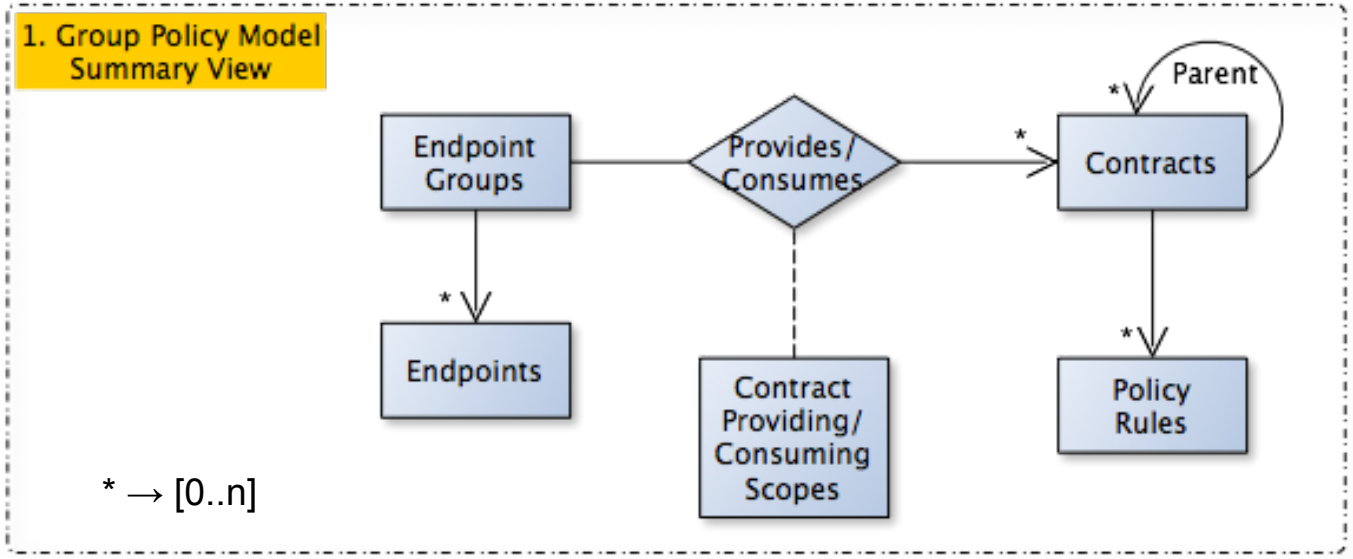
OpenStack
Group-based Policy

Policy Model (App Admin context)



Recent Terminology changes:

Old model	**Current model**
Contract	Policy Rule Set
Endpoint Group (EPG)	Policy Target Group (PTG)
Endpoint (EP)	Policy Target (PT)

Simple CLI-based example
Use Case: Define access to the Web Server Tier from the Outside Network

Create Classifier

```
neutron policy-classifier-create Insecure-Web-Access --port 80 --protocol TCP --direction IN
```

Create Policy Rule Set using the Classifier

```
neutron policy-rule-set-create Web-Server-Contract --policy-classifier Insecure-Web-Access --action ALLOW
```

Create PTG providing the Contract

```
neutron policy-target-group-create Web-Server-EPG --provides-policy-rule-set Web-Server-Contract
```

Create Policy Target in EPG

```
neutron policy-target-create --policy-target-group Web-Server-EPG
```

Launch Web Server VM using Policy Target in PTG

```
nova boot --image cirros --flavor m1.nano --nic port-id=<EP-NAME> Web-Server
```

Specify connectivity of Outside world VMs to Web Server

```
neutron policy-target-group-create Outside-EPG --consumes-policy-rule-set Web-Server-Contract
```

Policy_rules: Individual rules used to define the communication criteria between PTGs. Contains classifier/filter and action.

attribute	Type	Required	CRUD	Default	Notes
id	uuid	NA	R	generated	
name	string	No	CRU	None	
description	string	No	CRU	None	
tenant_id	uuid	NA	R	from Auth token	
policy_rule_set_filter	uuid	Yes	CRU	None	
policy_classifier	uuid	Yes	CRU	None	
policy_actions	list	Yes	CRU	None	

Policy Target Groups (PTG, formerly EPG): It is a collection of endpoints.

