
LF
NETWORKING
LFN Developer & Testing Forum

LF NETWORKING

LFN Developer & Testing Forum

# ASD App Package Management

## Onboarding and LCM Orchestration

Byung-Woo Jun (Ericsson)
Michael Morris, Andre Schmid (Ericsson)
Marian Darula, Zu Qiang (Ericsson)

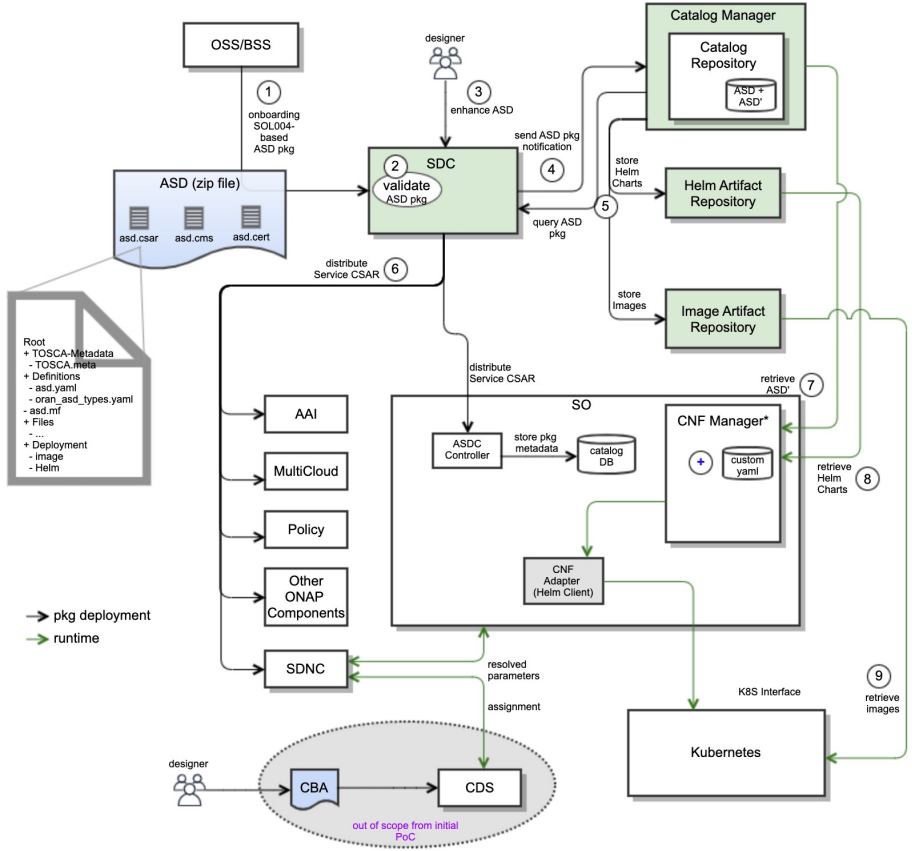
# ASD App Pkg Management PoC Goals



- ASD App package (focus on NF) onboarding to SDC
  - Onboard vendor ASD App Packages and the package validation
  - Manage large-size ASD App packages (i.e., handling large-size images)
  - Transform onboarding ASD models & packages into internal ONAP models & packages
- SDC Resource and Service Design with ASD-aligned CNF resources
  - Add additional artifacts such as vendor licensing models
  - Allow SDC designers to define class-level ASD input parameters add custom properties
- Service CSAR with ASD resources distribution to ONAP Runtime components
  - Keep the SDC pull distribution pattern
  - Distribute ASD Service CSAR to participating ONAP runtime components (notifications & querying)
  - Enhance ONAP Catalog Manager to do the following:
    - Store enhanced/internal ASD Descriptors and packages in the Catalog Repository
    - Store Helm Charts in the Helm Chart Repository
    - Store Images in the Image Repository
- SO Model-driven Orchestration for ASD-aligned CNFs (and NSs in the future)
  - Select ASD workflows (invoking CNF Manager) based on models, metadata and/or request types
    - Enhance SO model-driven orchestration for ASD-CNF
  - Demonstrate how ASD, input parameters and Helm Charts are processed by a new SO plugin (CNF Manager)
    - Resolve K8 resources based on input parameters and Cloud Artifacts
    - Get placement decisions by leveraging ONAP platform capabilities (OOM is a future consideration)
    - Decompose ASD (VF) into deployment Items (vf-modules) and deploy each deployment item (vf-module) to Kubernetes thru SO CNF Adapter; some dry-run would be executed based ASD parameters to avoid deployment failure
    - Support ASD-level (VF-level) topology (CNF and sub-resources) by leveraging CNF Manager

