OPEN NETWORK AUTOMATION PLATFORM

Documentation: how to describe detailed call flows

Eric Debeau, Orange

September, 25 2017

Context

- Many interactions between ONAP components
- API documentation is adapted to understand each ONAP component
- Postman tools, curl command lines are cools, but does not visualize the sequence of API calls.
- But…
  - Not easy to get he full ‘picture’ to understand the various API calls
- Avoid tshark installation for reverse-engineering or deep code analysis
  - Many HTTPS calls

Objective

- Get a practical tool to visualize the ONAP interactions

- Understand the ONAP component behavior
  - API calls to other components

- Get the detail API calls
  - Including the mandatory headers, the body description, the call return
  - To facilitate learning
  - To help some test cases

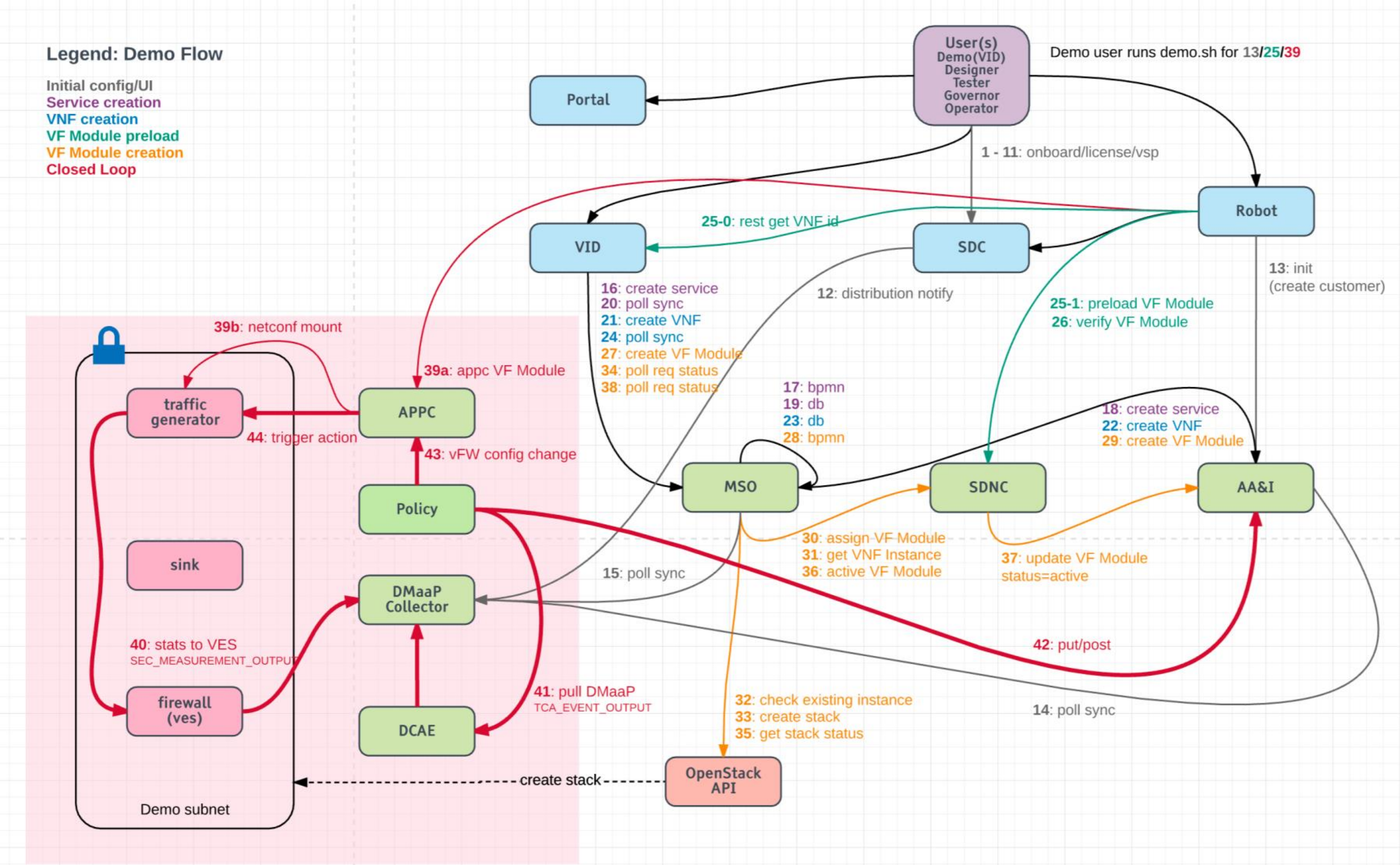
Existing solutions

- Basis call flows used for use-case description
  - Perfect to get a global vision
  - But not detailed to understand the variables exchanged between various calls

- Robot scripts
  - Perfect for tools to replay test suites
  - But “hard” to read and does not describe how a component reacted to an API call

```
Add ASDC License Group
  [Documentation] Creates an asdc license group and returns its id
  [Arguments] ${license_model_id} ${version_id}=0.1
