ONAP

OPEN NETWORK AUTOMATION PLATFORM

Service Orchestration- Need for Users

Background

Issues:

- Increased Capex and Opex
- Huge manual efforts involved
- single-purpose/ domain specific / homogenous networks

Possible way out:

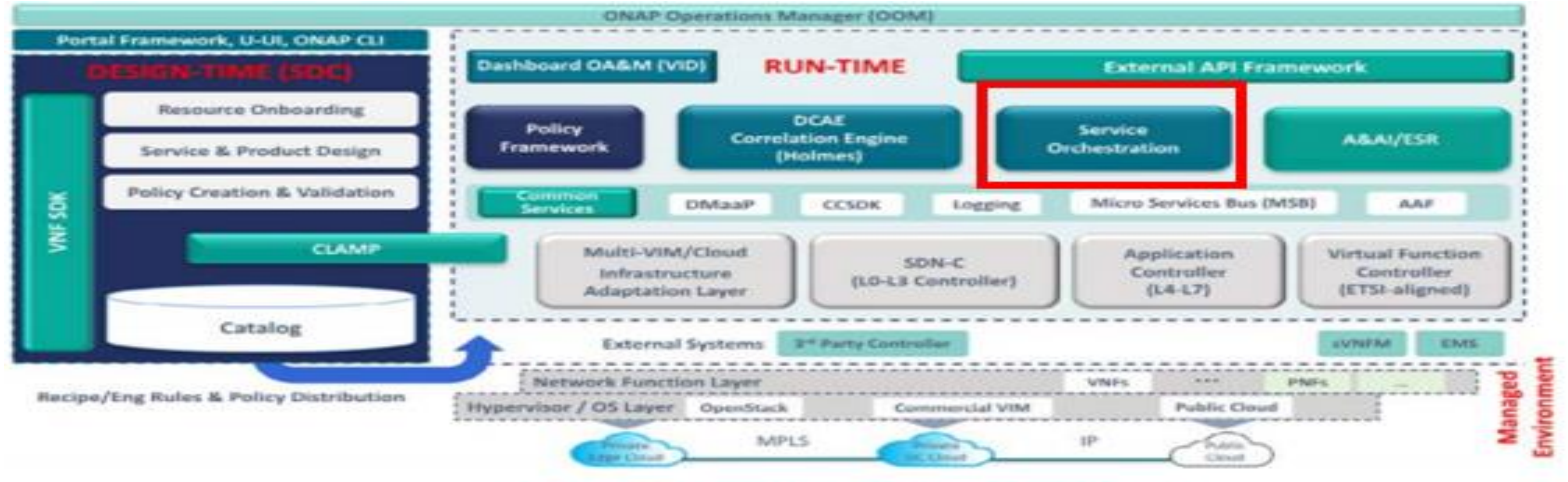
- SDN
- NFV
- Automation
- Orchestration

Automation vs Orchestration

- Automation is a way to eliminate the manual effort/human intervention involved in activities and bring in an ability to perform the same repeatedly, consistently and with better efficiency.

- Orchestration is executing the automation/manual ability of various modules in harmony as a consolidated process or workflow to accomplish the desired tasks.
  - Example : Managing lifecyle of a E2E service

ONAP – Service Orchestrator



ONAP Platform components (Amsterdam Release)

The Service Orchestrator (SO) component of ONAP provides orchestration at a very high level, with an end to end view of the infrastructure, network, and applications.

SO Key Interactions with ONAP components

SDC

- Distribution of orchestration artifacts (service & resource recipes and templates)
- UEB event notifications, HTTP artifact retrieval

AAI

- Query and update inventory
- RESTful API

Cloud (Platform Orchestrator)

- Instantiation of virtual resources in the cloud
- Openstack APIs (primarily Heat and Keystone)

