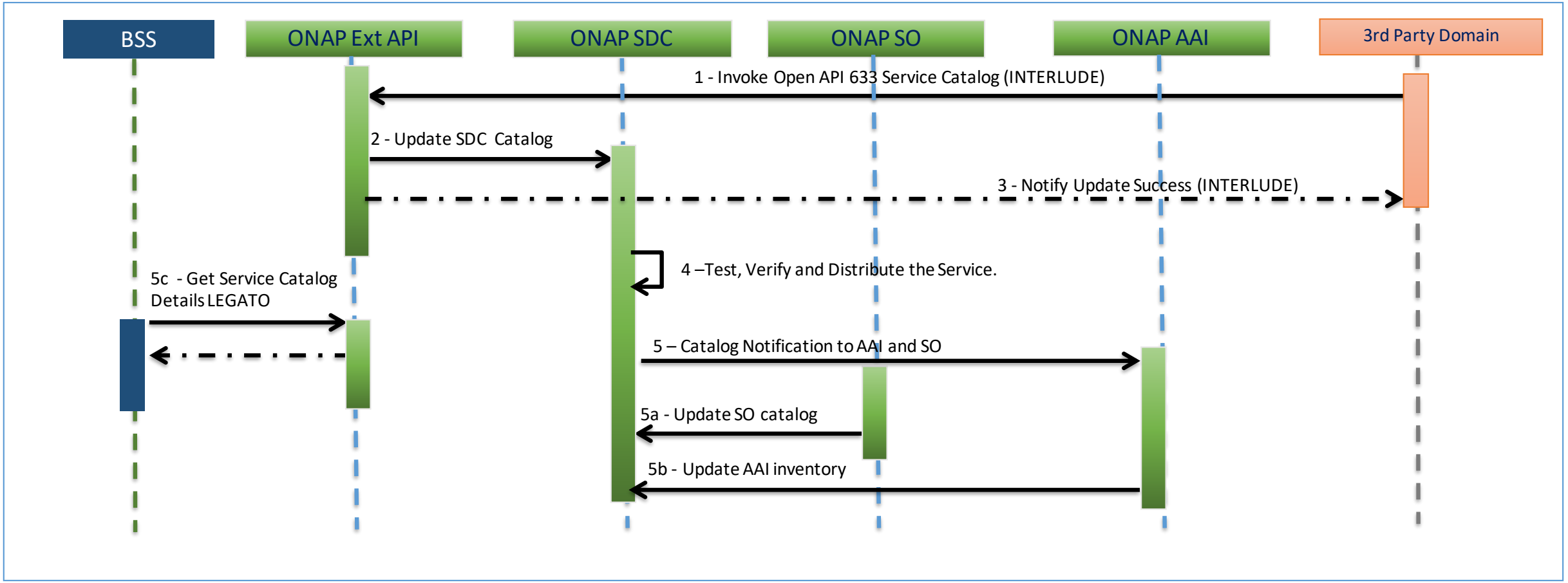
ONAP SDC– Third Party Service Definition Import

Telstra

May, 2019

Flow Diagram for 3rd Party Catalog Sync– Design Time



Upstream BSS layer (optionally NaaS layer can exist between BSS and ONAP)

ONAP components involved in the catalog sync and orchestration

External Partner Domain which is managed by ONAP

The flow steps (1 – 5c)

Catalog Sync Summary

1 – External Third party domain exports it service catalog details to Telstra. Telstra orchestrator ONAP exposes TMF Open API 633 Service Catalog API via ONAP Ext API component. Third Party Domain leverages the API 633 to POST the Service Catalog payload.
POST nbi/api/v2/serviceSpecification
Request body – TMF 633 Service Catalog compatible payload
Payload contents:
RFSS for Partner Domain Service
2 – ONAP Ext API updates SDC catalog by invoking internal SDC API
POST sdc/v1/catalog/services
3 – Ext API notifies Third party after successful update within ONAP
4 – Service Definition Updates / Creation of Composite Service happen in SDC UI (any manual updates to the received service definition)
Test, Verify and Distribute the Service definition. SDC updates other ONAP components (which have registered with SDC DMaaP) with catalog details
5a – SO pulls SDC catalog details
5b – AAI pulls inventory details
Ext API notifies northbound systems (BSS/NaaS) after successful import of the service catalog into ONAP.
5c - BSS retrieves catalog information from ONAP

