Change Management Extensions
Frankfurt and Beyond

Łukasz Rajewski (Orange)
Zu Qiang (Ericsson)
Ajay Mahimkar (AT&T)

14.01.2020

Change Management Vision

Zero touch deployment, zero service impact

Design:

- Fast
- Easy
- Flexible
- Complete

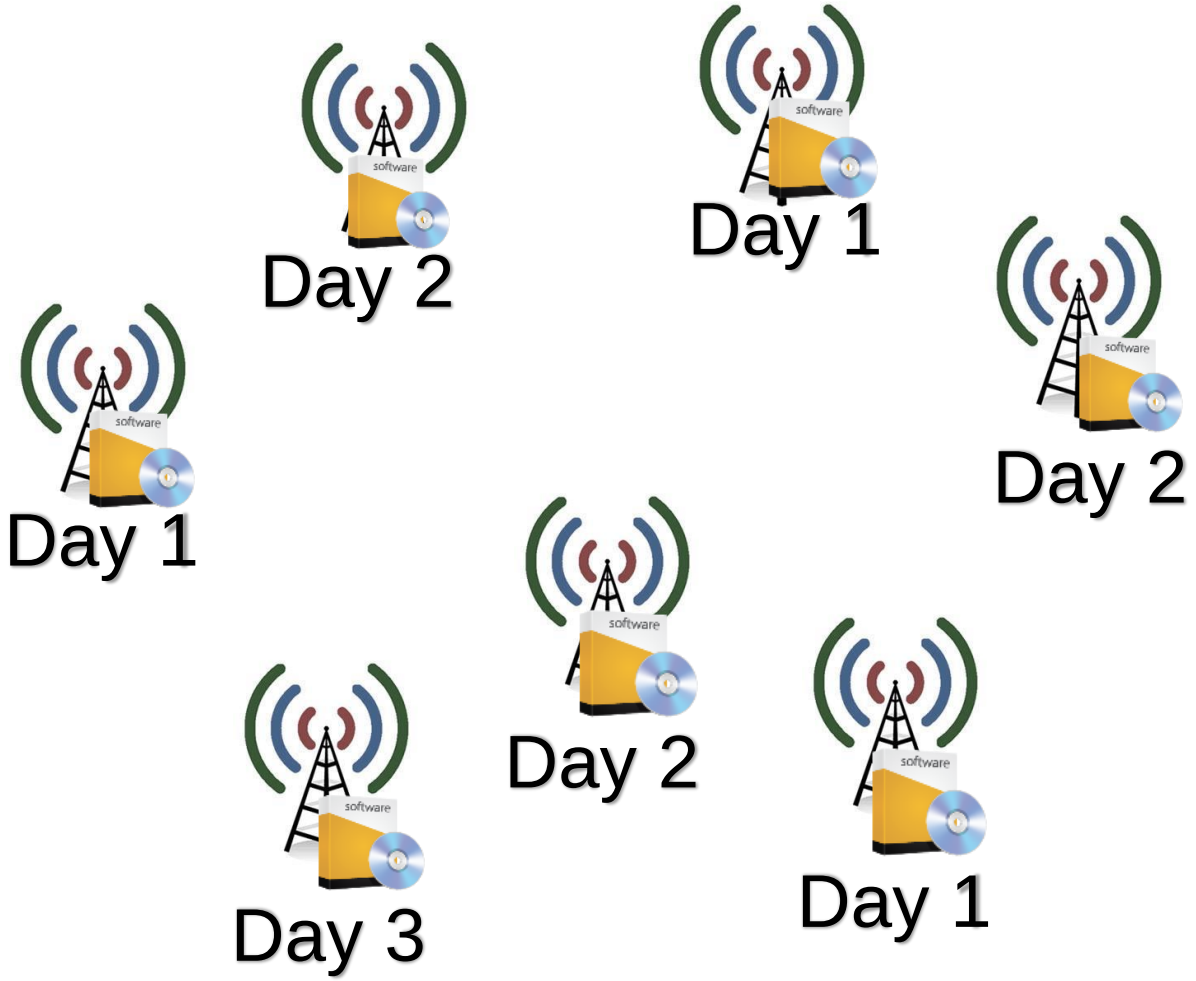
Scheduling:

- Automated
- Intelligent
- Cognitive

Execution:

- Automated
- Safe
- E2E

Rolling changes for network-wide deployment

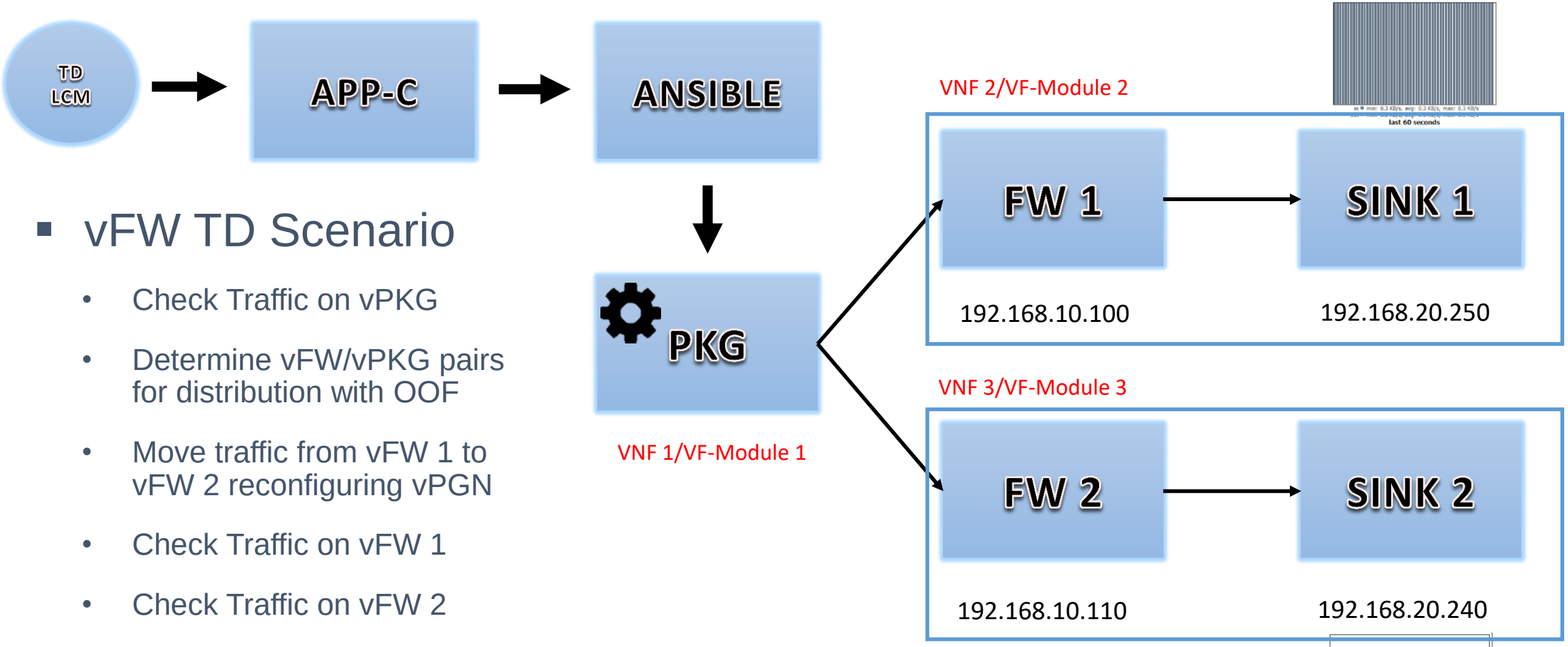


1. Minimize impact during deployment
   - Changes may require interruption to service
   - Leverage redundancy
2. Minimize impact post deployment
   - Unexpected performance
   - Damage control through early halt