# ASD App Pkg Onboarding & Distribution

LFN Developer & Testing Forum

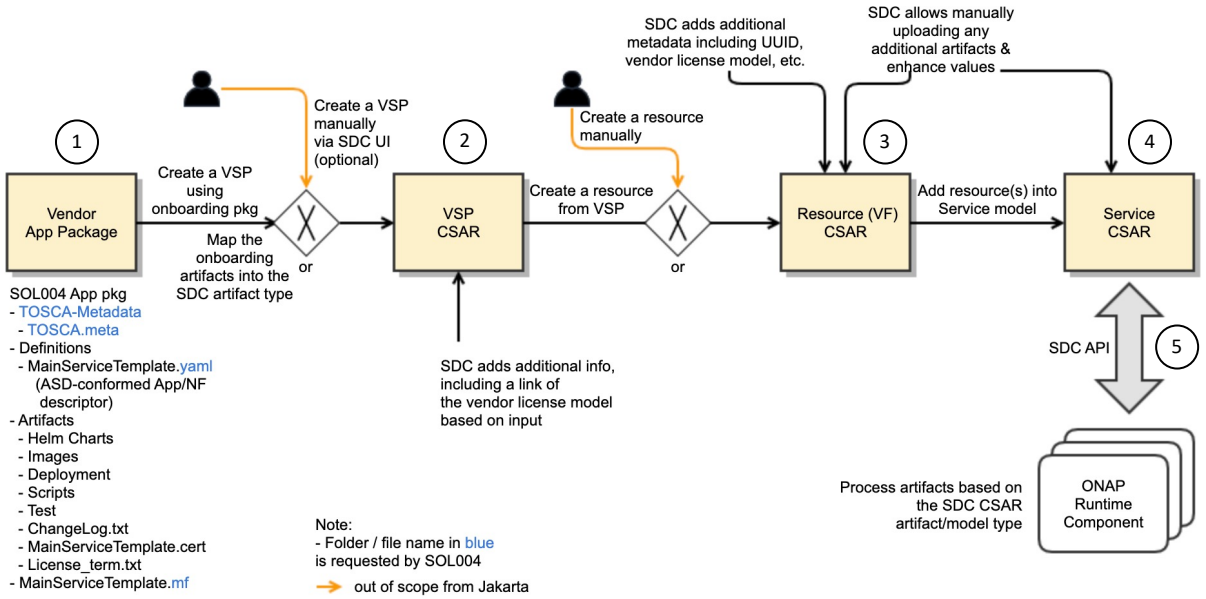



1. OSS/BSS onboards vendor ASD App packages, which include ASD, signature and certificate, to ONAP thru SDC.
   A. For the ASD and Package specification proposal, see https://wiki.lfnetworking.org/display/LN/2022-01-12+-+ONAP%3A+Application+Service+Descriptor+%28ASD%29+for+K8s+NFs
2. SDC validates onboarding ASD App packages. After then, SDC goes thru its onboarding process – see the subsequent slide page for details.
3. SDC designers can enhance ASD by adding/updating additional artifacts and customizing values.
4. Once the onboarded ASD App packages are tested and certified, SDC sends App pkg notifications thru DMaaP – SDC keeps its ”pull” distribution pattern.
5. Catalog Manager queries ASD App package from SDC and distribute:
   A. ASD and ASD’ CSAR to the Catalog Repository
   B. Helm Charts to the Helm Artifact Repository
   C. Images to the Image Artifact Repository
6. SDC Service CSAR is distributed to other participating ONAP runtime components, such as SO, AAI, MultiCloud, Policy, SDC, etc.
7. CNF Manager (or others) retrieves ASD/ASD’ from the Catalog Manager.
8. CNF Manager (or others) retrieves associated Helm Charts from the Helm Artifact Repository.
9. K8S (CISM, CIS) retrieves image info and images from the Image Artifact Repository.

# SDC ASD App Pkg Onboarding Process

LFN Developer & Testing Forum

## SDC Internal Process



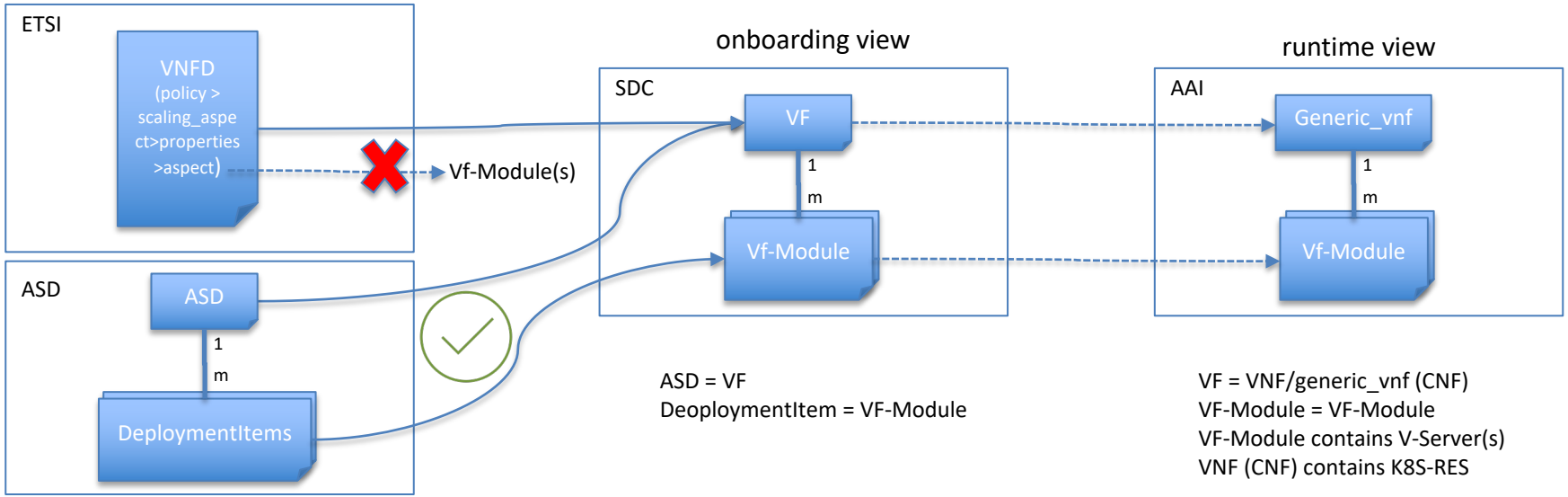

SDC component manages ASD App Package onboarding and distribution.

