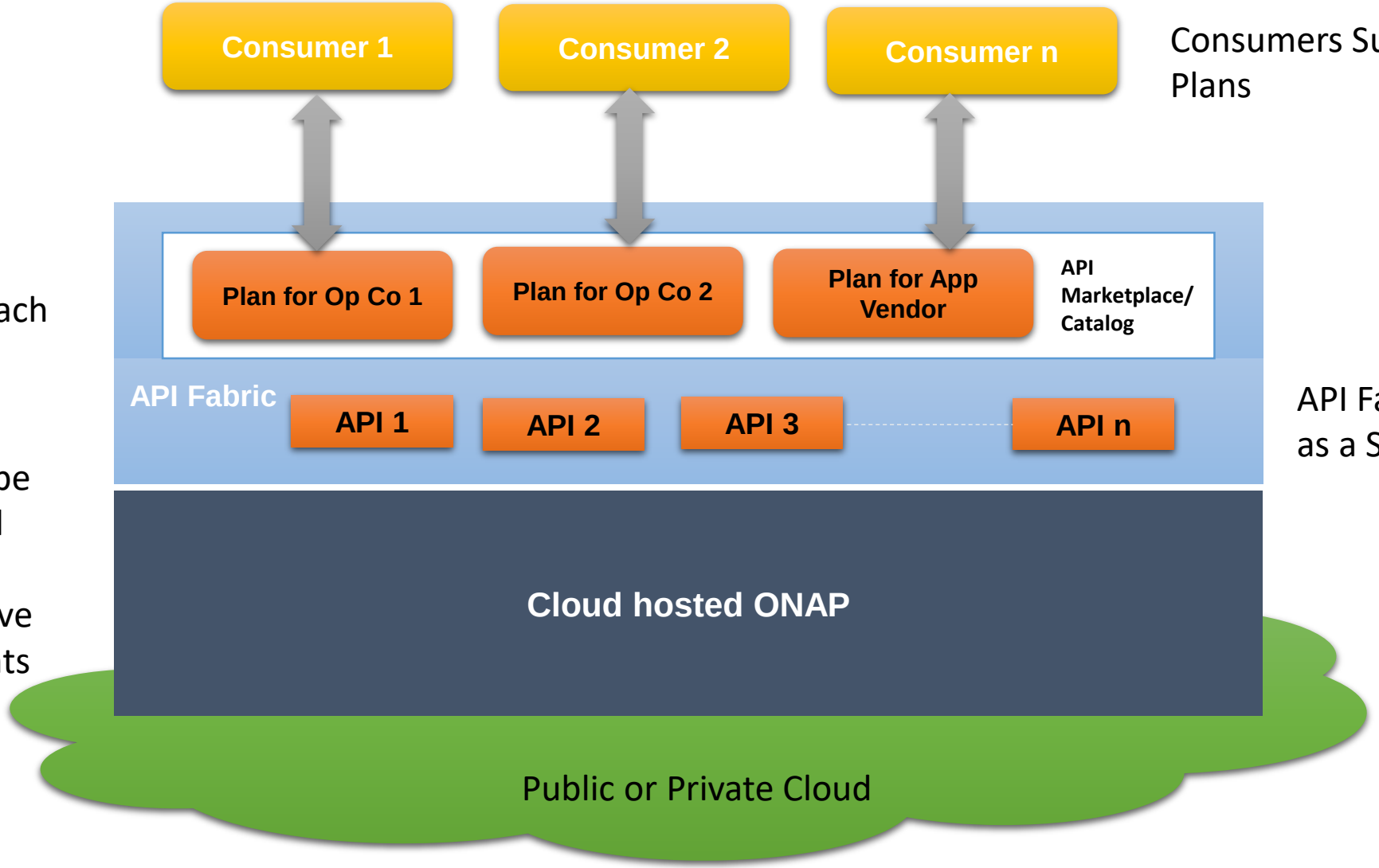
API Fabric as a generic Façade



API Fabric: As an Inter-Operator API Interaction Enabler with Security and Policy Management