  ${uuid}= Generate UUID
  ${shortened_uuid}= Evaluate str("${uuid}")[:23]
  ${map}= Create Dictionary key_group_name=${shortened_uuid}
  ${data}= Fill JSON Template File ${ASDC_KEY_GROUP_TEMPLATE} ${map}
  ${resp}= Run ASDC Post Request ${ASDC_VENDOR_LICENSE_MODEL_PATH}/${license_model_id}… ${data}
  Should Be Equal As Strings ${resp.status_code} 200 [Return] ${resp.json()['value']}
```

Call flow examples: global view



Legend: Demo Flow

Initial config/UI
Service creation
VNF creation
VF Module preload
VF Module creation
Closed Loop

User(s)
Demo(VID)
Designer
Tester
Governor
Operator

Demo user runs demo.sh for 13/25/39

Portal

Robot

VID

SDC

1 - 11: onboard/license/vsp

25-0: rest get VNF id

12: distribution notify

13: init
(create customer)

16: create service
20: poll sync
21: create VNF
24: poll sync
27: create VF Module
34: poll req status
38: poll req status

25-1: preload VF Module
26: verify VF Module

17: bpmn
19: db
23: db
28: bpmn

18: create service
22: create VNF
29: create VF Module

39b: netconf mount

39a: appc VF Module

traffic generator

APPC

44: trigger action

43: vFW config change

MSO

SDNC

AA&I

Policy

sink

DMaaP Collector

30: assign VF Module
31: get VNF Instance
36: active VF Module

37: update VF Module
status=active

15: poll sync

42: put/post

40: stats to VES
SEC_MEASUREMENT_OUTPUT

firewall
(ves)