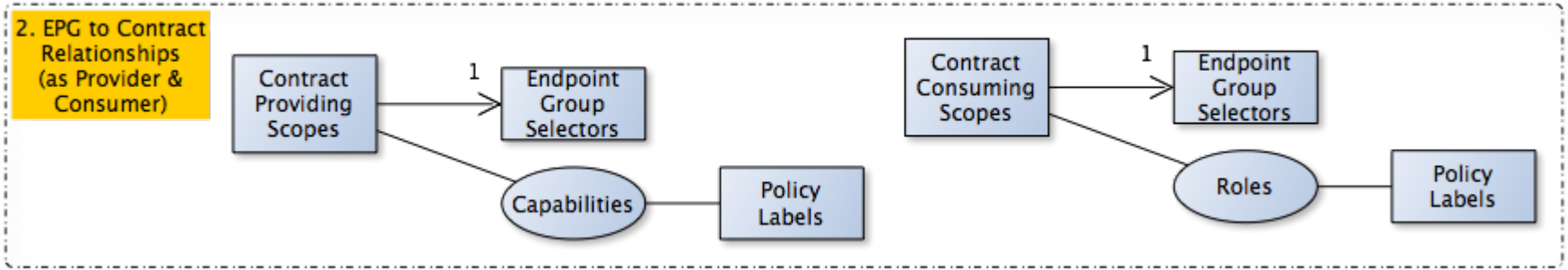
attribute	Type	Required	CRUD	Default	Notes
id	uuid	NA	R	generated	
name	string	No	CRU	None	
description	string	No	CRU	None	
tenant_id	uuid	NA	R	from Auth token	
endpoints	list	No	R	Empty list	List of endpoint uuids
provided_policy_rule_sets	list	No	CRU	Empty list	List of dict, where each dict is {“policy_rule_set_id”: <policy_rule_set_uuid>, “policy_rule_set_scope”: <policy_rule_set_provided_scope_uuid> or None}
consumed_policy_rule_sets	list	No	CRU	Empty list	List of dict, where each dict is {“policy_rule_set_id”: <policy_rule_set_uuid>, “policy_rule_set_scope”: <policy_rule_set_consumed_scope_uuid> or None}
policy_labels	list	No	CRU	Empty list	list of policy_label uuids

Policy Rule Sets (PRS, formerly Contracts): It defines how the application services provided by an PTG can be accessed. In effect it specifies how an PTG communicates with other PTGs. A PRS consists of Policy Rules.

attribute	Type	Required	CRUD	Default	Notes
id	uuid	NA	R	generated	
name	string	No	CRU	None	
description	string	No	CRU	None	
tenant_id	uuid	NA	R	from Auth token	
child_policy_rule_sets	list	Yes	CRU	None	list of policy_rule_set uuids
policy_rules	list	Yes	CRU	None	List of policy_rule UUIDs

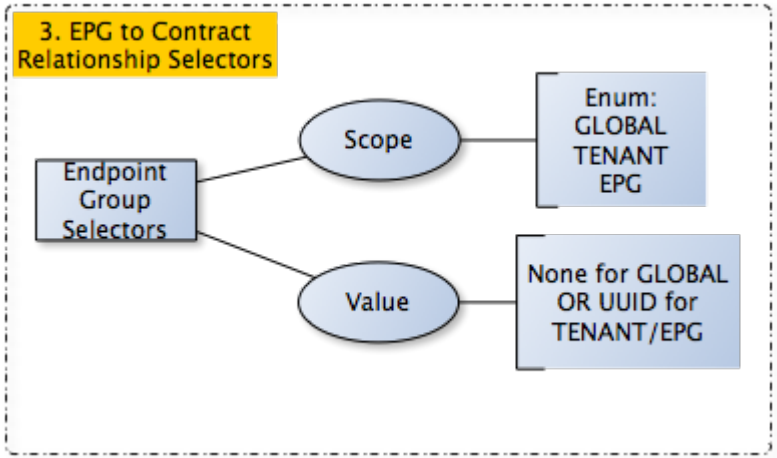
PTG to Policy Rule Set Relationship

- An PTG can *provide* or *consume* one or more PRSs.
- Each *provide* or *consume* relationship can be qualified with **optional** constraints. These constraints can be expressed in the PRS Providing Scopes and PRS Consuming Scopes resources as shown below.