Impact Analysis So Far for 3rd Party Catalog Sync

SDC

- Expose POST functionality of SDC Onboarding API as an external API within ONAP
- Reuse sdc-dao to update the Cassandra database and store the new service in SDC catalog
- Reuse SDC distribution functionality to distribute the new service to registered ONAP components (no change)
- Existing UUID creation logic will be used
- Last mile access service from 3rd party will be used for detailed analysis and reference implementation
- TOSCA based onboarding in work is progress in SDC, it supports heat based onboarding only. The TOSCA based work is ongoing separately in Modeling project. This dependency on Modeling project need to be looked into.

Ext API / NBI

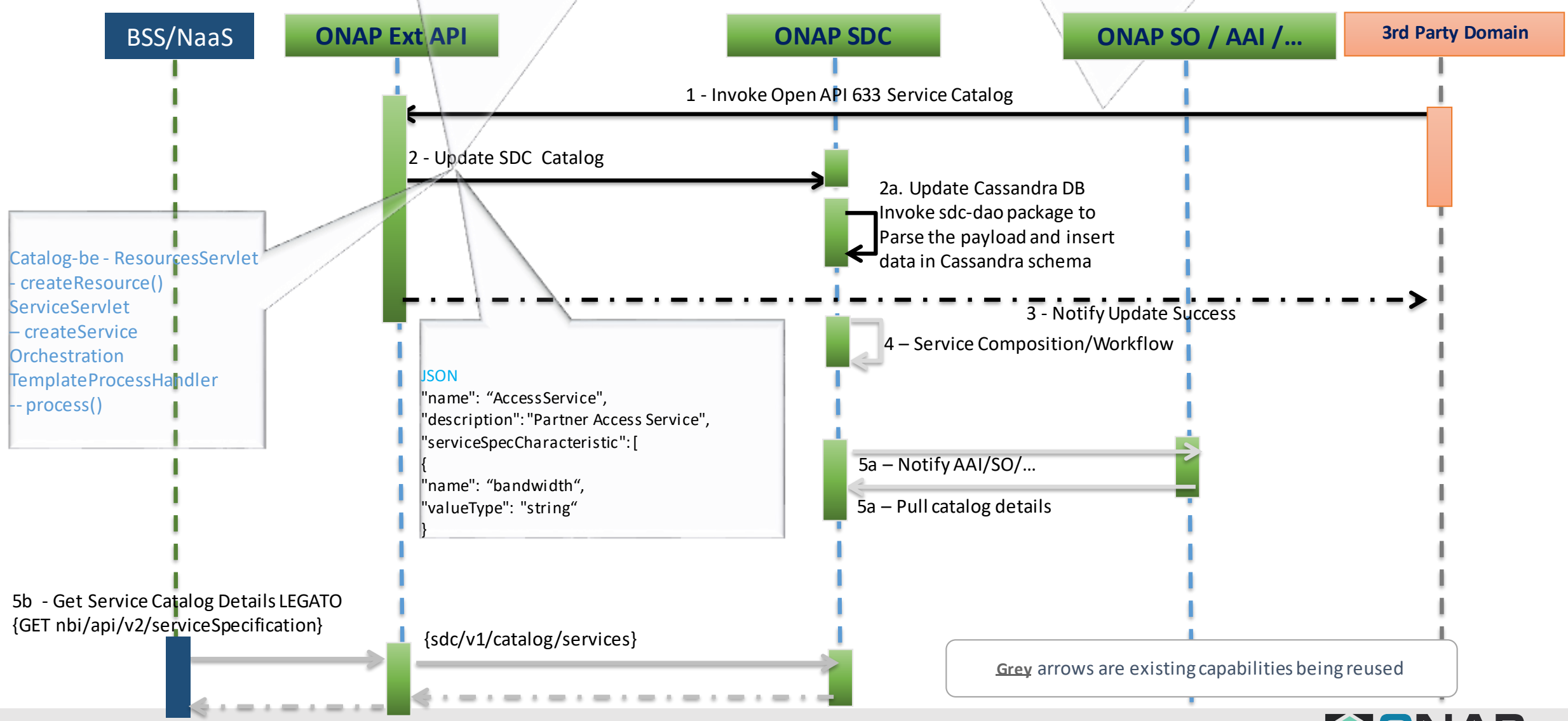
- Introduce POST for TMF API 633 – Service Catalog API
- Realization of POST operation in Ext API will depend on decisions taken during SDC implementation.