API Fabric: As a Tenancy Management Enabler, ONAP As a Service



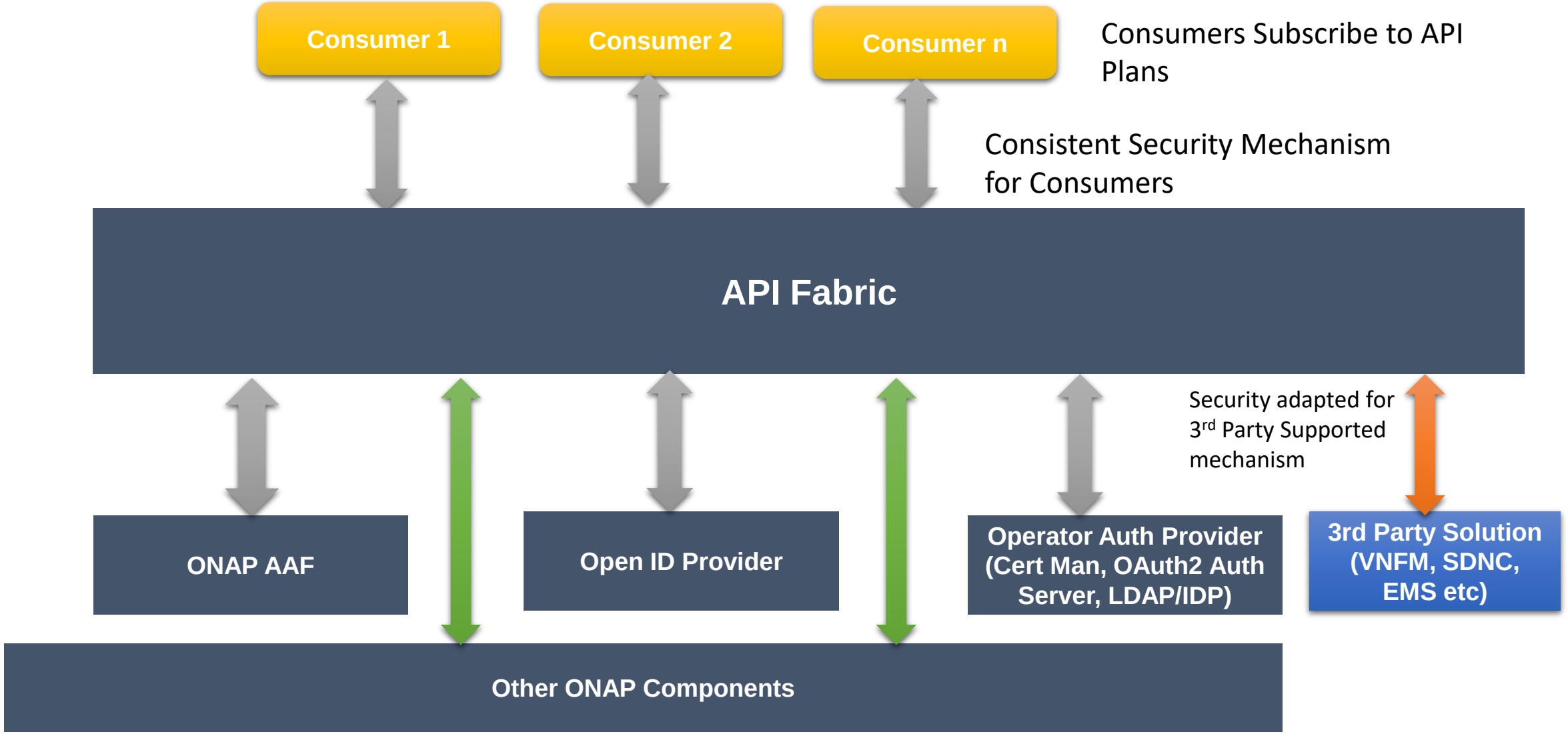
Consumers Subscribe to API Plans

API Plans are created for each Tenant

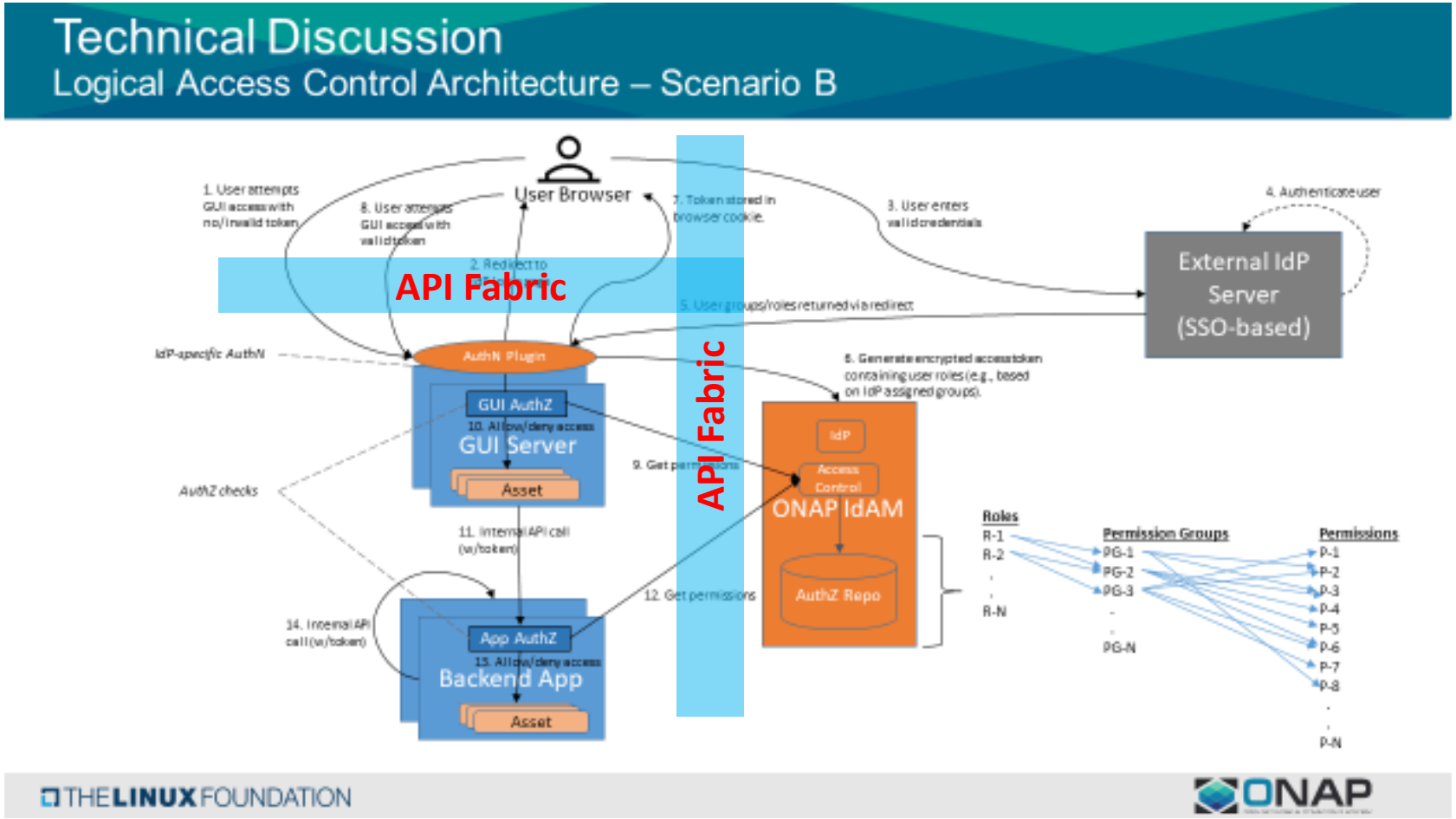
API Fabric Exposes ONAP as a Service

API Fabric can be distributed and APIs can be shard'ed to serve different Tenants

API Fabric : Interwork with multiple authentication providers and enable security mechanism