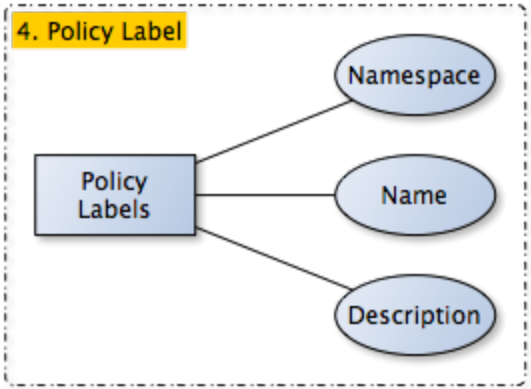
- Policy Target Group Selectors allow for providing constraints around choosing the matching provider or consumer PTGs (see following slide).
- Capabilities here correspond to the Policy Labels defined in the PRS Filters resource (see Policy Rules slide). Specifying capabilities in the “PRS Providing Scope” relationship allows an application deployer to indicate what subset of a particular PRS is being provided by the PTG participating in the relationship with the PRS. If no capabilities are specified, the default is the entire PRS.
- Roles here correspond to the Policy Labels defined in the PRS Filters resource (see Policy Rules slide). Specifying roles in the “PRS Consuming Scope” relationship allows an application deployer to indicate what subset of a particular PRS is being consumed by the PTG participating in the relationship. If no roles are specified, the default is interpreted as the intent to consume the entire PRS.

PTG selectors (optional)

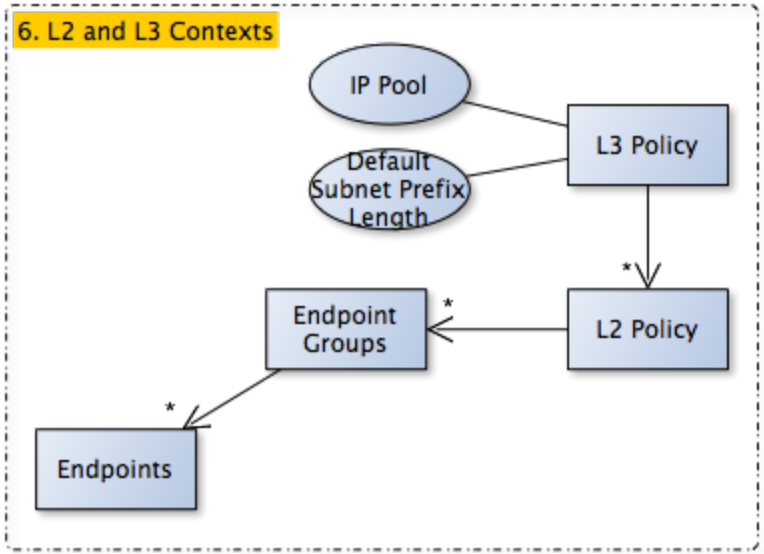


- Scope defines whether to match PTGs in the one of global, tenant or specific PTG scope. This is used by the policy engine to choose the matching PTG when multiple PTGs provide the same PRS (or to specify which PTGs can consume a PRS).
- Default scope is tenant.

Policy Labels (optional)



Policy Model (Network Admin context)



L2 Policy: Used to define a L2 boundary and impose additional constraints within that L2 boundary. (Additional constraints could be of the type “QUARANTINE”, “NO_BROADCAST”, etc.)

attribute	**Type**	**Required**	**CRUD**	**Default**	**Notes**
id	uuid	NA	R	generated	
name	string	No	CRU	None	
description	string	No	CRU	None	
tenant_id	uuid	NA	R	from Auth token	
policy_target_groups	list	NA	R	empty	List of PTG uuids (separate operations will be provided to add or remove from this list)

L3 Policy: Used to define a non-overlapping IP address space.

attribute	Type	Required	CRUD	Default	Notes
id	uuid	NA		generated	
name	string	No	CRU	None	
description	string	No	CRU	None	
tenant_id	uuid	NA		from Auth token	
ip_pool	string	No	CR	172.16.0.0/12	defines the IP address space used for creating subnets
default_subnet_prefix_length	string/int	No	CRU	26	default address prefix length for the subnet created automatically
l2_policies	uuid	NA	R	empty	List of l2_policy uuids