Reduced service downtime

- Deployment optimization
- Appropriate scheduling

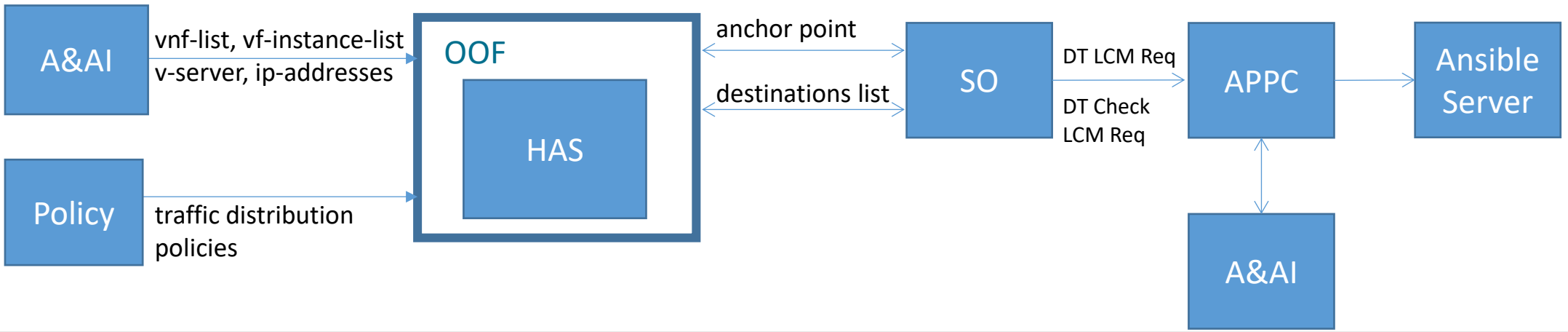
vFW Traffic Distribution UC (El Alto)



- vFW TD Scenario
  - Check Traffic on vPKG
  - Determine vFW/vPKG pairs for distribution with OOF
  - Move traffic from vFW 1 to vFW 2 reconfiguring vPGN
  - Check Traffic on vFW 1
  - Check Traffic on vFW 2

vFW Traffic Distribution Workflow (El Alto)

- Algorithm for Traffic Distribution Workflow
  - Traffic Distribution Optimization algorithm aims to deliver extra information for DistributeTraffic LCM in APPC
  - Anchor Point – Firstly VF-module instance that will perform an operation of Traffic Distribution. In the future other methods
  - Destination Point - VF-module instance(s) that will take a traffic
  - A&AI delivers an information about existing VNF instances, VF-Modules instances, V-Servers etc.
  - Policy delivers an information about traffic distribution policies
  - Algorithm delivers an anchor point and list of destinations for distribution



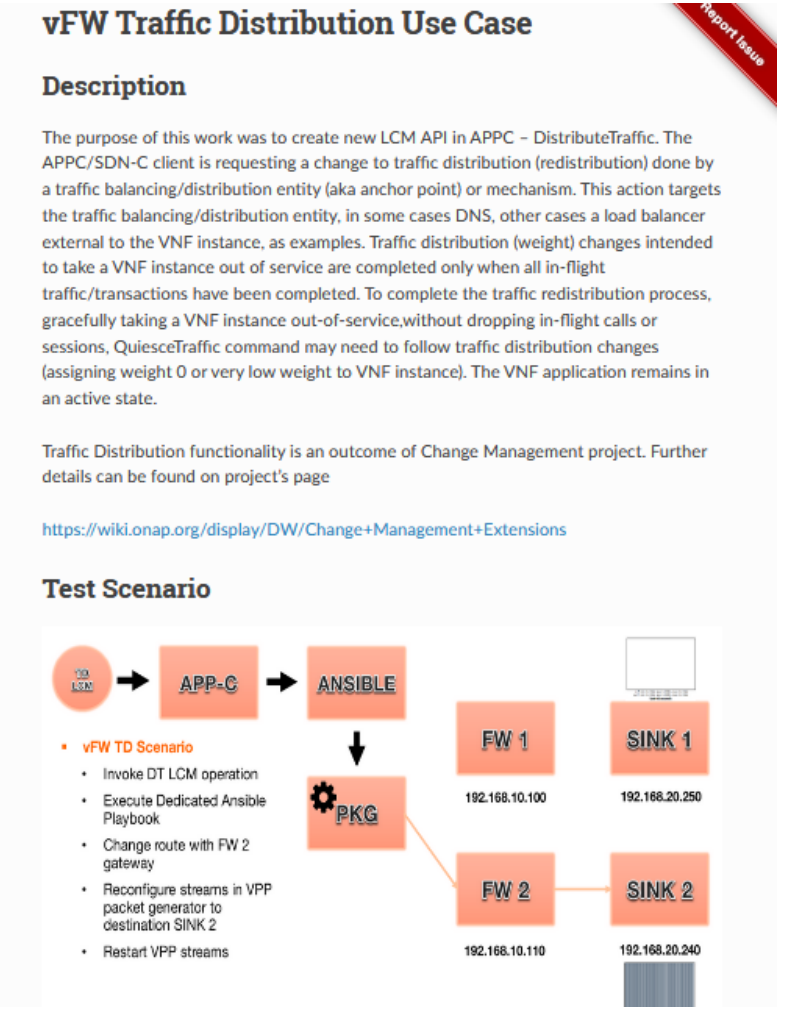
Dynamic selection of vf-module instance with OOF (El Alto)