1. The ASD App package is compliant to the ETSI NFV SOL004 CSAR structure with some extensions.
   A. The ASD App package has new “ASD” specific metadata, which is used for making orchestration path decisions.
2. SDC generates its internal “VSP” CSAR file, based on the onboarded ASD App Package.
   A. SDC adds additional info, including vendor license model.
3. Then, SDC generates Resource (VF) CSAR, based on the corresponding VSP.
   A. ASD-specific metadata is copied into the specific files, which will be used by Runtime components.
   B. Onboarded ASD models are mapped into the ONAP internal models; ASD <-> ONAP VF, DeploymentItems <-> VF-Modules).
4. SDC creates Service CSAR by adding resource(s) into the Service Model.
5. SDC distributes the Service CSAR to ONAP runtime components thru DMaaP (pull pattern)

# Comparing ETSI/ASD/SDC/AAI Models

LFN Developer & Testing Forum

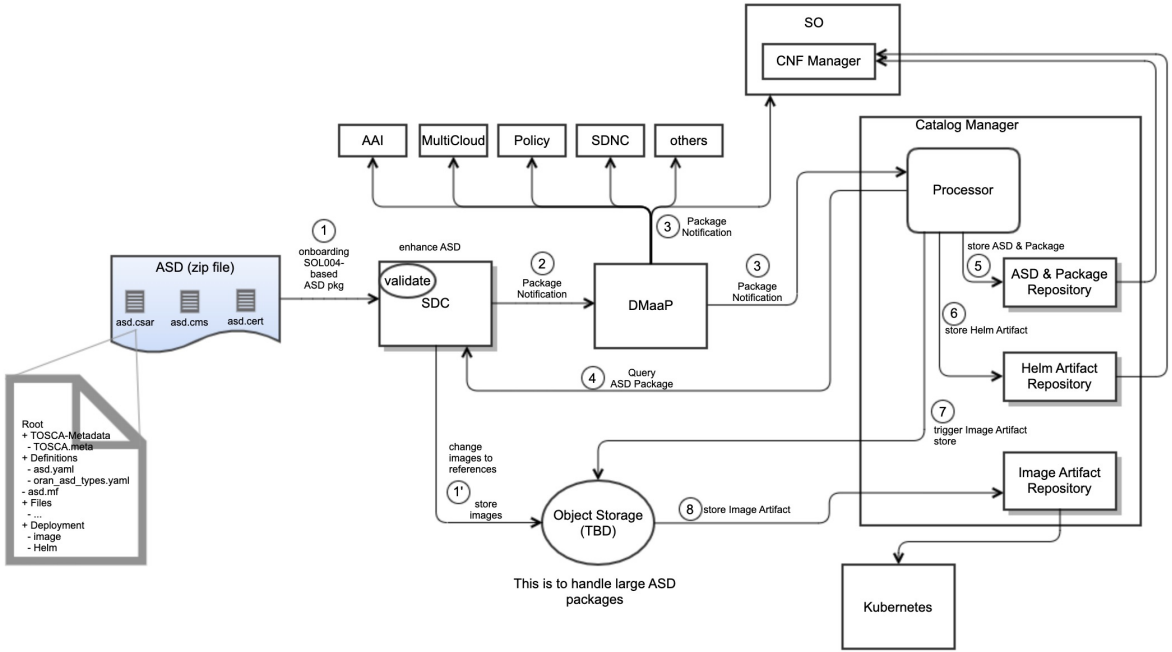
- ETSI VNFD does not have its sub-structure to correspond to ONAP vf-module
  - During mapping, vf-module could be deuced based on the VNFD policies<scaling_aspects>properties>aspect, this mapping is not yet realized for its complexity; current ONAP ETSI-Alignment supports VNF and above, not vf-module, by leveraging external components
  - There are some issues to support closed loop (healing and scaling) that are run by ONAP components
- In ASD, there is the deploymentItems (1..N) under the ASD, and each deploymentItem would be mapped to the vf-module by SDC to minimize impact on existing ONAP components
- SDC VF and vf-module will be corresponded to AAI generic-vnf and vf-module, accordingly for runtime instance representation (like CNFO)



# SDC & Catalog Manager Design Proposals for ASD App Pkg Distribution

LFN Developer & Testing Forum

Keep SDC pull distribution pattern




1. ASD App package is onboarded to SDC
   i. if image(s) are large (> 2M), stores them to some Object Storage (TBD)
   ii. SDC changes embedded Image(s) to Image references in the package
2. SDC sends the ASD App package notification to DMaaP
3. DMaaP participating components get the notification and query ASD App package from SDC

Catalog Manager processor performs the following:

4. Queries ASD App packages including all the artifacts
5. Stores ASD packages to Catalog Repository
6. Stores Helm Artifacts to Helm Artifact Repository
7. Triggers Image Artifact store
8. Image Artifact is stored to the Image Artifact Repository by Catalog Manager

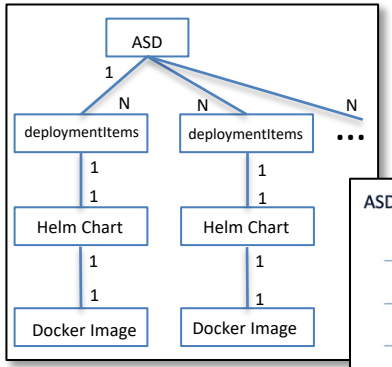
Note: Catalog Manager can provide CRUDQ APIs for ASD App Packages

# Complex ASD App Package Management

LFN NETWORKING
LFN Developer & Testing Forum

| Attribute | Qualifier | Cardinality | Content | Description |
|---|---|---|---|---|
| asdId | M | 1 | Identifier | Identifier of this ASD information element. This attribute shall be globally unique. The format will be defined in the data model specification phase. |
| asdSchemaVersion | M | 1 | Version | Specifies the version of the ASD's schema (if we modify an ASD field definition, add/remove field definitions, etc.) |
| asdProvider | M | 1 | String | Provider of the AS and of the ASD. |
| asdApplicationName | M | 1 | String | Name to identify the Application Service. Invariant for the AS lifetime. |
| asdApplicationVersion | M | 1 | Version | Specifies the version of the Application (so, if software, deplolymentItems, ASD values, … change, this changes) |
| asdApplicationInfoName | M | 0..1 | String | Human readable name for the Application service. Can change during the AS lifetime. |
| asdInfoDescription | M | 0..1 | String | Human readable description of the AS. Can change during the AS lifetime. |
| asdExtCpd | M | 0..N | datatype.ExtCpd | Describes the externally exposed connection points of the application. |
| enhancedClusterCapabilities | M | 0..1 | datatype. enhancedClusterCapabilities | A list of expected capabilities of the target Kubernetes cluster to aid placement of the application service on a suitable cluster. |
| deploymentItems | M | 1..N | DeploymentItem | Deployment artifacts |