SDN Controller

- Assign and configure network resources
- Yang-based RPC and REST

VFC Controller

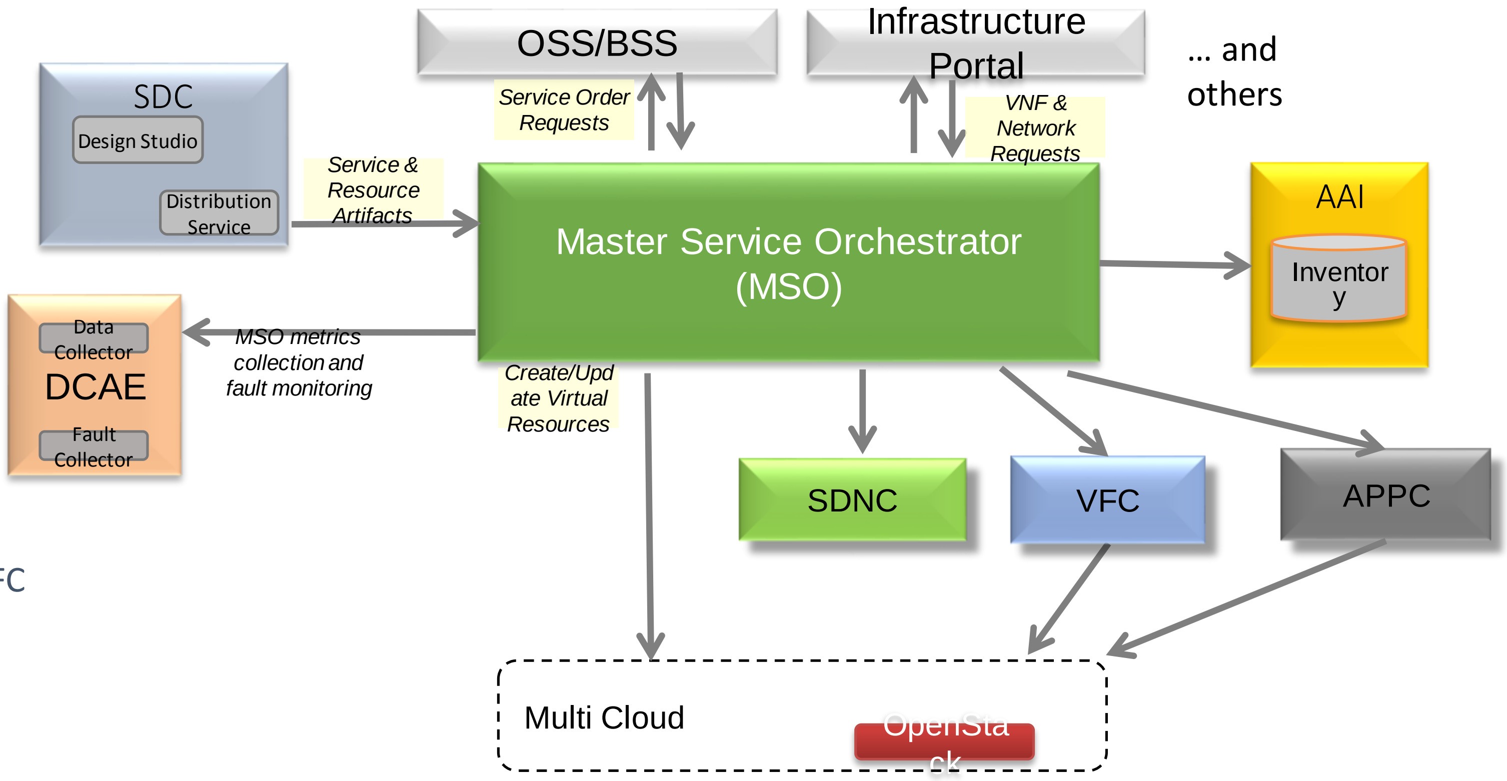
- Able to delegate the Network service details to VFC
- Rest Based interaction over MSB

App Controller (pending)

- Assign and configure application resources
- Yang and/or event based API

Policy

- Able to execute the policy recipe for a given policy ID



SO Software Architecture

API Handler

RESTful interface to northbound clients, could be OSS/ BSS, external API, portal,…

Handle service-level and infrastructure (VNF & network) requests

Use SO Catalog to map input requests to recipes (BPEL flows)