- Ext API changes to be planned for future release

Flow Diagram for 3rd PARTY SDC Catalog Sync

Invoke onboarding API {POST sdc/v1/catalog/services} JSON contents:
RFSS for partner service to be created as a Service in ONAP

Publish Resource or RFSS to ONAP using JSON
(INTERLUDE) {POST nbi/api/v2/serviceSpecification}



Steps of Flow Diagram SDC Catalog sync

Steps 1 to 3 on previous slide - which are part of new functionality - are explained here.
Steps 4 onward depict existing functionality reused

1 – Invoke TMF 633 Service Catalog API
3rd Party Domain's Payload to be submitted as a JSON –
Expected format - Service Specification payload specified by TMF 633

2- Ext API to invoke SDC onboarding API to updated ONAP SDC catalog
Invoke ServiceServlet - createService() – JSON payload
Currently on-boarding API is invoked when Create Service button is clicked in SDC UI
Ext API needs to be added as a consumer of the API
Existing logic to be reused:
- UUID creation in validateServiceBeforeCreate
- Logic to add default TOSCA components

2a – Persist the service in SDC database

3-Ext API will Notify 3rd Party after SDC catalog update
- Register for Distribution: Ext API will register itself with SDC.
- Ext API will receive distribution notification from SDC after service catalog creation in SDC
- Ext API notifies 3rd Party Domain

Payload structure of input to ONAP from Third Party


```
{
  "id": "2944ce7c-a7ce-4816-b08c-d51b8bbb2830",
  "name": "partner Access",
  "description": "Partner Access",
  "version": "v1.0.0",
  "lifecycleStatus": "Active",
  "serviceSpecCharacteristic": [
    {
      "name": "serviceDetails",
      "description": "Service details",
      "valueType": "object",
      "@type": "ServiceSpecCharacteristic",
      "minCardinality": 1,
      "maxCardinality": 1,
      "access": [
        "Create",
        "Read",
        "Update"
      ],
     "serviceSpecCharacteristicAttributes":[…],
      "configurable": false,
      "isUnique": false,
      "extensible": false
    },
    {  "name": "order"…},
    {  "name": "access"…},
],
  "@type": "NetworkServiceSpecification",
  "isBundle": false,
  "lastUpdate": "2019-05-17T06:37:31.911Z"
}
```

Payload structure of input to SDC Service creation API-with sample

```
{
"contactId": "cs0008"
"categories": [{}]
"name": "ExtService",
"tags": ["ExtService"],
"componentType": "SERVICE",
"projectCode": "010203",
"properties": [{}]
"inputs": [{}]
"ecompGeneratedNaming": true,
"serviceApiArtifacts": {},
"instantiationType": "A-la-carte",
"environmentContext": "General_Revenue-Bearing"
}
```

```
"name": "Partner",
"normalizedName": "Partner",
"uniqueId":
"serviceNewCategory.Partner",
"icons": ["Partner"],
"subcategories": null,
"version": null,
"ownerId": null,
"empty": false,
"type": null
```

```
{
"uniqueId": "",
"type": "integer",
"required": false,
"definition": false,
"description": "size",
"password": false,
"name": "addressId",
"hidden": false,
"immutable": false,
"parentUniqueId": "",
"isDeclaredListInput": false,
"schemaType": "",
"schemaProperty": {
"type": "",
"required": false,
"definition": true,
"password": false,
"hidden": false,
"immutable": false,
"isDeclaredListInput": false,
"getInputProperty": false,
"empty": false
},
"getInputProperty": false,
"ownerId": "",
"empty": false
}
```