- An ASD App package could represent complex applications, which consist of multiple deployment items/Helm Charts/Images
  - 1 ASD contains 1..N DeploymentItems
  - In order to support complex applications that require multiple artifacts (like Helm charts) to be installed in a particular order
  - ASD App Orchestrator (SO CNF Manager) manages chaining of these artifacts, based on the deploymentOrder
- Each deployment item corresponds to a Helm Chart.
- Each Helm Chart corresponds to a docker image.




| Attribute | Qualifier | Cardinality | Content | Description |
|---|---|---|---|---|
| deploymentItemId | M | 1 | Identifier | The identifier of this deployment item |
| artifactType | M | 1 | String | Specifies the artifact type. One of following values can be chosen: "helm_chart", "helmfile", "crd", "terraform" |
| artifactId | M | 1 | String | Reference to a DeploymentArtifact. It can refer to URI or file path. |
| lifecycleParameters | M | 0..N | String | The list of parameters that can be overridden at deployment time (e.g., the list of parameters in the values.yaml which can be overridden at deployment time) |
| deploymentOrder | M | 0..1 | Integer | Specifies the deployment stage that the DeploymentArtifact belongs to. A lower value specifies that the DeploymentArtifact belongs to an earlier deployment stage, i.e. needs to be installed prior to DeploymentArtifact with higher deploymentOrder values. If not specified, the deployment of the DeploymentArtifact can be done in arbitrary order and decided bu the orchestrator. |


note: Information Models are shown here for illustration purposes. The corresponding ASD data models are available at https://wiki.onap.org/display/DW/Application+Service+Descriptor+%28ASD%29+Resource+Data+Model

# ASD Data Model



- ASD Data Model (https://wiki.onap.org/display/DW/Application+Service+Descriptor+%28ASD%29+Resource+Data+Model)
- During onboarding, SDC transforms this Data Model into SDC Data Model

**tosca.nodes.asd**

```
tosca.nodes.asd:
    derived_from: tosca.nodes.Root
    description: "The ASD node type"
    version: 0.1
    properties:
      descriptor_id:
          type: string # UUID
          required: true
          description: Identifier of this ASD. It is in UUID format as specified in RFC 4122
      descriptor_invariant_id:
          type: string # UUID
          required: true
          description: >
              Identifier of this descriptor in a version independent manner. This attribute
              is invariant across versions of ASD. It is in UUID format as specified in RFC 4122
      version:
          type: string
          required: true
          description: Identifies the version of the ASD.
      schema_version:
          type: string
          required: true
          description: Identifies the Identifies the version of this ASD's schema.
      function_description:
          type: string
          required: false
          description: Description of the application service described by this ASD.
      provider:
          type: string
          required: true
          description: Identifies the provider of the ASD.
      application_name:
          type: string
          required: true
          description: Name to identify the application service described by this ASD
      application_version:
          type: string
          required: true
          description: Identifies the version of the application service described by this ASD.
      ext_cpds:
          type: list
          required: false
          entry_schema:
              type: tosca.datatype.asd.extCpdData
          description: >
                  Describes the externally exposed connection points of the application
                  service described by this ASD
      enhanced_cluster_capabilities:
          type: tosca.datatype.asd.enhancedClusterCapabilities
          required: false
          description: >
                  A list of  expected capabilities of the target Kubernetes cluster to aid
                  placement of the application service on a suitable cluster.
```

**tosca.artifacts.asd.deploymentItem**

```
tosca.artifacts.asd.deploymentItem:
    version: 0.1
    derived_from: tosca.artifacts.Root
    description: "Describes the artifact type of asd deployment item"
    file: "URI or path of the artifact"
    properties:
      item_id:
        description: "The identifier of this asd deployment item"
        required: true
        type: string
      artifact_type:
        description: >
           Specify artifact type.
        required: true
        type: string
        constraints:
          - valid_values: ["helm_chart", "helmfile", "crd", "terraform" ]
      deployment_order:
        description: >
           Specifies the deployment stage that the DeploymentArtifact belongs to.
           A lower value specifies that the DeploymentArtifact belongs to an earlier
           deployment stage. When this value is omitted, the deployment order
           will be decided by the orchestrator.
        required: false
        type: integer
      lifecycle_parameters:
        description: "list of parameters that can be overridden at deployment time "
        required: false
        type: list
        entry_schema:
           type: string
```

- tosca.datatypes.asd.extCpdData
- tosca.datatypes.asd.networkInterfaceRequirements
- tosca.datatypes.asd.paramMappings
- tosca.datatypes.asd.enhancedClusterCapabilities
- tosca.datatypes.asd.customResourceRequirement
- tosca.datatypes.asd.requiredPlugin

# SDC Model Transformation between ASD & SDC Resource VF Models



- The org.onap.asd.CNF properties are defined in the nodes.yaml (A)
- node template added for the ASD type generated based on the definition in the onboarded ASD model (B)
- Set the “type” in the asd_instance indicates ASD (e.g., type: ASD) by leveraging the ASD package metadata ( C )
  - entry_definition_type: [asd]
  - A few options are under discussion to identify the ASD App package type
- ASD deploymentItem(s) are transformed into group(s) in the resource template yml, and the group is represented by vfModule (D)
  - deploymentItem = vfModule
- Could extend the org.openecomp.groups.VfModule to hold the DeploymentItems properties, such as deployment_order and lifecycle parameters ( E )
- Helm Charts will be added to Artifacts/Deployment/HELM (like what is done today for onap zip cnf packages) (F)
- Image files will be added to Artifacts/Deployment/IMAGE (G)

nodes.yaml

A

```
org.onap.asd.CNF:
  derived_from: tosca.nodes.Root
  properties:
    asd_id:
      type: string
      description: UUID of ASD
      required: true
    asc_schema_version:
      type: string
      description: version of ASD
      required: true
    asd_provider:
```