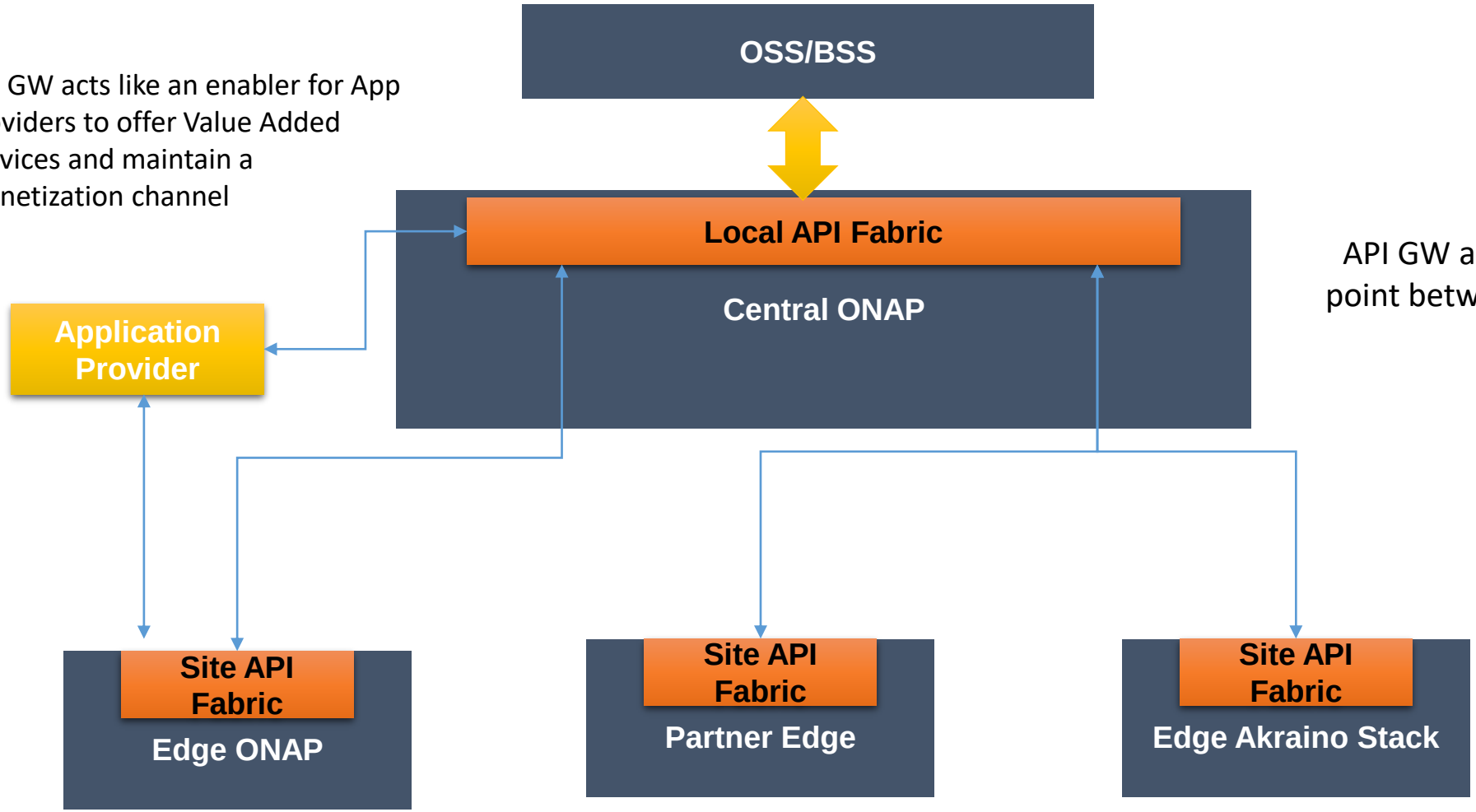
API Fabric : RBAC Enabler



Role of API Fabric in the Verizon proposal for RBAC

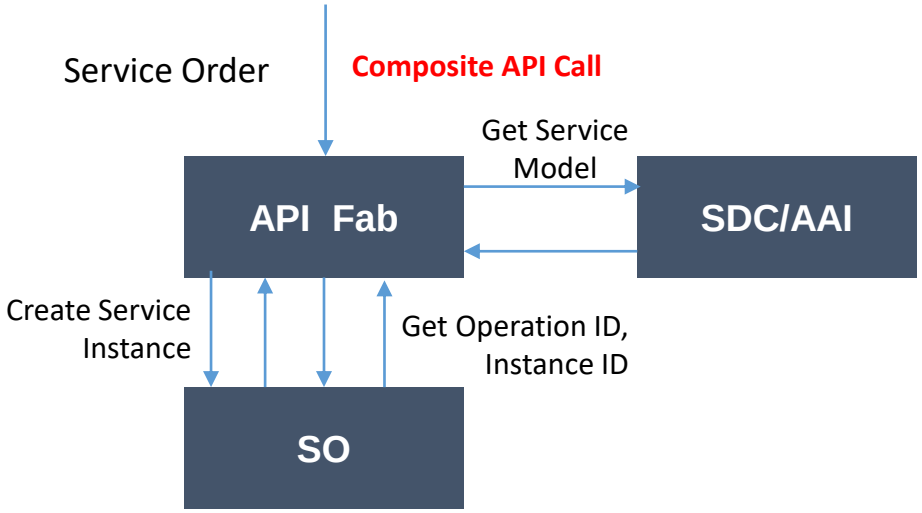
API Fabric : As an Edge Proxy

API GW acts like an enabler for App providers to offer Value Added Services and maintain a monetization channel