DCAE

41: pull DMaaP
TCA_EVENT_OUTPUT

32: check existing instance
33: create stack
35: get stack status

14: poll sync

OpenStack API

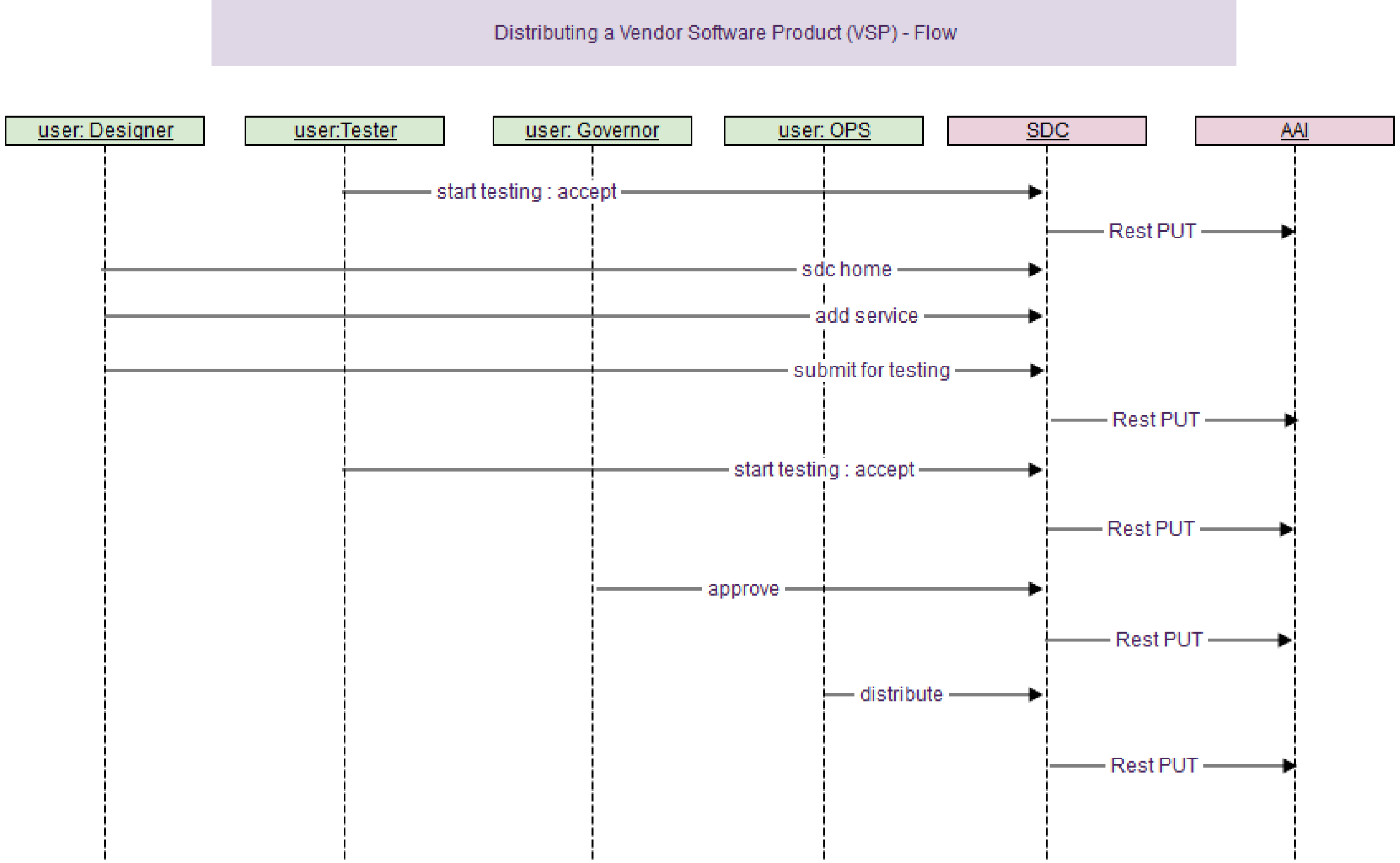
create stack

Demo subnet

https://wiki.onap.org/download/attachments/3245081/20170727_closed_loop_flow_Screenshot%202017-07-27%2011.57.40.png?version=1&modificationDate=1501171169000&api=v2

