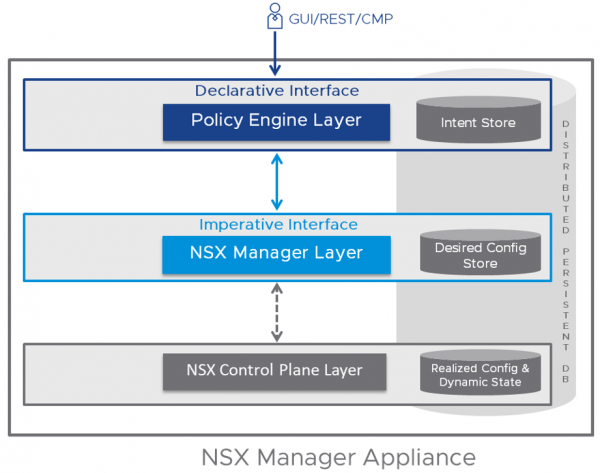
Problem statement:

Support NSX-T policy API in ACI integration with NSX-T.

Background:  
NSX-T release 2.4 introduced a new policy api. It uses a declarative API model and can be used to create the entire intent in one go without caring about ordering or having to make multiple API calls.

From NSX-T 2.4 release, users interact with the NSX Manager using the Simplified UI. The traditional objects will be available under the Advanced UI.

The objects created in NSX-T 2.3 and before will be exposed in the Advanced UI section. There is a very clear separation of function that is defined between objects created with Policy APIs vs objects created using traditional MP APIs:



Policy api URI has the prefix: /policy/api/v1/

As of now, we have discovered that LS needs to be supported. We will add support for more objects if we find during the feature.

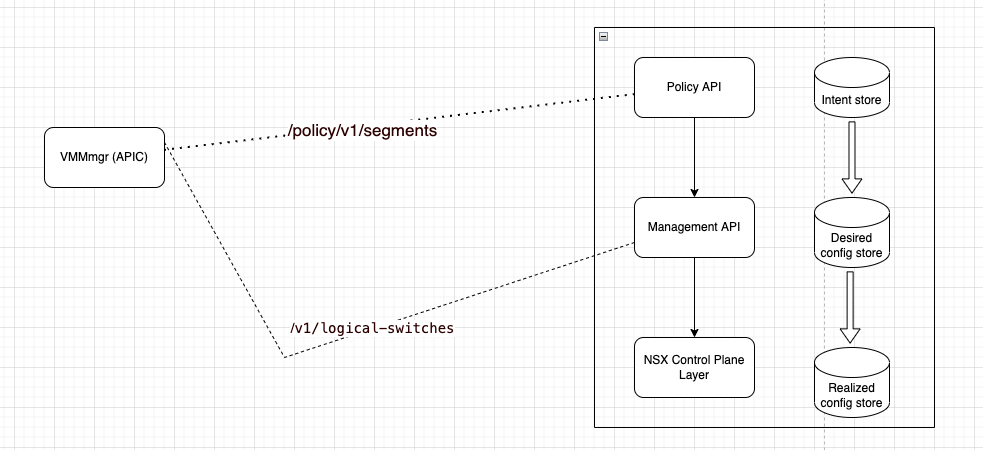
**Policy API introduces new names for following contructs:**

|  |  |  |
| --- | --- | --- |
| Management API | Policy API | ACI equivalent |
| Logical switch | segment | Port group |
| T1 logical router | Tier 1 Gateway | NA |
| T0 logical router | Tier 0 Gateway | NA |
| NSgroups, IP sets, MAC sets | Group | NA |
| Firewall section | Security policy | NA |
| Firewall rule | Rule | NA |
| Edge firewall | Gateway firewall | NA |

To support Logical switch in mgmt API, we need to support segment in policy api.

Following considerations are present to support this new object:

1. There will be knob at vmmDomain level which identifies API mode (policy (new) or mgmt.(old))
2. Once user selects “policy” as new api mode, policy api will be used to for all CRUD operations to NSX-T manager. Existing objects from mgmt mode will be transformed to policy mode. (To verify if we can creare a object using policy api and associate to existing object. For eg, if an LS is already present, can we create a new segment and associate to existing LS, or we need to delete existing LS and create fresh segment?)
3. Downgrade: if user downgrades to unsupported version, retain the policies created with the new policy. (why ?? ‘cos the policies are present in VC DB and we do not want to delete those policies. In the Lower version, mgmt. API will be used to perform CRUD operation. (Verify with setup if we can delete and update objects created using policy API, from mgmt API)
4. Upgrade: What happens when the user upgrades again? If the New mode is selected, apic will read the LS construct. Or it will create a new LS.  Same as #2.
5. User cannot switch back and forth. (we should block the operation new to old. Old to new is allowed with a warning “can not revert back .. “  )



**REST API in POLICY API:**

Policy APIs areregular REST APIs. They support the traditional GET/PUT/DELETE calls. Note that there are no POST calls. PATCH API calls are supported. (parentless config)

In vmmmgr code, POST is used at following places:

1. Creating transport zone. (maps to hvs:LNode.)
2. Creating logical switch. (maps to comp:EpPD)
3. Setting up NSX manager connection.

We need to convert these POST calls to PUT/PATCH.

Current code is using POST to create and GET+PUT to update.

We can directly use

1. PUT to create objects.
2. PATCH to update objects. (PATCH doesn not need \_revision property to be provided). We can also use PUT, but update via PUT involves an additional GET call to get latest \_revision of the object. Using PATCH saves an additional RPC call.

Todo: Verify PUT or PATCH to create and update segments and establish connection.

**Setup bringup:**

NSX-t manager is run as a virtual appliance in an esxi in VCenter environment. OVF file can be deployed to create the virtual appliance.

TBD: License and OVF file.

**Future support (**NSX-T 3.2 Promotion tool **):**

**NSX Manager to NSX Policy Promotion Tool** – Provides ability to promote existing configuration from NSX Manager to NSX Policy without data path disruption or deletion/recreation of existing objects.