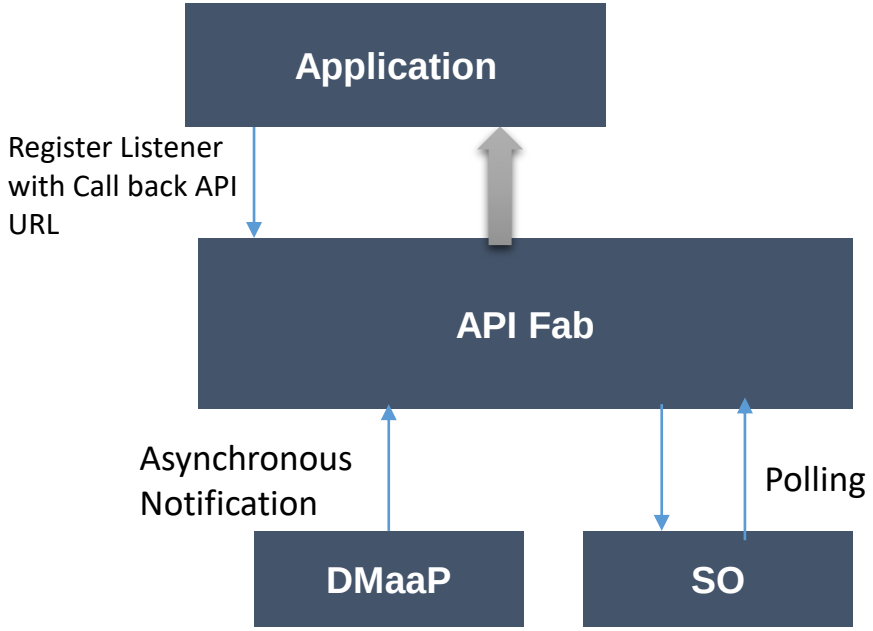


API GW acts like an anchoring point between Central and Edge Sites

API Fabric : Enabler for API Composition and Patterns



Single API call on API GW results in multiple atomic API calls in the backend. All atomic API calls chained in API GW through Configuration. Intermediate state and context cached in between atomic API calls



API GW used for Asynchronous Notification (Hub Resource Management)

Similar pattern followed in SOL003, Ext-API TMF APIs and SOL005.

Section 1

Discussion Summary So Far & Next Steps

Next Steps (Require Feedback from Community)

- Develop a PoC that showcase the façade API and API Mediation capability
  - Show a demo in the E time frame (Select one of the use cases, Suggested use cases in next two slides)
  - Can be based on multiple technologies depending on resource availability (sequentially or in parallel)
  - Showcase the merits , demerits from developer, end user point of view
- Selected technology option to get into E Release as experimental feature
- Formalize the PoC as part of a project or use case during F Release timeframe

Proposed Use Cases 1/2

Scenario 1: Dynamic Routing and Request/Response Transformation for SOL005 API

- API Fabric Exposing two types of interfaces
  - **Simplified internal API which hides SOL005 API or API exposed by external NFVO**
  - **Pure SOL005 which can be used for integration with OSS/BSS**
- Use Case
  - **Case 1) ONAP Component wants to access an External NFVO for LCM operation (sub domain)**
  - **Case 2) ONAP Component wants to work with a component internal/external via SOL005 API**
- Operation
  - **Case 1: API Fabric takes care of transforming the simplified internal API to corresponding API calls to external NFVO APIs**
  - **Case 2 : API Fabric receives SOL005 API calls and enriches/transforms the API with internal/external API call**

Scenario 2 : Dynamic Routing and Request/Response Transformation for SOL003 API

- API Fabric Exposing two types of interfaces
  - **Simplified internal API which hides SOL003/Vendor complexity**
  - **Pure SOL003 (without VNFM specific extensions)**
- Use Case
  - **Case 1) ONAP Component wants to use Simplified API for VNF instantiation**
  - **Case 2) ONAP Component supports pure SOL003 API but not aware of vendor extensions**
- Operation
  - **Case 1: API Fabric takes care of transforming simplified internal API to corresponding multiple API calls - SOL003 specific or Vendor specific APIs**
  - **Case 2: API Fabric receives pure SOL003 request and enriches the request with vendor specific SOL003 extended parameters**