Call flow examples: Sequence

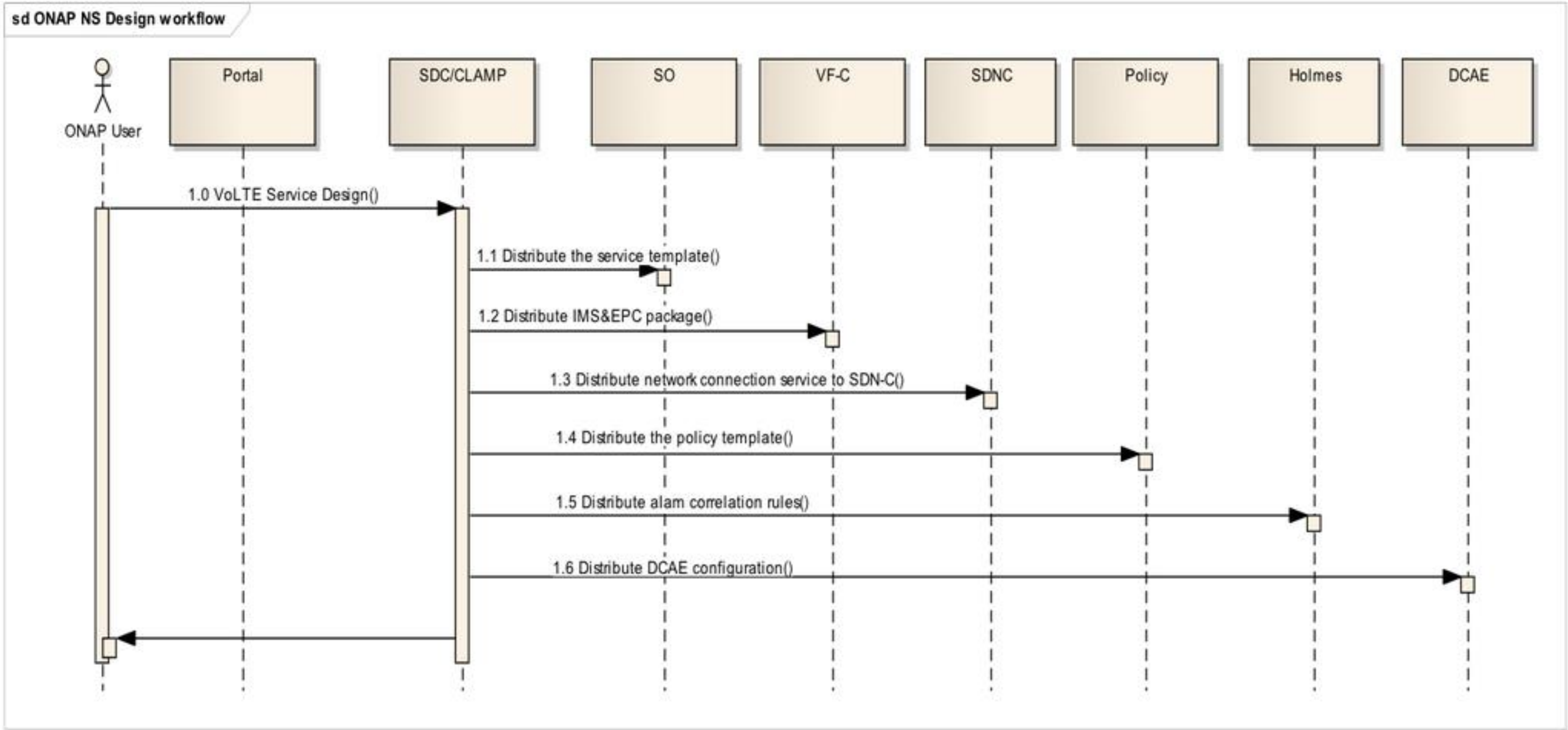
SDC Distribution Flow



Based on Gliffy tool, intregrated in the ONAP Wiki

https://wiki.onap.org/display/DW/Tutorial%3A+Verifying+and+Observing+a+deployed+Service+Instance#Tutorial:VerifyingandObservingadeployedServiceInstance-vFirewallFlow

Call flow examples: use-case description


sd ONAP NS Design workflow
ONAP User
Portal
SDC/CLAMP
SO
VF-C
SDNC
Policy
Holmes
DCAE
1.0 VoLTE Service Design()
1.1 Distribute the service template()
1.2 Distribute IMS&EPC package()
1.3 Distribute network connection service to SDN-C()
1.4 Distribute the policy template()
1.5 Distribute alam correlation rules()
1.6 Distribute DCAE configuration()