D:\Telstra\
teService-access1.j

Placeholder for attributes needed for instantiation

Structure of SDC generated TOSCA CSAR

Below is the expanded view of the TOSCA CSAR generated by ONAP SDC.
Definitions – contains the interface yaml file which contains the metadata definition of the properties defined in the payload (detailed in previous slide)

No **Artifacts** folder as there are no deployment artifacts for the partner service

- Definitions
- TOSCA-Metadata
- csar.meta

- annotations.yml
- artifacts.yml
- capabilities.yml
- data.yml
- groups.yml
- interfaces.yml
- nodes.yml
- policies.yml
- relationships.yml
- service-ServicePocTest1-template.yml
- service-ServicePocTest1-template-interface.yml

```yaml
tosca_definitions_version: tosca_simple_yaml_1_1
imports:
- nodes:
    file: nodes.yml
- datatypes:
    file: data.yml
- capabilities:
    file: capabilities.yml
- relationships:
    file: relationships.yml
- groups:
    file: groups.yml
- policies:
    file: policies.yml
- annotations:
    file: annotations.yml
node_types:
  org.openecomp.service.ServicePocTest1:
    derived_from: org.openecomp.resource.abstract.nodes.service
    properties:
      bandwidth:
        type: scalar-unit.size
        description: capacity
        required: false
      contact:
        type: object
        required: false
        entry_schema:
          type: string
      UnstructuredAddress:
        type: object
        required: false
        entry_schema:
          type: string
```

TOSCA-Meta-File-Version: 1.0
CSAR-Version: 1.1
Created-By: Carlos Santana
Entry-Definitions: Definitions/service-ServicePoc2-template.yml
Name: csar.meta
Content-Type: text/plain

SDC-TOSCA-Meta-File-Version: 1.0
SDC-TOSCA-Definitions-Version: 9.0

Service spec characteristics get added here

service-ServicePocTest1-csar.zip

Service Definition – Sample Access Service – Service Catalog API payload

ServiceSpecCharacteristics – sample access service

- Object – serviceDetails
- Object – order
- Object - access

serviceSpecCharacteristicAttributes – serviceDetails

- lineOfBusiness – enum
- bandwidth – object

serviceSpecCharacteristicAttributes – order

- installationWorkforce – enum
- orderSLA – enum
- orderReferenceId – string
- appointmentId – string
- endUserEngagementType – string
- informedConsent - boolean

serviceSpecCharacteristicAttributes – access

- addressId – number
- transitionOrder – object
- vlanMode – enum
- ntd – object
- copperPairId – string
- features – object
- voice - object

Service Definition – Sample Access Service – Service Catalog API payload

serviceSpecCharacteristicAttributes - bandwidth

1. type – enum
2. downstream - enum
3. upstream - enum
4. units - enum

values – type [standard, guaranteed]
values – downstream [128, 512, 1, 2, 1000]
values – upstream [128, 512, 1, 2, 1000]
value s- unit – [k, m]

Challenges

No option to map child parent property in SDC Design

No option to add a property of type object with its child attributes

Options

Option – 1
Introduce logic to group the properties retrieved from the service catalog payload and map the child property to its parent

Option – 2
Accept the payload in hierarchical form so that the service spec attributes relationship can be maintained

ONAP
OPEN NETWORK AUTOMATION PLATFORM

Thank you