B

```
node_templates:
  asd_instance:
    type: org.onap.asd.CNF
    metadata:
      invariantUUID: 3948bd3d-f4e2-41a9-b3b
edc6c6db927e
      UUID: 05329593-52f8-482d-a967-8dbb28b
      name: Asd1.CNF
      description: Not reusable inner VFC
      category: Generic
      version: '1.0'
      customizationUUID: 59c10655-f68e-47d3
d8f5698f6f75
      type: ASD      // used be VFC
      subcategory: Abstract
      resourceVendor: d
      resourceVendorRelease: 2.6.1
      reourceVendorModelNumber: ''
    properties:
      asd_id: ASD Instance
      asd_schema_version: 1.0.0
```

C

D

```
group:
  Asd..helmA..module-0:
    type: org.openecomp.groups.VfModule
    metadata:
      vfModuleModelName: Asd..helmA..module-0
      vfModuleModelInvariantUUID: 766017db-5c11-
47f9-a3c4-1fed0dbae9cb
      vfModuleModelUUID: 57a35aad-4290-4b55-a0b2-
150aad6da058
      vfModuleModelVersion: '0.0'
    properties:
      min_vf_module_instances: 1
      vf_module_label: helmA
      max_vf_module_instances: 1
      vf_module_type: Base
      isBase: true
      initial_count: 1
      volume_group: false
      deployment_order: 1
      lifecycle_parameters:
        - "Values.db.fullBackupInterval"
        - "Values.db.walConsolidationInterval"
  Asd..helmB..module-1:
    type: org.openecomp.groups.VfModule
    metadata:
      vfModuleModelName: Asd..helmB..module-0
      vfModuleModelInvariantUUID: 766017db-5c11-
47f9-a3c4-1fed0dbae9cb
```

E

Artifacts

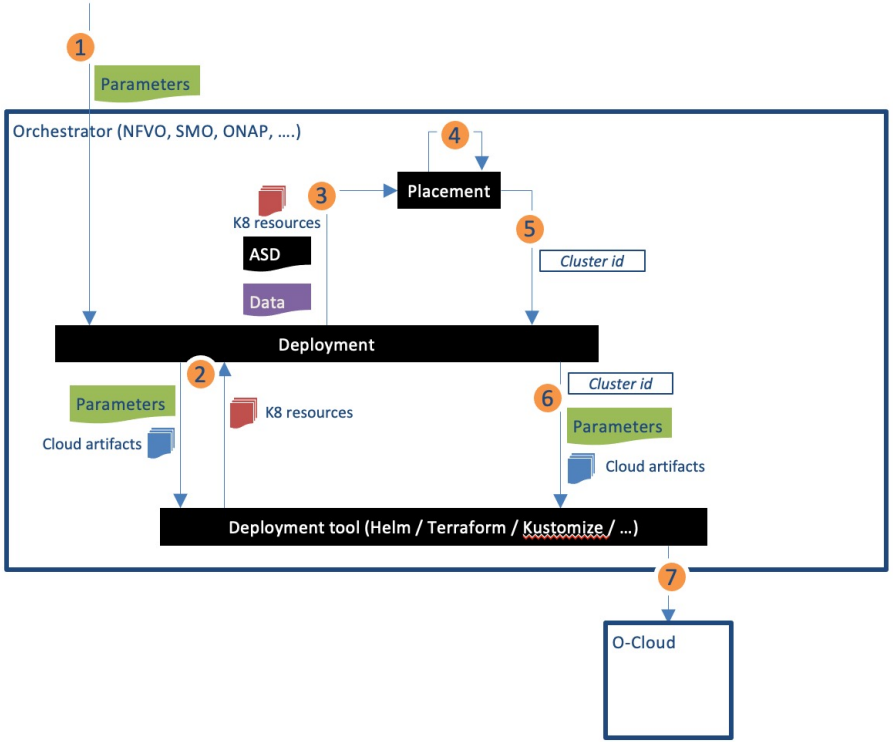
- Deployment
  - ORAN_PACKAGE
    - AS_<...>DataTypes.csar
  - HELM
    - <HELM Chart file>
    - <HLEM Chart file>
  - IMAGE
    - <image>
    - <image>