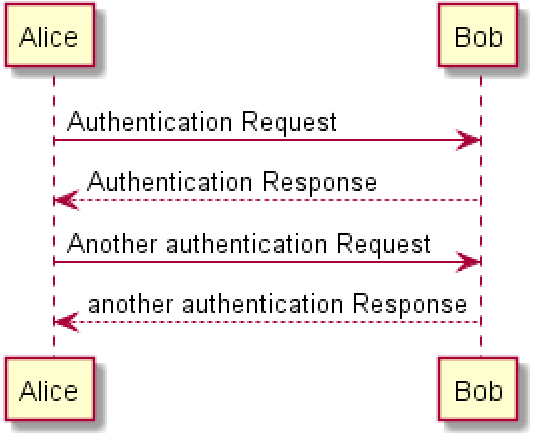
https://wiki.onap.org/download/attachments/6593603/image2017-5-17_18-34-39.png?version=3&modificationDate=1499700041000&api=v2

Proposal: using EtherPlant

- **EtherPlant** is an open-source tool allowing users to create UML diagrams from a PAD using PlantUML language
  - https://github.com/Orange-OpenSource/EtherPlant

- PlantUML
  - Text language to describe UML

```
@startuml
Alice -> Bob: Authentication Request
Bob --> Alice: Authentication Response
Alice -> Bob: Another authentication Request
Alice <-- Bob: another authentication Response
@enduml
```