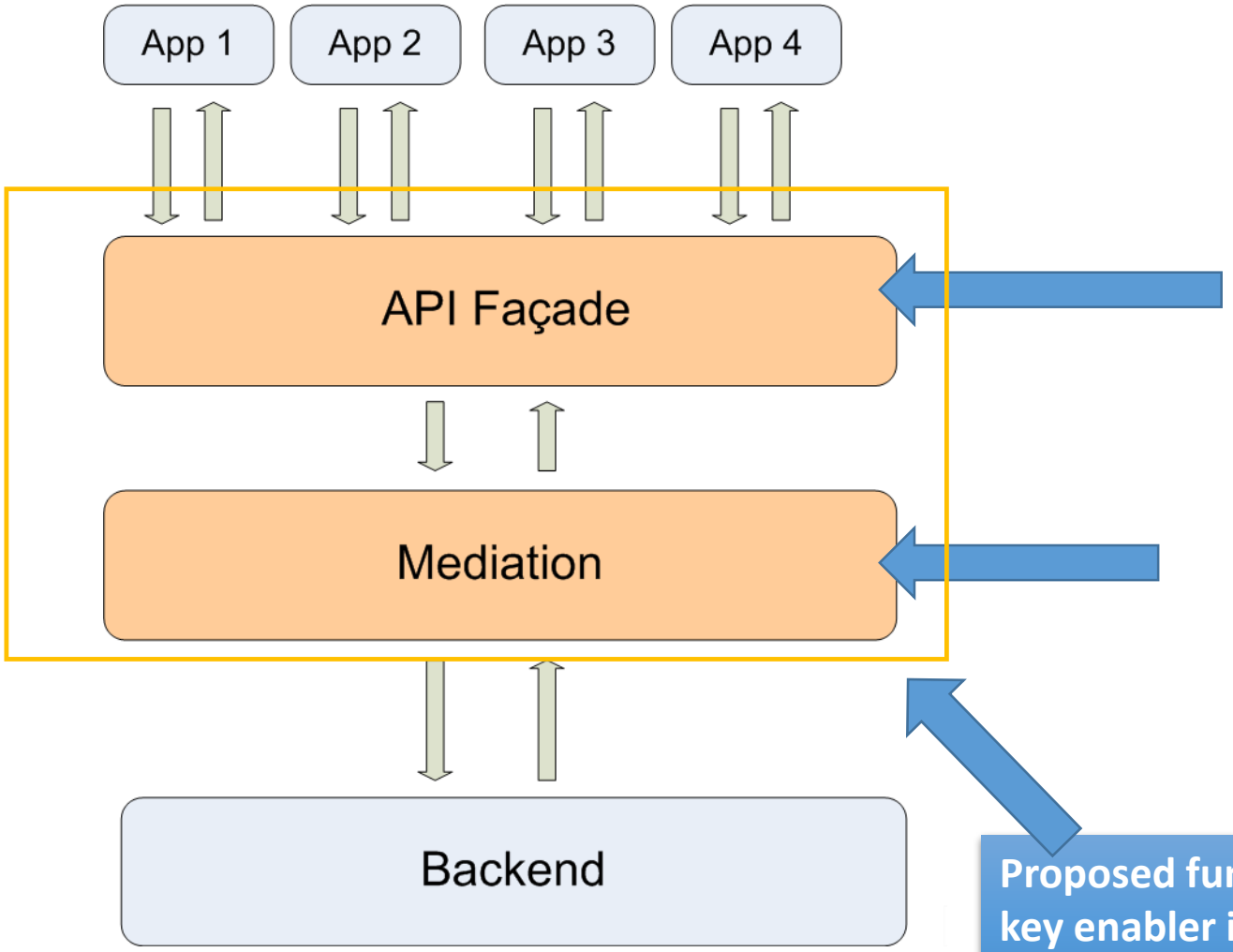
Proposed Use Cases (Any one to start with) 2/2

Scenario 3:

Security Enablement for External API supported TMF APIs

- Support Oauth 2.0 based Authentication/ Authrorization Support Token based authentication – with API Fabric acting as a mediator or Authorization Server
  - Short Term : Support Auth Grant based on – Implicit ot Client Credentials
  - Long Term : Support Auth Code, PKCE, Token based Authentication, Open ID based Authentication
- Support integration with AAF as Auth Provider
- Control the API calls using Scope
- Support Role based Access Control
- Enable HTTPS based secure channel between OSS/BSS and ONAP Ext-API

How the proposal differs from existing ONAP Functions ?



MSB

- MSB API GW has limited capabilities to support API Façade
- MSB Primarily supports Routing and Discovery. Tool set for API Composition and Management missing

Ext-API

- External API limited Mediation and Façade capability
- Purpose built for TMF without much reuse for other API facades
- No gateway, Transformation, Composition toolsets.

Proposed function combines both the functional capabilities and key enabler is a set of toolsets for reusing the same for multiple use cases, not limiting to MSB or Ext-API

Thank You