1. OOF / HAS algorithm used
   a) Selection of vf-module Instance for service/vnf
   b) Selection is policy based
      - Filtering by AAI attributes
      - Relation i.e. by region
      - Exclusion/Inclusion
   c) Exclusion/Inclusion
2. OOF selects vf-module candidates for requested demands
   a) We can find vf-module that satisfies specified criteria
   b) We can find related vf-modules
3. Usefull in workflows when we do not know the exact vf-module to perform action on


```
"placementInfo": {
    "requestParameters": {
        "chosenRegion": "RegionOne",
        "chosenCustomerId": "DemoCust_4fb7d3cf-5ddc-4d8c-8acf-70cc9174d18f"
    },
    "subscriberInfo": {
        "globalSubscriberId": "dbc2c763-6383-42d6-880a-b7d5c5bc84d9",
        "subscriberName": "oof-so-chm"
    },
    "placementDemands": [
        {
            "resourceModuleName": "vFWDTvFW",
            "serviceResourceId": "vFWDTvFW",
            "resourceModelInfo": {
                "modelInvariantId": "6f3fd439-fd5f-4a2d-95bc-b6bf8787001a",
                "modelVersionId": "202d2fd8-a045-4c9a-b767-2a1639c10291"
            },
            "excludedCandidates": [{
                    "identifierType": "vfmodule",
                    "identifiers": []}],
            "requiredCandidates": [{
                    "identifierType": "vfmodule",
                    "identifiers": []}]
        },
        {
            "resourceModuleName": "vFWDTvPGN",
            "serviceResourceId": "vFWDTvPGN",
            "unique": "false",
            "resourceModelInfo": {
                "modelInvariantId": "3f356335-7b36-41ee-8f74-72d0a2ec3ebf",
                "modelVersionId": "6bfe954e-bb00-4111-be3c-33eed9d20a8c"
            }
        }
    ]
},
"serviceInfo": {
    "serviceInstanceId": "2ad369d4-9056-4dc9-8e6d-df24f45e8729",
    "serviceName": "vFW_TD",
    "modelInfo": {
        "modelInvariantId": "TD-invariantId",
        "modelVersionId": "TD-versionId"
    }
}
}
```

vFW In-Place Upgrade & Traffic Distribution Workflow (Frankfurt)

- Workflow and use case modified towards In-Place Upgrade with Traffic Distribution
  - Upgrade (Pre/Post Check, Upgrade), Lock (Check, Lock, Unlock) and Traffic Distribution LCMs
  - For Guilin planned further modifications
- Improvements in APPC for VNFC scope reconfiguration with Ansible
  - APPC is able to auto generate NodeList section for vnf, vf-module or vnfc scope
  - Requires oam Ips configured in AAI (vnf and vnfc-lvl)
  - NodeList is generated from payload/request-parameters section
  - Will evolve into generic vnfc and vf-module identifiers support for APPC LCM actions for any kind of protocol



vFW Traffic Distribution Use Case

Report Issue

Description

The purpose of this work was to create new LCM API in APPC – DistributeTraffic. The APPC/SDN-C client is requesting a change to traffic distribution (redistribution) done by a traffic balancing/distribution entity (aka anchor point) or mechanism. This action targets the traffic balancing/distribution entity, in some cases DNS, other cases a load balancer external to the VNF instance, as examples. Traffic distribution (weight) changes intended to take a VNF instance out of service are completed only when all in-flight traffic/transactions have been completed. To complete the traffic redistribution process, gracefully taking a VNF instance out-of-service,without dropping in-flight calls or sessions, QuiesceTraffic command may need to follow traffic distribution changes (assigning weight 0 or very low weight to VNF instance). The VNF application remains in an active state.

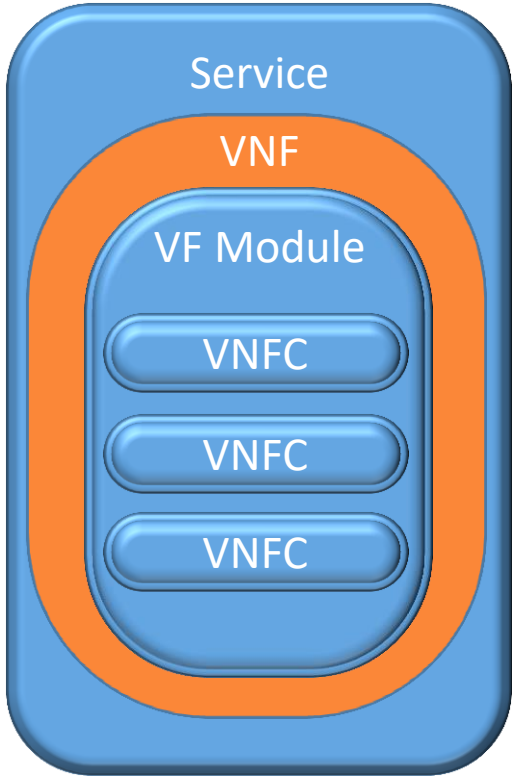
Traffic Distribution functionality is an outcome of Change Management project. Further details can be found on project's page

https://wiki.onap.org/display/DW/Change+Management+Extensions

Test Scenario

Current SO Building Blocks Supported Use Cases

SO Building Block implementation:
The SO building blocks are a set of database-driven, configurable and generic process steps to be leveraged through several actions defined as 'Macro' flows. For each of the macro flows, there are a set of actions to the orchestration services and various type of resources orchestrated by ONAP.



<u>Supported Services & Resource types</u>
These resource types are essentially the ones defined in the model - through the SDC framework. SO orchestrates service, vnf and vfModule building block for assign, create configure and activate.

There is a lack of vnfc orchestration in ONAP that is required in order to support complex lifecycle management for various vnf use case.