F

G

# ASD App Deployment Use Case



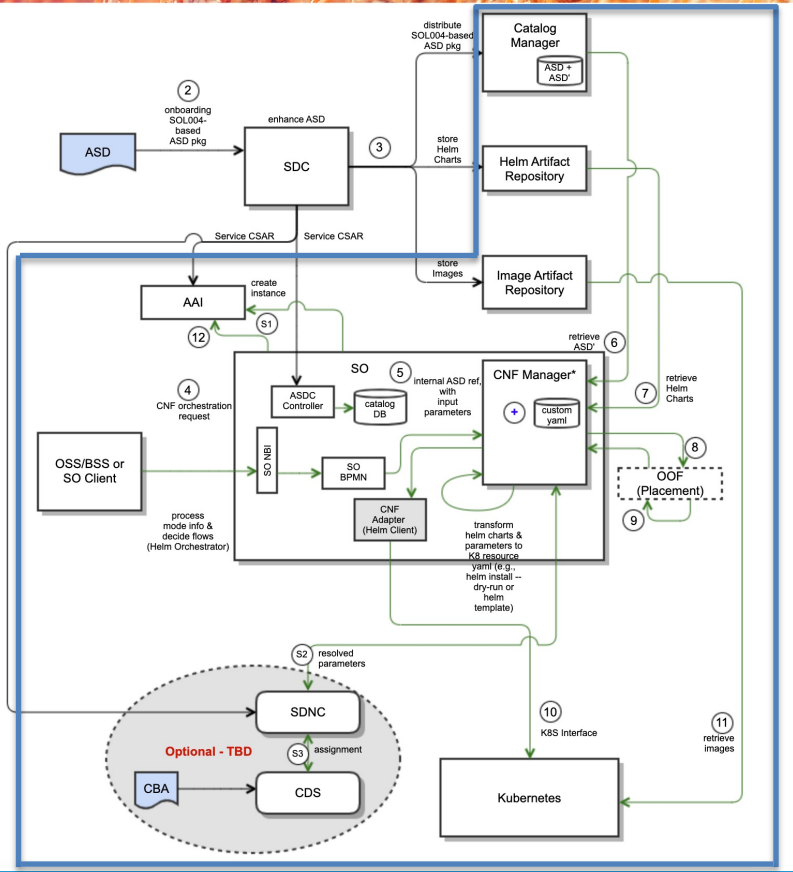



<source: ASD for K8S NFs, https://wiki.lfnetworking.org/display/LN/2022-01-12+-+ONAP%3A+Application+Service+Descriptor+%28ASD%29+for+K8s+NFs >

1. A deployment order is received, along with the required lifecycleParameters values
2. The cloud-native deployment tool is invoked with the received parameters to transform the cloud artifacts into K8S resource descriptions.
3. The K8S resource descriptions, ASD and any other relevant data is sent to the placement function
4. Placement decision is done based on input data
5. Inform deployment of placement
6. Request the cloud native deployment tool to deploy on the identified target cluster
7. Cloud native deployment tool deploys application in the chosen cluster using the K8S API.

# ASD App Orchestration Architecture

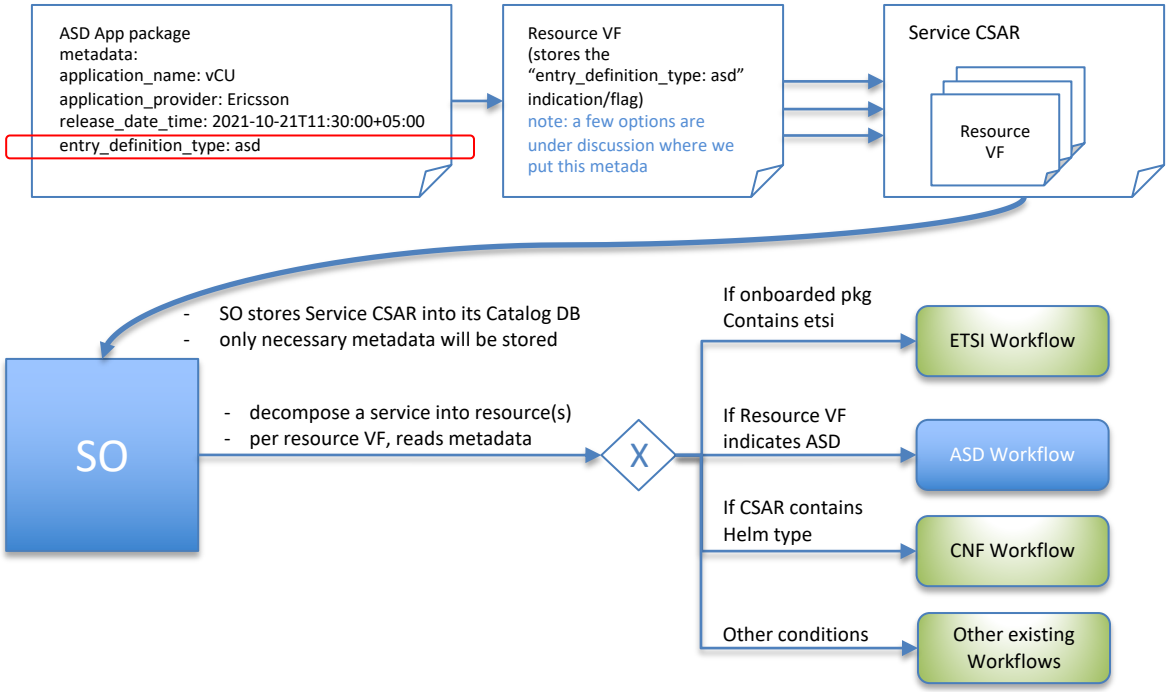
LFN Developer & Testing Forum




- Requirements (https://jira.onap.org/browse/REQ-1043)
  - Support for provisioning ASD-based CNFs using ONAP platform and external K8S Manager
  - Leverage SO and new SO plugin capabilities for ASD- and Helm-based CNF orchestration
  - Limit ASD and Helm Chart exposures to existing ONAP runtime orchestration components; i.e., make ASD and Helm Chart transparent to most of the ONAP components
- Proposed Process
  - SO gets the ASD-based CNF package from SDC and stores its metadata to its Catalog DB;
    - like current ETSI-Alignment, SO does not need to understand ASD and Helm Chart details
  - To decide orchestration flows, SO processes ASD model info, metadata and incoming request parameters
    - Decomposes a Service into VF resource(s); if NSD (or equivalence) is used, SO NFVO can be leveraged to handle NS (out-of-scope from the initial PoC)
    - If VF resource metadata indicates ASD-based VF, SO chooses the ASD-CNF workflows
  - SO launches ASD-CNF orchestration workflows and delegates ASD-CNF orchestration to a new SO plugin, SO CNF Manager (CNFM)
    - Passes ASD references along with input parameters
    - Use of CBA and CDS is future consideration (like CNFO). Currently, SO clients need to provide instance-level input parameters that are defined in the ASD and DeploymentItems
  - SO creates instances to AAI, by leveraging existing AAI VF and VF-Module schema