Track open and completed requests via SO Request DB

BPEL Execution Engine

Execute BPMN service recipes

Sequence orchestration steps by invoking Adapters for each Resource in the recipe

request and configure network resources via SDN-C

manage cloud resources via Mukti cloud(OpenStack)

configure Application VNFs via APP-C

Configure Network services viia VFC

update inventory via AAI

Perform additional orchestration steps (consult policy, etc.) per individual recipes

Perform error handling/rollback

Controller Adapters

Provide interfaces to lower level controllers and other ONAO components

Platform Orchestrator, SDN-Controller, APP Controller

Hides the details of complex interfaces (e.g. OpenStack APIs) via higher-level calls

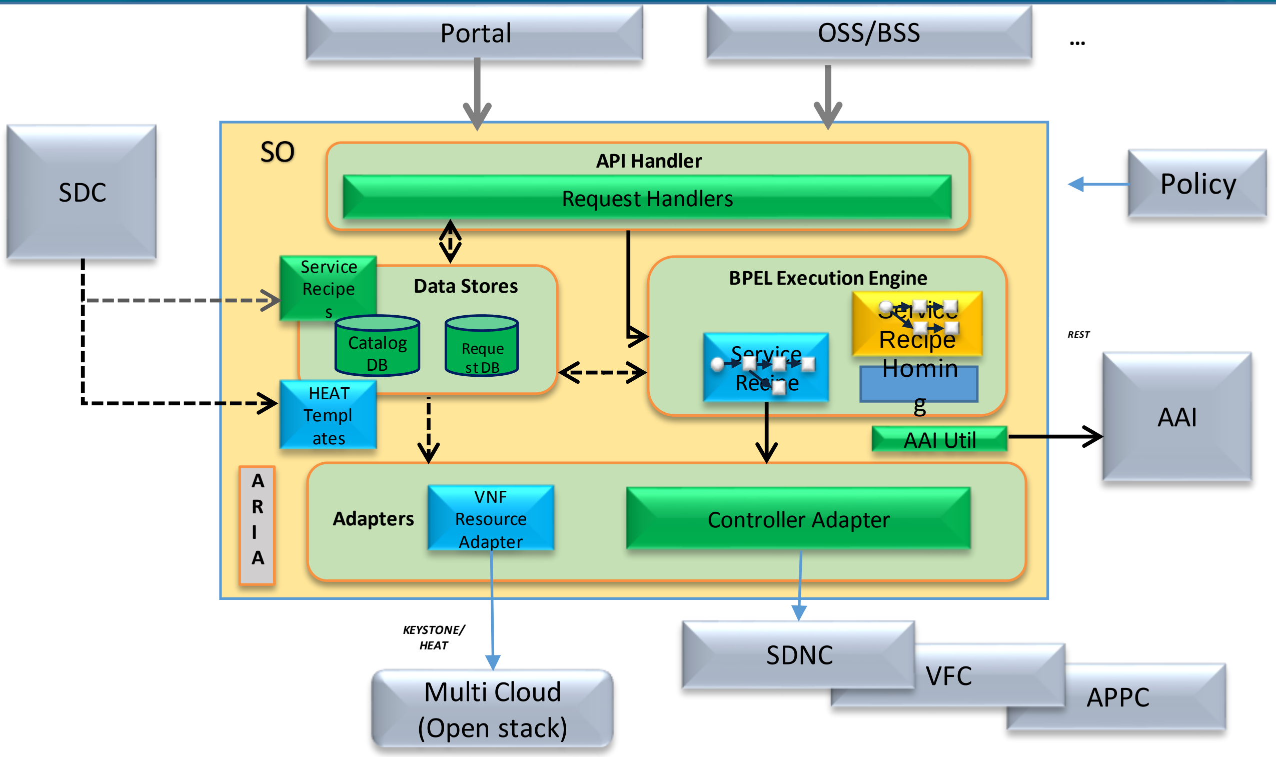
Expose interfaces to BPEL flows as SOAP or REST APIs (synchronous/asynchronous)