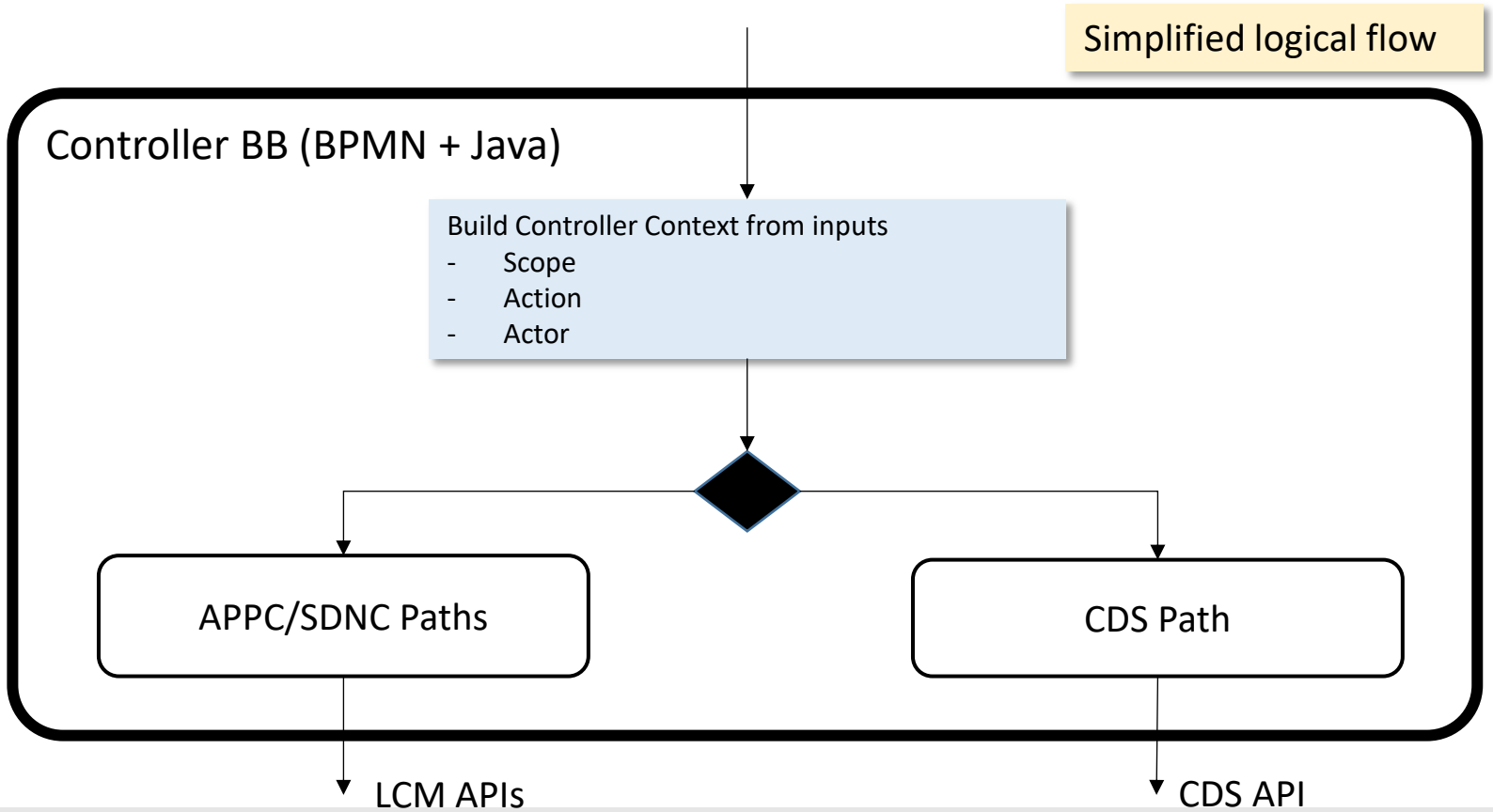
- ☑ Services
- ☑ VNF (Virtual Network Function)
- ☑ VF modules (i.e. a deployment unit, such as a HEAT stack)
- ☒ VNFC (virtual network function component)

SO MACRO Generic Building Block Orchestration (Frankfurt)

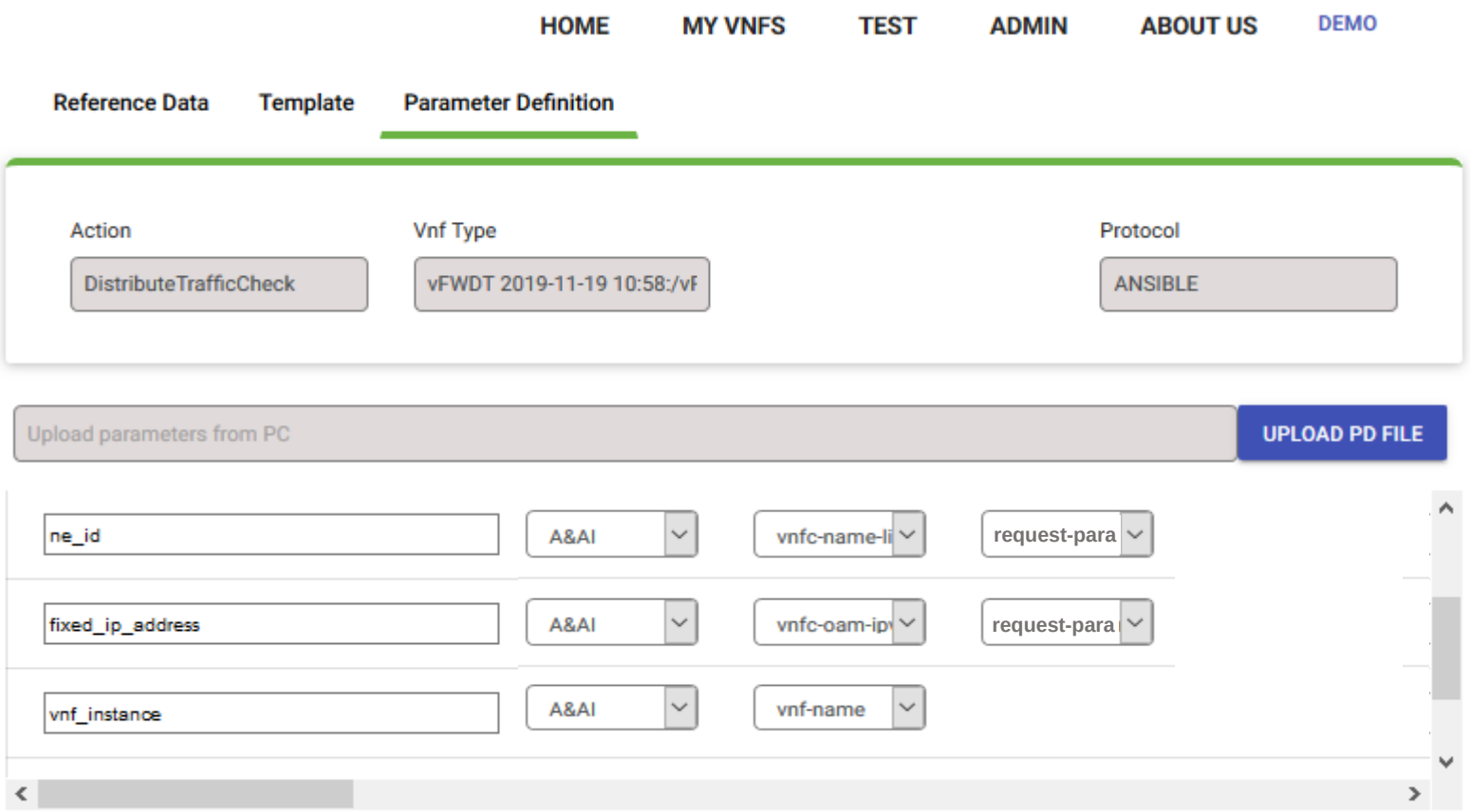
Id	Composite_Action	SEQ_NO	FLOW_NAME	FLOW_VERSION	LOOKUP_ID	SCOPE	ACTION
1	Service-Macro-Assign	1	ControllerExecutionBB	1	1	vnf	config-assign
2	Service-Macro-Assign	2	ControllerExecutionBB	1	1	vfModule	config-deploy

Scope
- service
- pnf
- vnf
- vfModule
- vnfc (to be added)

NOTE: Scope drives input action identifiers to controller LCM execution such as ssid, vnf-id, vf-module-id, vnfc-name, etc..