# SO Model-Driven ASD Workflow Selection

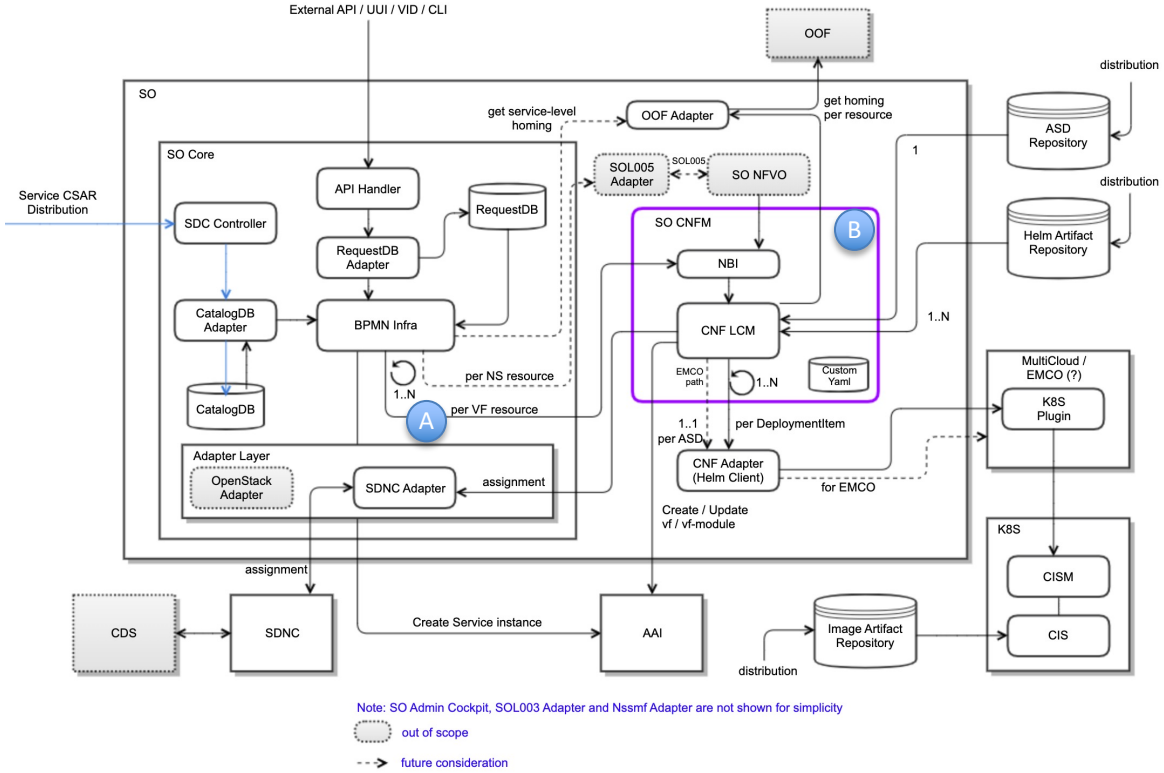
LFN Developer & Testing Forum




- ASD App package contains the package metadata.
- One of the metadata indicates the package type; entry_definition_type: asd
  - This is a required metadata
- During the ASD App package onboarding, SDC copies the metadata into the Resource VF.
- Resource VF(s) will be added to a Service CSAR for distribution.
- SO stores the Service CSAR into its Catalog DB; only necessary metadata will be stored
  - SO does not need to understand ASD and Helm Charts fully
  - Only ASD App metadata would be necessary (TBD)
- SO decomposes a service into resource VF(s)
- per resource VF, SO reads the corresponding metadata and determines a proper workflows
  - For the ASD case, SO will invoke SO CNF Manager
  - Most of existing SO workflows will be reused

# SO Internal Architecture for ASD LCM



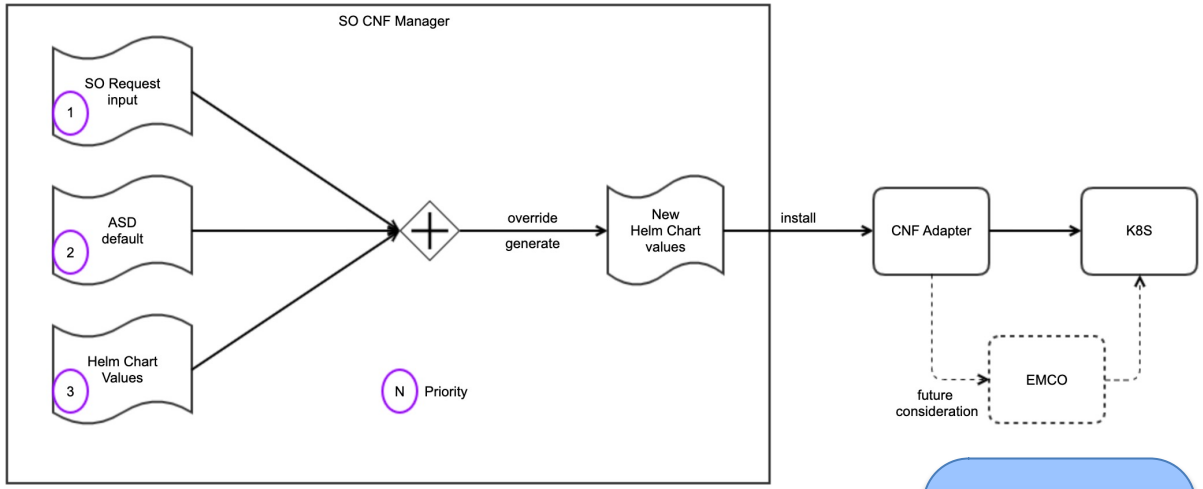



- SDC Controller saves Service CSAR into SO Catalog DB
- Per VF resource, SO BPMN Infra invokes SO CNFM for ASD instance orchestration (A)
- SO CNFM processes ASD-based CNF lifecycle orchestration (support ASD-level topology including vf-modules)
- for NS, SO can invoke SO NFVO per NS resource (out of scope from the initial PoC) thru the SOL005 Adapter
- SO CNFM processes ASD instance lifecycle (B)
  - Gets an ASD App onboarded package that is stored in the Catalog Repository
  - Gets associated Helm Chart(s) that are stored in the Helm Artifact Repository
  - Processes ASD deployment artifacts (each deployment artifact = a Helm Chart)
  - Per deployment artifact, constructs a new customized Helm Chart values file, based on 1) instance-specific incoming parameter values, 2) default values from ASD and 3) Helm Chart
  - Transforms ASD cloud artifacts into K8S resource descriptions
  - Gets a placement decision (e.g., target cluster) from the placement function (e.g., OOF)
  - Thru the SO CNF Adapter, sends request(s) to K8S to deploy ASD instance(s) on the target cluster
  - SO CNF Adapter invokes MultiCloud K8S plugin or EMCO (in the future); if EMCO is used, CNF adapter may need to be handle at the ASD-level (and also SO CNFM will be adjusted as needed).