Use SO Catalog to map resource requests to a recipe/template

VNF > Heat templates

SDN Resource > Yang models

Merge input parameters with templates at run-time



Data Stores

Request DB

Tracks open and completed requests

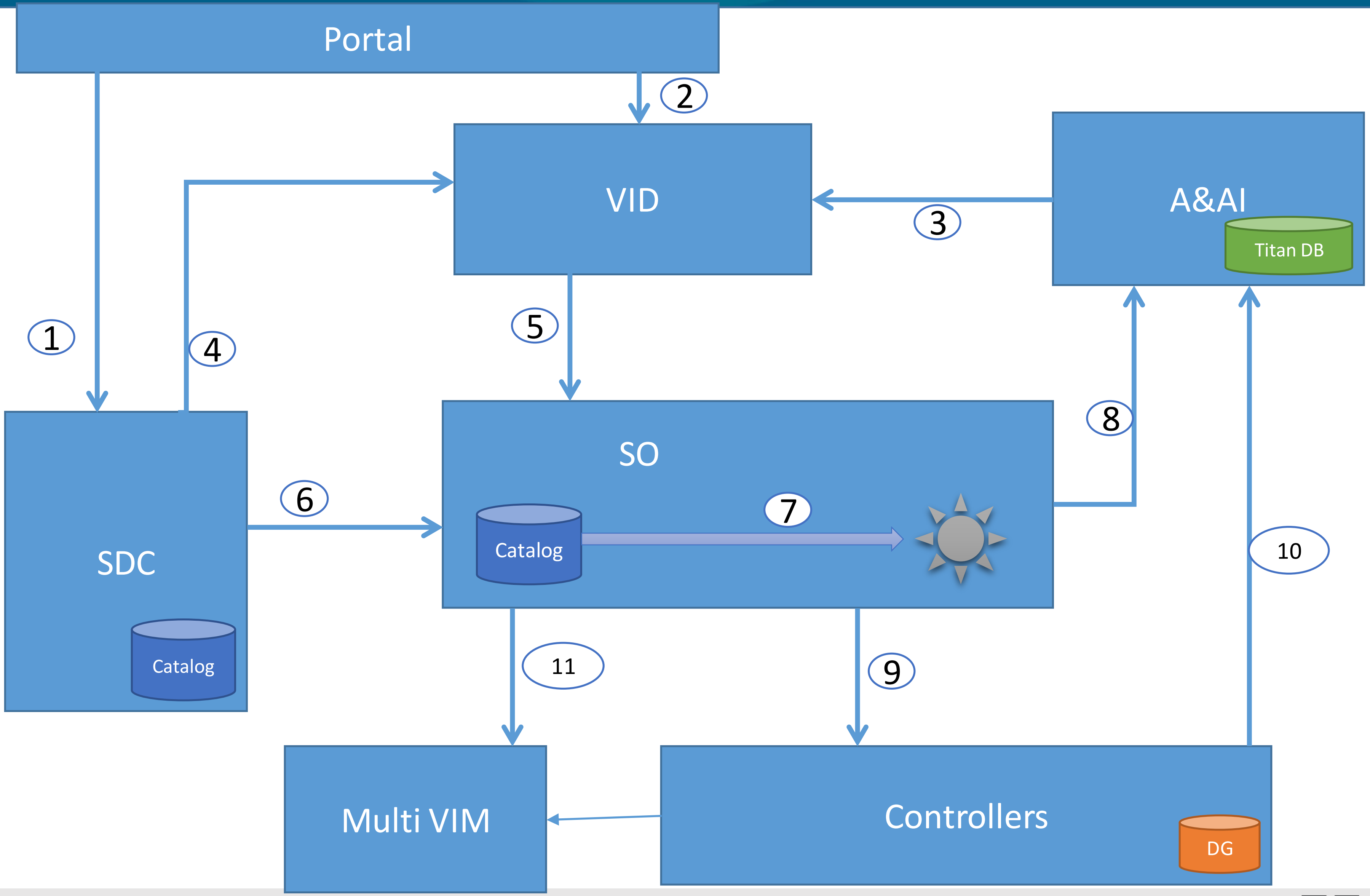
SO Catalog

SO view of the SDC Catalog

Service and resource recipes, templates, and definitions

Populated via SDC distribution service

Service Instantiation Flow



How to SO

API Handler

1. All the NBIs reside here,
Key classes for SO:
ServiceInstaces.java
E2EServiceInstaces.java
MSORequest.java

2. Update the request information in the Request db, Read the catalog infor synced from SDC and make the request for the workflow

3.
•The original request received by the API handler from the portal or other client.
•Metadata such as the request-id generated by the API Handler for the request.
•The name of the BPMN process to execute (obtained by the API Handler from the mso_catalog.service_recipe table.