APPC/CDT Template Definition for VNFC (Plan)

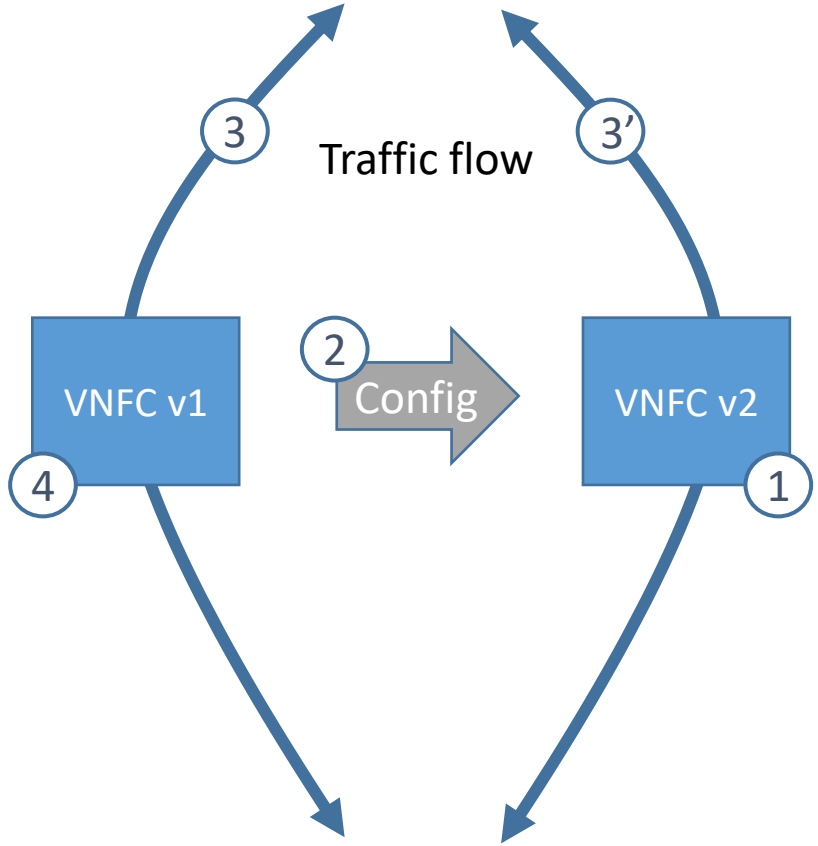


- Requires vnfc info in AAI
- ne_id – APPC resolves from AAI -> VNFC -> vnfc-name
- fixed_ip_address – APPC resolves from AAI -> VNFC -> vnfc-ipaddress-v4-oam-vip
- action-identifiers: vnf-id
- filtering by request-parameters: vf-module-id, vnfc-type.
- SO determines vf-module-id and vnfc-type. Most likely vnf-type can come from the input
- Can be applied for:
  - UC #1
  - UC #2
  - UC #3
  - UC #4

The best option that allows to configure concrete VM for vnf, vf-module and vnfc scope in APPC

vFW / vDNS Build & Replace Upgrade Workflow & UC (Plan)

- Possible change of the use case from vFW to vDNS
- Change of the Instantiation methods to Macro Flow with CDS
- Integration with vDNS ScaleOut/In use Case
  - Depends on the plans of Scaling team for implementation of ScaleIn scenario
  - Without ScaleIn only ScaleOut with new software version
- LCM actions execution through the CDS and APPC
  - Workflow defined in the Orchestration Flow Table in SO
  - Use of ControllerExecutionBB
  - Actions executed on VNFC level with Ansible protocol (but ready for NetConf)
  - Dynamic selection of vf-module instance with OOF
    - Integration with SO (probably in the ControllerExecutionBB)
    - MDSAL as a new source of data for filtering in HAS



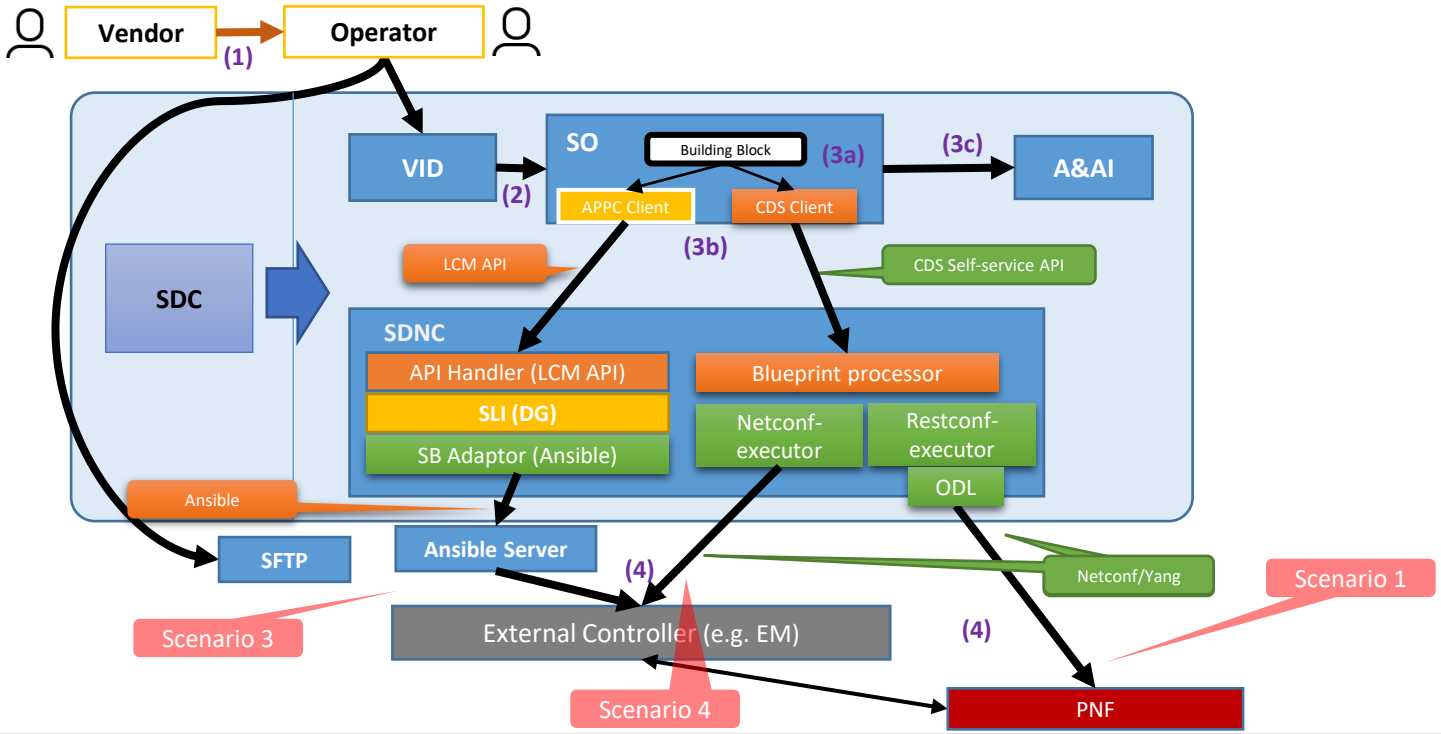
E2E PNF Software Upgrade (Frankfurt)