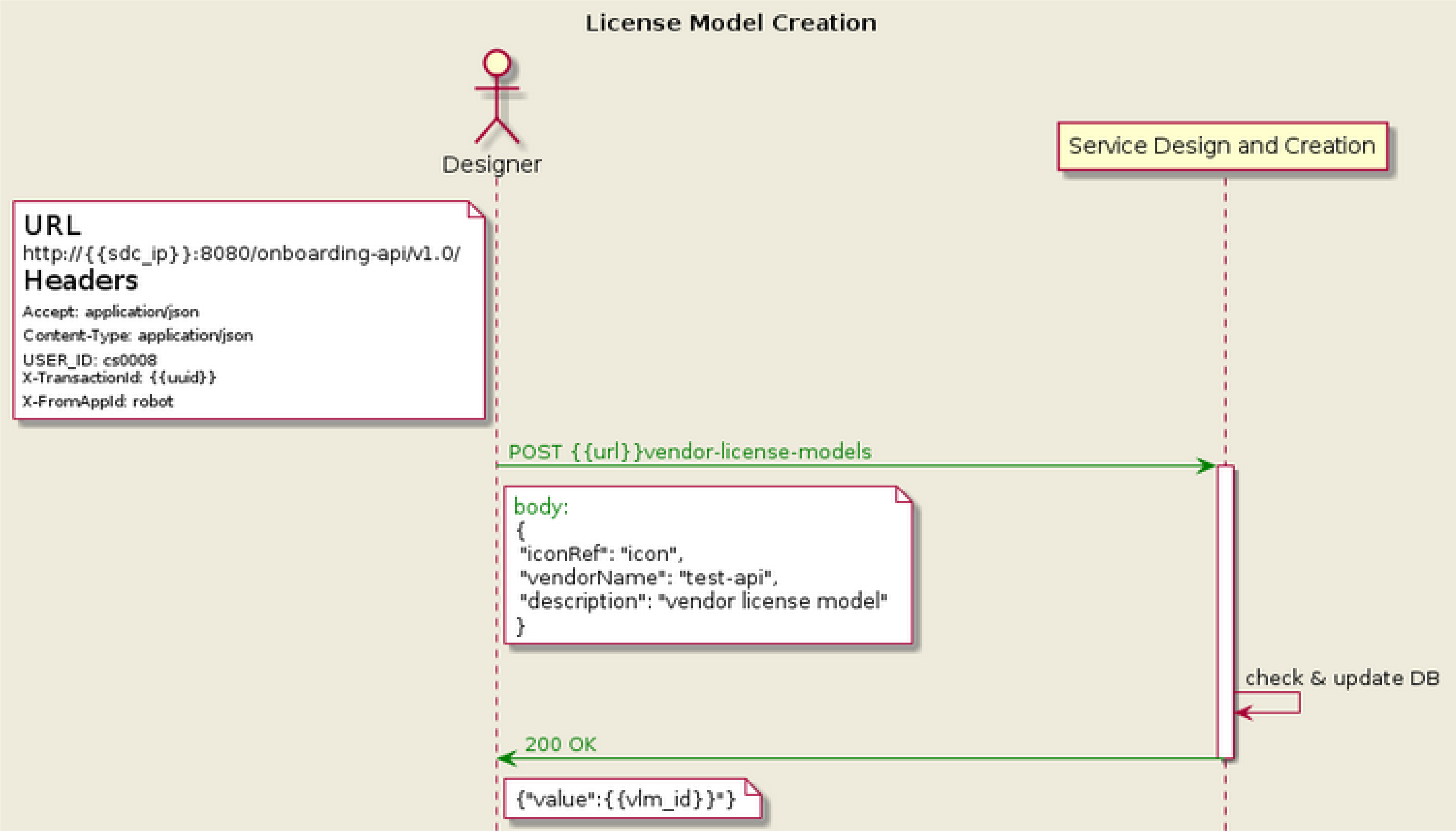

Alice
Bob
Authentication Request
Authentication Response
Another authentication Request
another authentication Response
Alice
Bob

Example: License Model Creation (SDC) - PlantUML

```
@startuml
title License Model Creation
skinparam backgroundColor #EEEBDC
actor Designer
participant "Service Design and Creation" as SDC
note left of Designer #white
<size:18>URL</size>
<size:12>http://{{sdc_ip}}:8080/onboarding-api/v1.0/</size>
<size:18>Headers</size>
<size:10>Accept: application/json</size>
<size:10>Content-Type: application/json</size>
<size:10>USER_ID: cs0008</size>
<size:10>X-TransactionId: {{uuid}}</size>
<size:10>X-FromAppId: robot </size>
end note

Designer -[#green]> SDC : <font color=green>POST {{url}}vendor-license-models
note right Designer #white
<font color=green>body:
{
 "iconRef": "icon",
 "vendorName": "test-api",
 "description": "vendor license model"
}
end note
activate SDC
SDC -> SDC : check & update DB
SDC -[#green]> Designer : <font color=green>200 OK
note right Designer #white
{"value":{{vlm_id}}"}
end note
deactivate SDC
```

Example: License Model Creation (SDC) - Graphic



License Model Creation

Designer

Service Design and Creation

URL

http://{{sdc_ip}}:8080/onboarding-api/v1.0/

Headers

```
Accept: application/json
Content-Type: application/json
USER_ID: cs0008
X-TransactionId: {{uuid}}
X-FromAppId: robot
```

POST {{url}}vendor-license-models

body:

```
{
 "iconRef": "icon",
 "vendorName": "test-api",
 "description": "vendor license model"
}
```

check & update DB

200 OK

```
{"value":{{vlm_id}}"}
```

Using EtherPlant to write PlantUML text description

- http://orange-opensource.github.io/EtherPlant/src/index.html?pad

- Pad example
  - Model License Creation : https://etherpad.opnfv.org/p/onap_plantuml1
  - Add resource to Catalog : https://etherpad.opnfv.org/p/onap_plantuml2

Discussion

- Is it useful for the community ?

- How to guarantee consistency with the code/API ?

- To be introduced in Documentation project ?
  - Sphinx extension available
  - http://build-me-the-docs-please.readthedocs.io/en/latest/Using_Sphinx/UsingGraphicsAndDiagramsInSphinx.html#id19

MERCI ;-)