BPMN main process flow

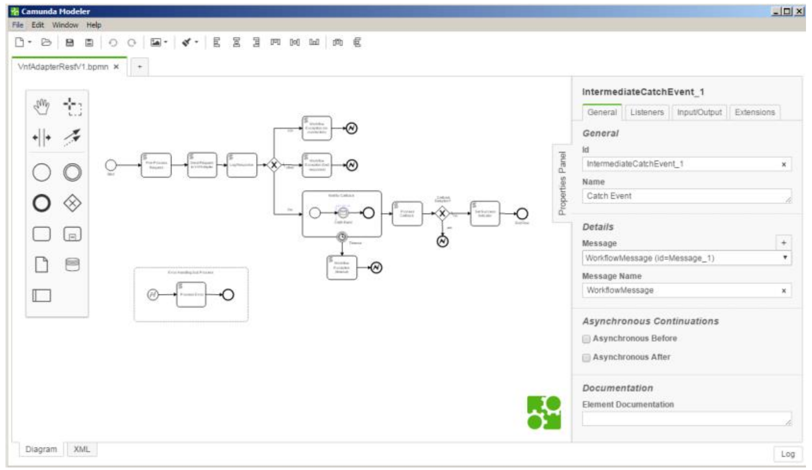
The BPMN application (war) exposes a REST endpoint to which the API Handler(s) send requests for flow execution. The message sent by the API Handler to this endpoint is a JSON wrapper. All main process flows implement an asynchronous service model. The connection to the API Handler is kept open until the main process flow sends back a response.

BPMN subprocess flow

The subprocess would be delegated with the specifics of the tasks defined in the main flow.

Groovy scripts

The business logics of the tasks are defined in the groovy,i(deally these can be java plugin too) that would call the adapters and share the request and get response for the defined artifacts.



General
Id: IntermediateCatchEvent_1
Name: Catch Event

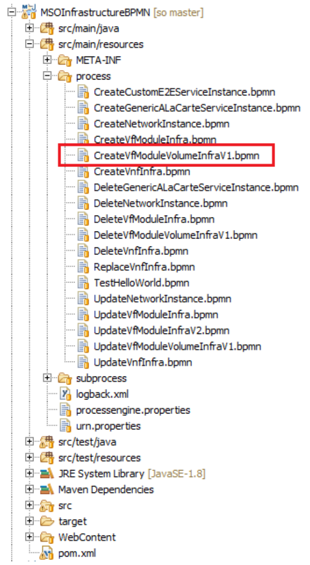
Details
Message: WorkflowMessage (id=Message_1)
Message Name: WorkflowMessage

Asynchronous Continuations
Asynchronous Before
Asynchronous After

Documentation
Element Documentation

Adapters

Adapters are things that would be integrating the other ONAP components south bound to the SO and takeup the actions further.



```
MSOInfrastructureBPMN [so master]
  src/main/java
  src/main/resources
    META-INF
    process
      CreateCustomE2EServiceInstance.bpmn
      CreateGenericALaCarteServiceInstance.bpmn
      CreateNetworkInstance.bpmn
      CreateVfModuleInfra.bpmn
      CreateVfModuleVolumeInfraV1.bpmn
      CreateVnfInfra.bpmn
      DeleteGenericALaCarteServiceInstance.bpmn
      DeleteNetworkInstance.bpmn
      DeleteVfModuleInfra.bpmn
      DeleteVfModuleVolumeInfraV1.bpmn
      DeleteVnfInfra.bpmn
      ReplaceVnfInfra.bpmn
      TestHelloWorld.bpmn
      UpdateNetworkInstance.bpmn
      UpdateVfModuleInfra.bpmn
      UpdateVfModuleInfraV2.bpmn
      UpdateVfModuleVolumeInfraV1.bpmn
      UpdateVnfInfra.bpmn
    subprocess
    logback.xml
    processengine.properties
    urn.properties
  src/test/java
  src/test/resources
  JRE System Library [JavaSE-1.8]
  Maven Dependencies
  src
  target
  WebContent
  pom.xml
```