- PNF in-place software upgrade procedure on a single PNF instance without schema update
  - Scenario 1: Using direct Netconf/Yang interface with PNF
  - Scenario 3: Using Ansible protocol with EM
  - Scenario 4: Using Netconf/Yang interface with EM

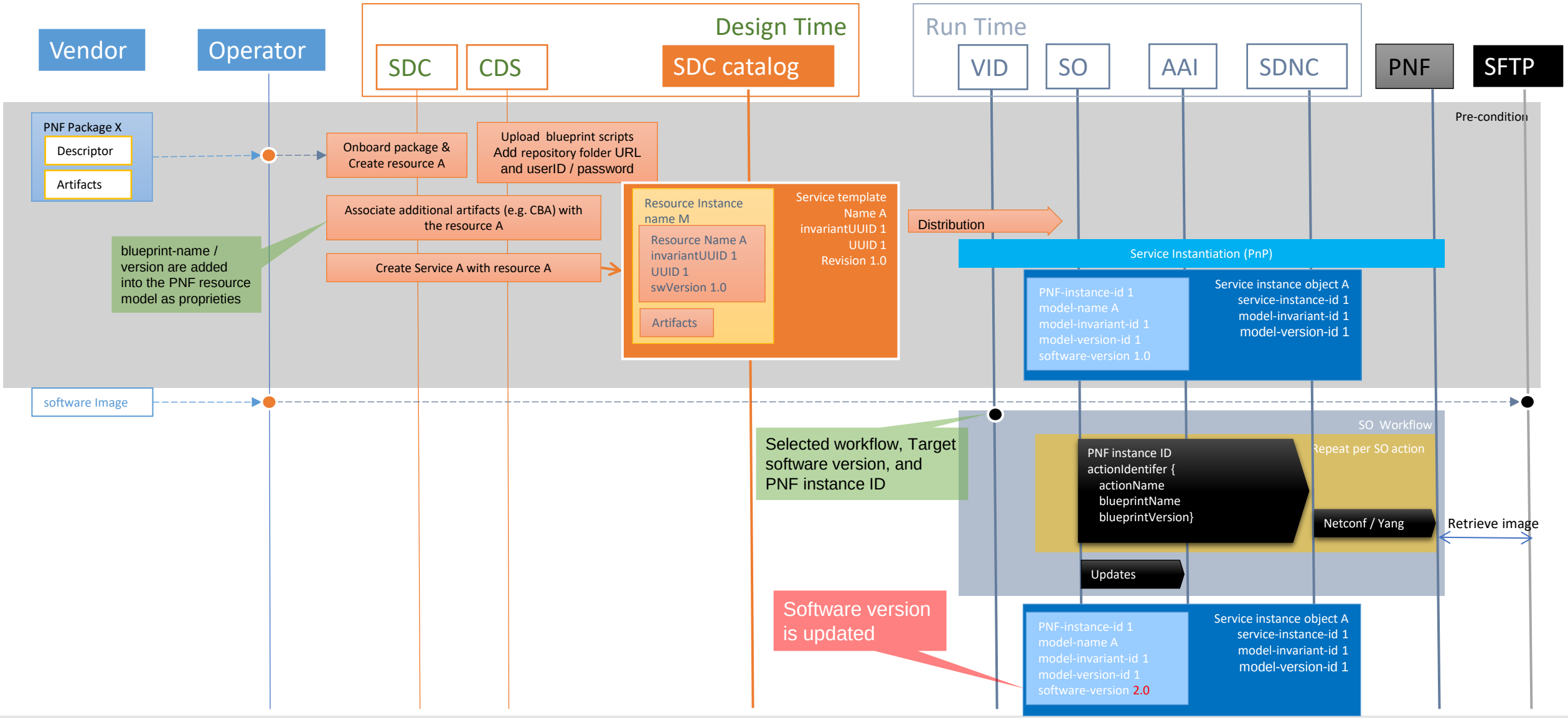
Only the 1st part of the E2E solution

- Service level schema update is proposed as "Enable service level LCM operations"
  - only "PNF software version onboarding" is committed

The 2nd part of the E2E solution



Update one PNF instance without schema update (Using direct NetConf/Yang interface with PNF - Frankfurt)

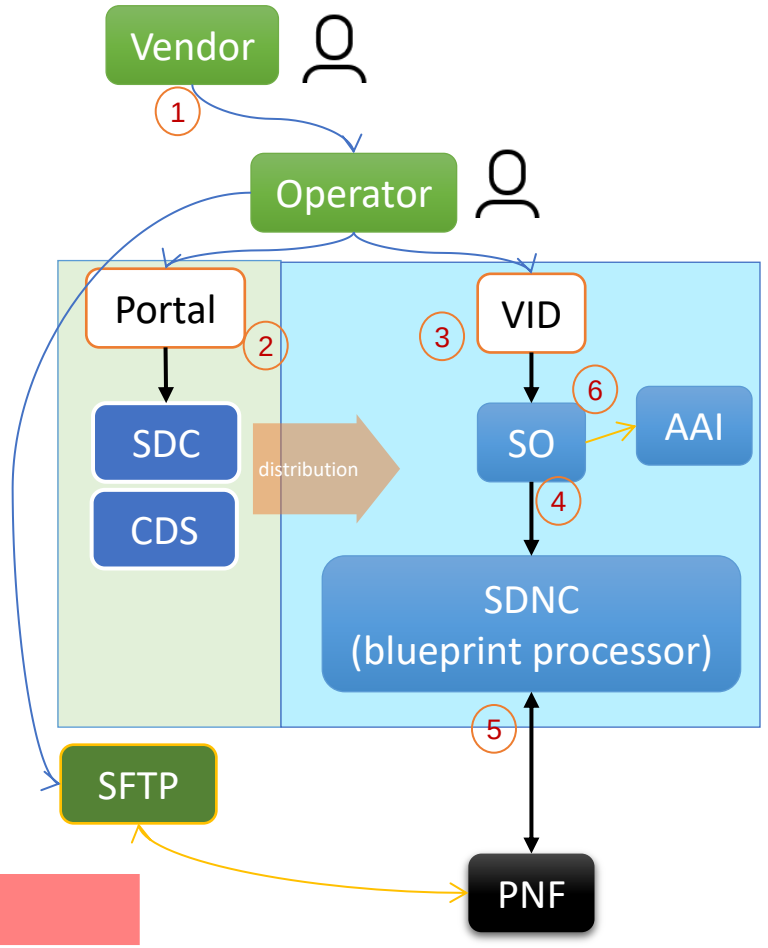


Upgrade with modification of the schema

- New VSP -> new/updated resource (PNF model or VF-Model)
  - Modification of Service Model required to provision new blueprints/artifacts
- Today we cannot move existing service instances to the new service model
  - We can only create new service instance for new service model
  - Changing model of existing service instance would not mean only changes in MariaDB and AAI
  - Dedicated procedures required for specific schema transitions (1.0 ->2.0 != 2.0->3.0)
- Schema update based on the Service Model
  - Build and replace Pattern to be used for CNFs and VNFs
    - For CNFs controlled by K8s, for VNFs by ONAP controller
  - Data migration and reconfiguration for VNF / PNF
  - Workflow needs to operate on cross-service scope
  - We need cross-service LCM operations
    - For CNFs not required since MultiCloud/K8s will provision that
  - OOF may help to coordinate the upgrade process