Note: Use of CBA/CDS is a future consideration (plan to collaborate with CNFO)

# CNF Manager Input Parameter Handling

LFN Developer & Testing Forum




| Attribute | Qualifier | Cardinality | Content | Description |
|---|---|---|---|---|
| deploymentItemId | M | 1 | Identifier | The identifier of this deployment item |
| artifactType | M | 1 | String | Specifies the artifact type. One of following values can be chosen: "helm_chart", "helmfile", "crd", "terraform" |
| artifactId | M | 1 | String | Reference to a DeploymentArtifact. It can refer to URI or file path. |
| lifecycleParameters | M | 0..N | String | The list of parameters that can be overridden at deployment time (e.g., the list of parameters in the values.yaml which can be overridden at deployment time) |
| deploymentOrder | M | 0..1 | Integer | Specifies the deployment stage that the DeploymentArtifact belongs to. A lower value specifies that the DeploymentArtifact belongs to an earlier deployment stage, i.e. needs to be installed prior to DeploymentArtifact with higher deploymentOrder values. If not specified, the deployment of the DeploymentArtifact can be done in arbitrary order and decided bu the orchestrator. |

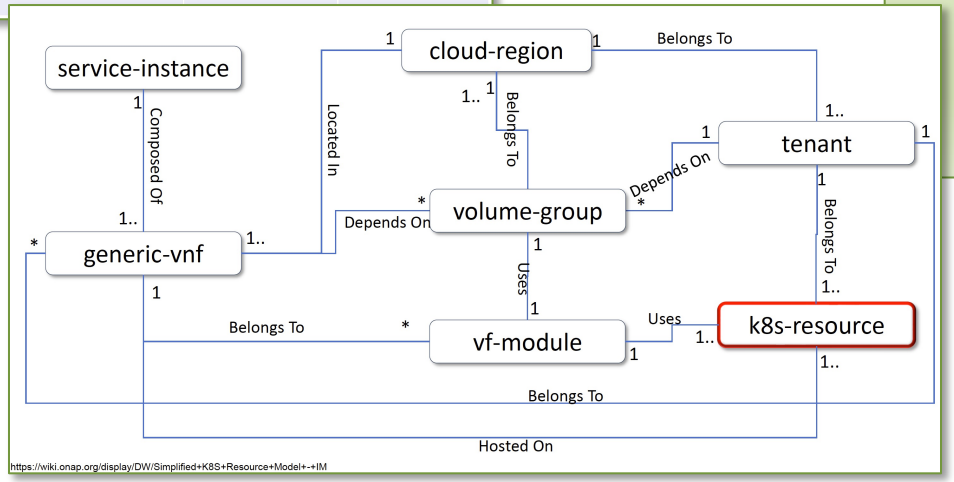
based on the list of customizable parameters that is defined in the ASD>deploymentItem>lifecycleParameters, SO client can formulate the customized parameters in the request

- SO CNF Manager creates new Helm Chart instance values based on input from:
  1. Request Input parameter thru SO, based on the ASD lifecycle Parameter
  2. ASD default
  3. Default Helm Chart Values
- Leveraging generated Helm Chart instance values (file), SO CNF Manager invokes ASD-CNF deployment towards K8S thru SO CNF Adapter
- Use of CBA/CDS for input parameter assignment is a future consideration (plan to collaborate with CNFO)

# AAI Data Model Impact



| Attribute | Type | Mandatory |
|---|---|---|
| id | UUID | Yes (PK) |
| name | String | Yes |
| group | String | Yes |
| version | String | Yes |
| kind | String | Yes |
| labels | List of strings | No |
| namespace | String | Yes |
| selflink | URI | Yes |




- ASD-CNF Orchestration leverages AAI models which are proposed/developed by CNFO.
- Accordingly, this PoC will share the same limitation and enhancement path with CNFO.
  - Collaboration effort between ASD PoC and CNFO will be arranged
- SDC transforms ASD models to SDC internal models to reduce impact to ONAP components, including AAI.

<source: CNFO presentation: https://wiki.lfnetworking.org/display/LN/2022-01-11+-+ONAP%3A+CNF+Orchestration+Tutorial >

# ASD PoC Scope Summary



- ASD App Package onboarding to SDC
  - Manage large-size ASD App package in SDC
  - Mapping between ASD and ONAP Resource VF
- ASD Package distribution
  - Store ASD and ASD' package to the runtime Catalog Repository
  - Store Helm Chart(s) to the runtime Helm Artifact Repository
  - Store Image(s) to the runtime Image Artifact Repository
- Deploy/Instantiate ASD package to K8S
  - Support SO Model-Driven orchestration
  - Delegate ASD/HelmChart handling (instantiation) to SO CNF Manager
  - Input parameters handling based request input and Helm Chart default
- Future Considerations
  - Day 2 Configuration – future consideration
  - ASD CNF Upgrade Support – future consideration, but plan to collaborate with CNFO
  - Policy & CLAMP support – future consideration
  - Integration with external K8S orchestrators (e.g., EMCO) – future consideration
  - leverage with CBA/CDS for assignment – future consideration
  - On-Demand healthcheck could be supported by leveraging SO REST API (like CNFO)