OPEN NETWORK AUTOMATION PLATFORM

SO –R2 enhancements proposal

SO Functional Evolution and Road Map

- R0 – Support for specific resource orchestration, NB system is required to call multiple SO API for orchestrating E2E service
- In R1 SO
  - Homing solution brought - demonstrated the placement of VCPE VNFs
  - Declarative TOSCA orchestrator integrated to the SO code as a DO, yet to be functionally attached.
  - introduced new E2E service instance NB API, yet it is currently calling a specific BPMN recipe for orchestrating VOLTE service which orchestrate VFC and SDNC
- In R2+ on top of existing support (Requires Brianstorming)
  - S3p Improvement and make carrier grade
  - E2E service should evolve to provide ability to orchestrate any service modeled in SDC in model driven manner
  - APIs should be generic (based on SDO) and based on the model
  - Allow service composition , i.e. high level service depends on lower level service in a recursive manner
  - Each orchestration flow can pre define the interaction of SO adapters as a design time decision

SO Carrier-Grade Mission – Proposal

Proposal

- To make SO Carrier-Grade, non-functional requirements must be fulfilled. ONAP defined the following non-functional requirements.
  - Scalability (Scale in and out)
  - Stability (managing steady load for required time period)
  - Resiliency (Failover, HA)
  - Security (Secure Communication, AAA, Security Logging/Auditing)
  - Performance (Response Time, Transaction/message rate, Latency, Footprint)
  - Manageability (SSO, Logging/Tracing, Monitoring)
  - Usability (conform to ONAP-level Usability)
- To achieve the above, SO needs platform-level enhancements.
  - SO process monitoring is another important factor from a Carrier-Grade perspective.
    - The monitoring provides manageability (and could be usability).
  - We need to avoid vendor lock-in (e.g., adding commercial products into ONAP).

Open Issues

ONAP Carrier-Grade requirements are not finalized. That is the dependency.

How will OOM, CHAP projects impact ONAP components?

Need to decide SO scalability strategy: scale at the SO component level or at the SO sub-component level.

Need to define the Container Manager functionality and realization.

Need to define the Security Framework functionality and realization.

Dependencies :

OOM / Kubernetes

CHAP (Common HA Platform)

AAF

KMS (Key management system- Security)

SO R2+ Extensibility requirements

- Ability to add new recipe externally to SO upstream
  - Ability to add new BPMN flows + their associated groovy/Java externally to SO
- Ability to add new resource adaptation externally to SO
  - Ability to adapt the third party controllers (not managed by ONAP) to be able to associate with the SO
- Ability of bringing in the (Declarative) TOSCA as a DO inside SO and able to delegate the orchestration request to that block.
  - Define Tosca Model / types and distribute to the SO at runtime.

What can be done inside SO

Functionality	Limitations/Observations
APPC adapter in SO is not yet used	a. Integrated APPC client in SO
Improve Troubleshooting of the SO Code	a. Better debugging of the java and BPMN code
	b. Enhance the monitoring capability of the workflow process
	c. Strengthen the UT cases of the flows
Better Packaging of the application code	a. Bring in the container , dockers/ Spring boot
	b. Split the packages further (eg MSOCommon BMPN)into sub sects and create modules to as to handle them better.
Restructure of the MSO API handler	a. Bringing the hierarchy and abstraction based on the functionality
	b. bringing plugins for the specific functionality
Replace the adapters with the java plugins	a. Replace the groovy scripts tasks with the Java service tasks
	b. New code of the BPMN to adopt the Java server tasks to have better control