Service level Schema update procedure

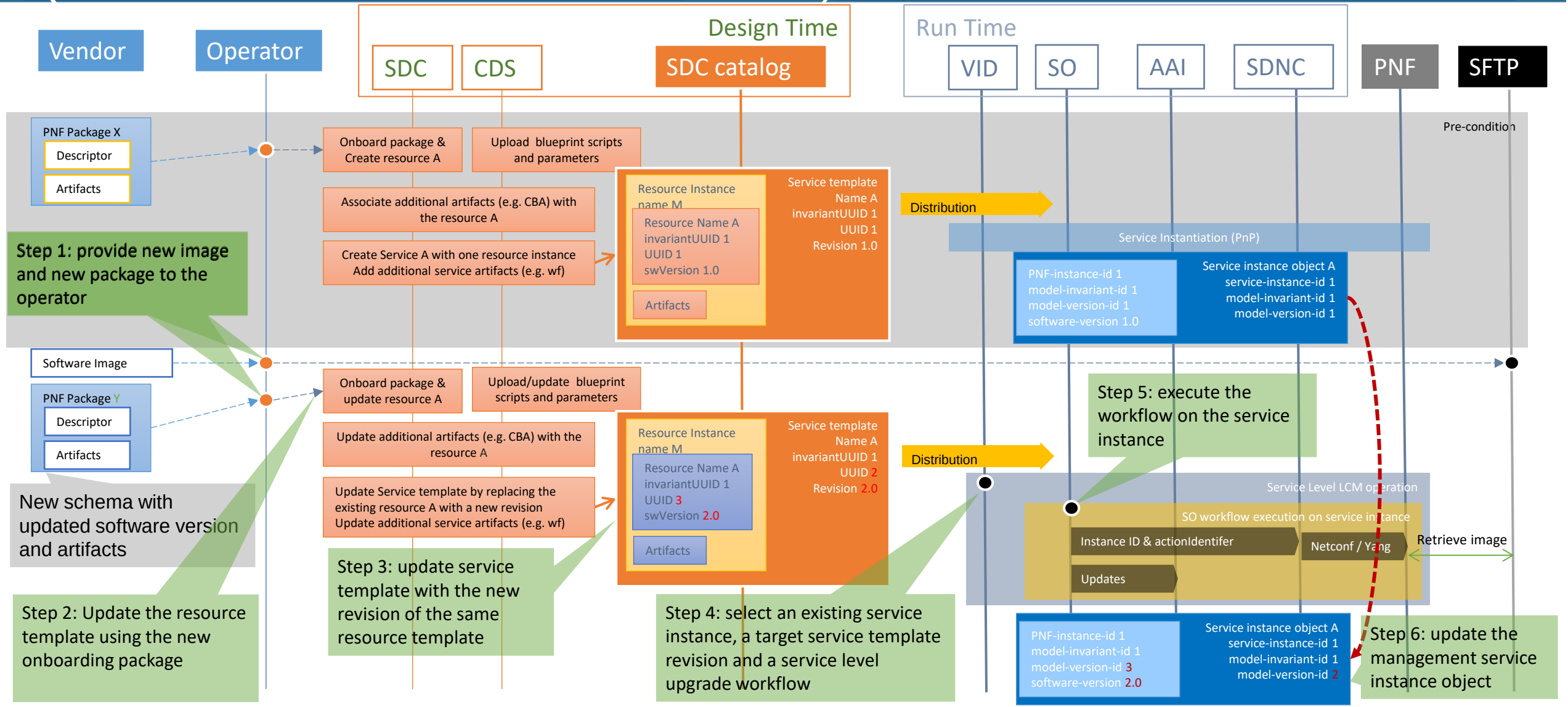
1. A vendor shall provide
   - a new VNF/PNF package with updated artifacts, and
   - the new VNF/ PNF software image to the operator.
2. At receiving of the new package, the operator shall
   - onboard the new package and create a new resource template or update the existing resource template
   - update the existing service template with the new or updated resource template
   - distribute the updated service template to run time.
3. At run time, the operator shall, based on the updated service template,
   - upgrade a service instance and its resource instances, and
   - update the run time catalog with the updated service/resource template



Schema upgrade must be handled at service level for both design time and run time

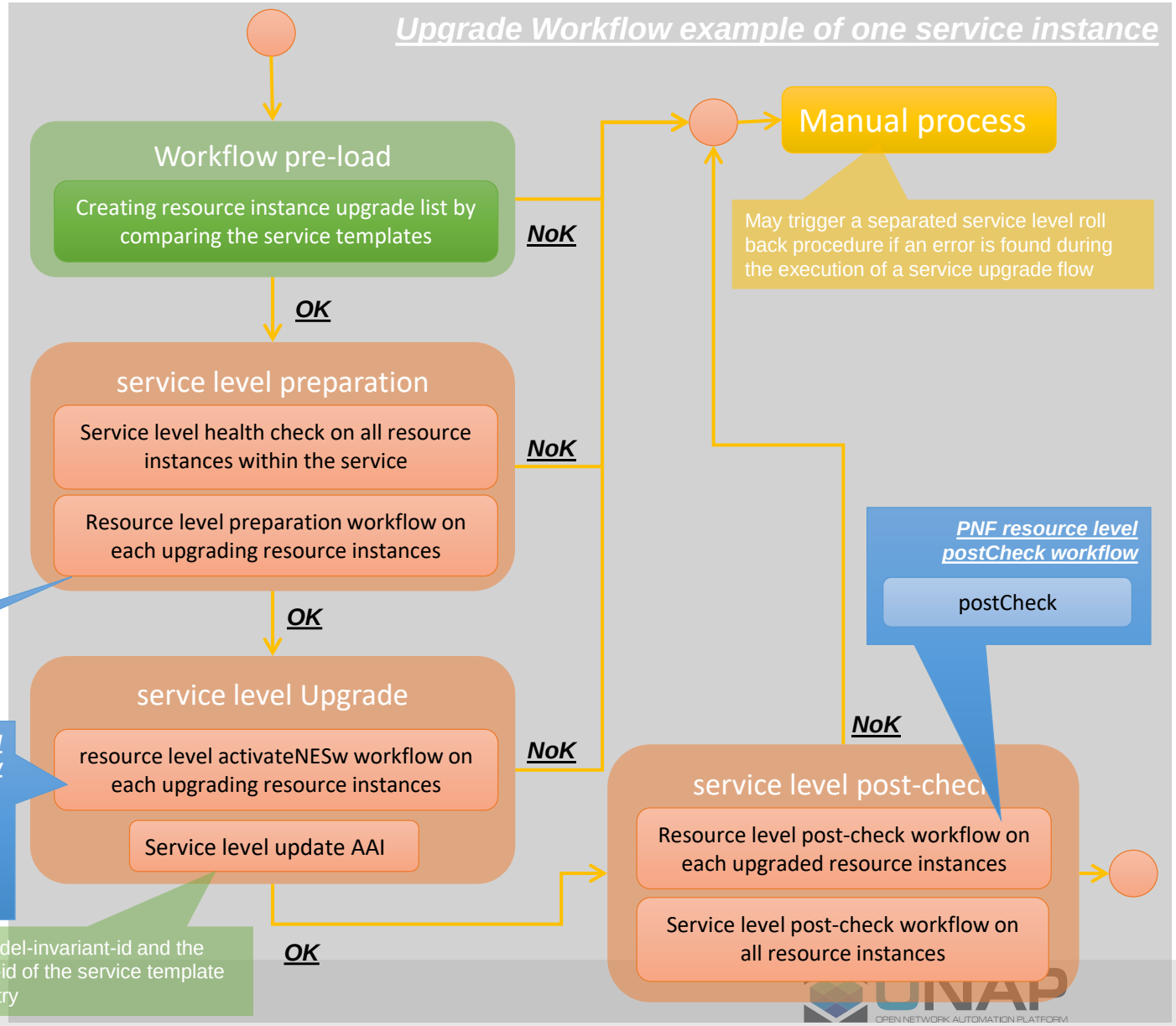
Then ONAP can make use of the new artifacts provided by the vendors, after schema update

Service level LCM operation on PNF example (with the same resource name - Guilin)



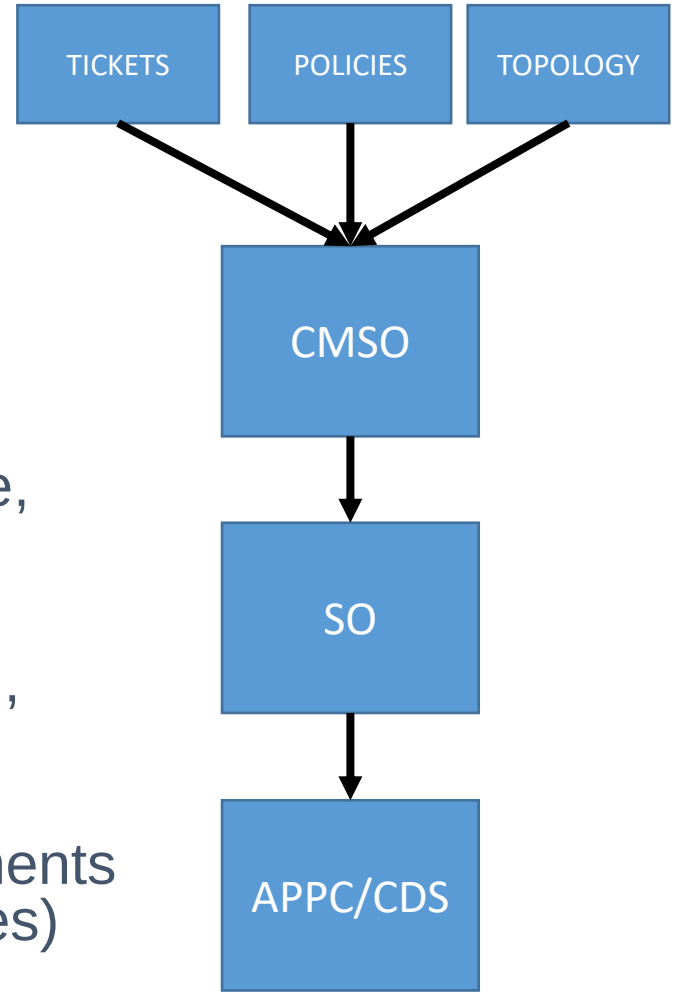
Service level workflow on PNF Example

- Design time
  - indicates which resource instance shall be upgraded
  - indicates which sub-workflow / building blocks shall be used at resource level
- Run time
  - After service level upgrade, all resource instances must be upgraded to the revision defined in the service template



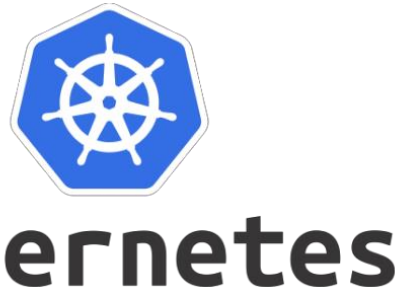
Change Management Scheduling in ONAP and extensions

- As of Frankfurt
  - Automated schedule discovery with conflict avoidance
  - OOF for schedule discovery/optimization algorithms
    - use of MiniZinc / model-driven approach
    - static composition of constraints
  - Constraints supported: conflict, concurrency
- Guilin and beyond
  - Support wide variety of constraints – service impact, timezone, grouping based on attributes
  - Policy-enabled discovery – store constraints in policy engine
  - Enable modular and dynamic composition of constraints – i.e., user specified selection of constraints for each schedule discovery
  - Scale up schedule discovery to large number of network elements (especially for network edge – hundreds of thousands of nodes)



Integration of CNFs as enabler for Change Management

- CNFs/k8s have clear advantage over VNFs
  - Faster Instantiation
  - Scaling enabled
  - Distribution of Traffic
  - Upgrading and Release Management
- MutliCloud-K8s Project
  - introduces CNF Deployment & Configuration Capabilities into ONAP
  - CNF/K8s Management Integrated with ONAP MultiCloud
  - Tested for deployment of vFW CNFs/VNFs with VNF API
- vFW CNF Upgrade Use Case
  - Available Scaling and Upgrade mechanisms in K8s
  - Platform provided Traffic Distribution mechanisms
  - Easier implementation of Build & Replace Upgrade Scenario for CNFs but could work also for VNFs (thanks to virtlet in K8s)
  - Acceleration of change management operations
  - Needed integration with CDS to improve service instantiation and its upgrade



Use of CDS for CNF Instantiation with K8S Plugin (Frankfurt)

- Utilization of SO Macro workflow
- CDS used for provisioning of input parameters for instantiation
- CDS uploads RB profile as a part of resource assignment workflow
- Service Design: Many Helm Charts in one CSAR
  - Decomposition into many VF-modules under one VNF
- MutliCloud-K8s Enhancements
  - Change of identifiers used to mapping between Helm Chart and vf-module
  - Simplified creation of vf-module with helm chart
    - Parameters from User Directives moved to SDNC Directives
    - Implemented default RB Profile so it is no longer mandatory resource for RB instantiation
    - K8splugin accepts input time parameters in time of instantiation

More about integration of CDS with MulticloudK8s: Today at 2 PM -> Terrace 2A