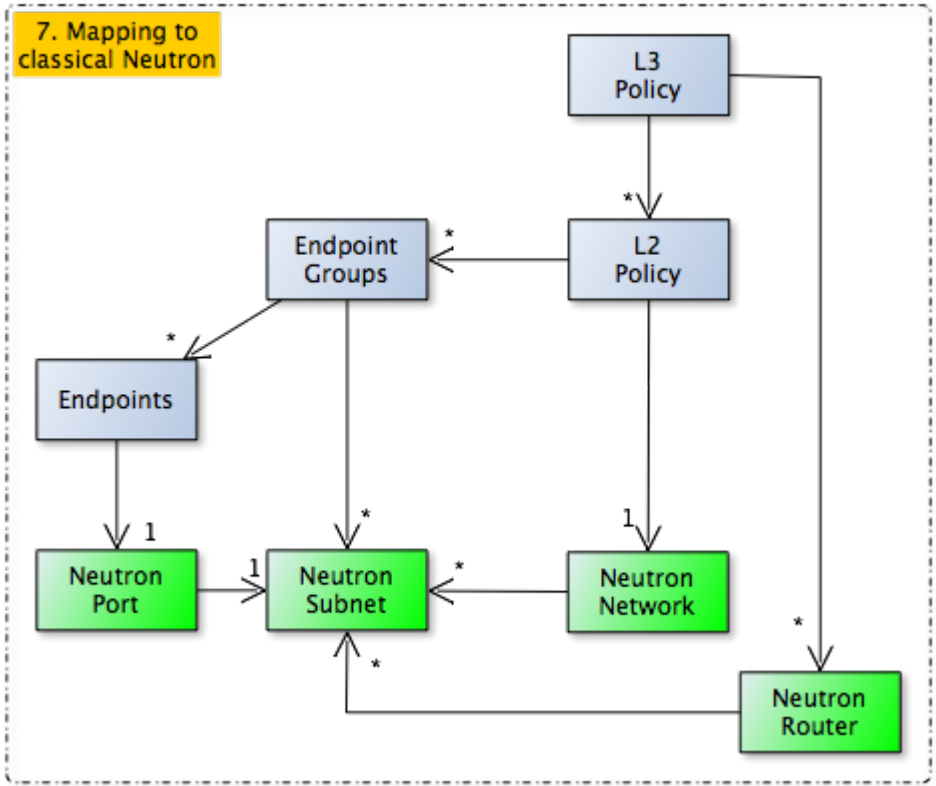
Constraints introduced by Infra admin

- An infrastructure admin can introduce constraints in the communication between two PTGs without the knowledge of those PTGs.
- Example: An infra admin creates a policy to redirect all traffic between the Web PTG and the App PTG via a firewall. This change is “orchestrated” as a side effect (and without any app deployer visible impact).
- Infra constraints can be achieved by PRS hierarchical composition. In the above example, the App PRS can be made a child of an infra admin created PRS. The constraint (redirect to the firewall) is introduced in the parent PRS.

Redirect Action

- The composed PRS can have multiple REDIRECT Actions
- Policy Rules with REDIRECT actions will be processed after all other Policy Rules are processed
- The Policy Rule with a REDIRECT Action in the parent PRS will be given priority over other Policy Rules with a REDIRECT Action in the child PRS. Beyond this constraint, the order of Policy Rules with REDIRECT Actions is undefined.
- If a Policy Rule has multiple Actions, including REDIRECT, the REDIRECT will be processed last. There is no ordering on the other Actions.

Mapping to Traditional Neutron



* → [0..n]

Neutron existing non-policy constructs

Policy Target: Extended Attribute for legacy Neutron mapping.

attribute	**Type**	**Required**	**CRUD**	**Default**	**Notes**
port (reference to Neutron port)	uuid	No	CR	None	

Policy Target Groups (PTG): Extended Attribute for legacy Neutron mapping.

attribute	Type	Required	CRUD	Default	Notes
subnets (reference to Neutron subnet)	list	No	CRU	empty	reference to list of neutron subnet uuids

L2 Policy: Extended Attribute for legacy Neutron mapping..

attribute	**Type**	**Required**	**CRUD**	**Default**	**Notes**
network (reference to Neutron network)	string	No	CR	None	Reference to a Neutron network

L3 Policy: Extended Attribute for legacy Neutron mapping..

attribute	**Type**	**Required**	**CRUD**	**Default**	**Notes**
routers (reference Neutron router)	list	No	CRU	empty list	List of uuids referencing Neutron routers present in this routing domain

Service or Chain Representation

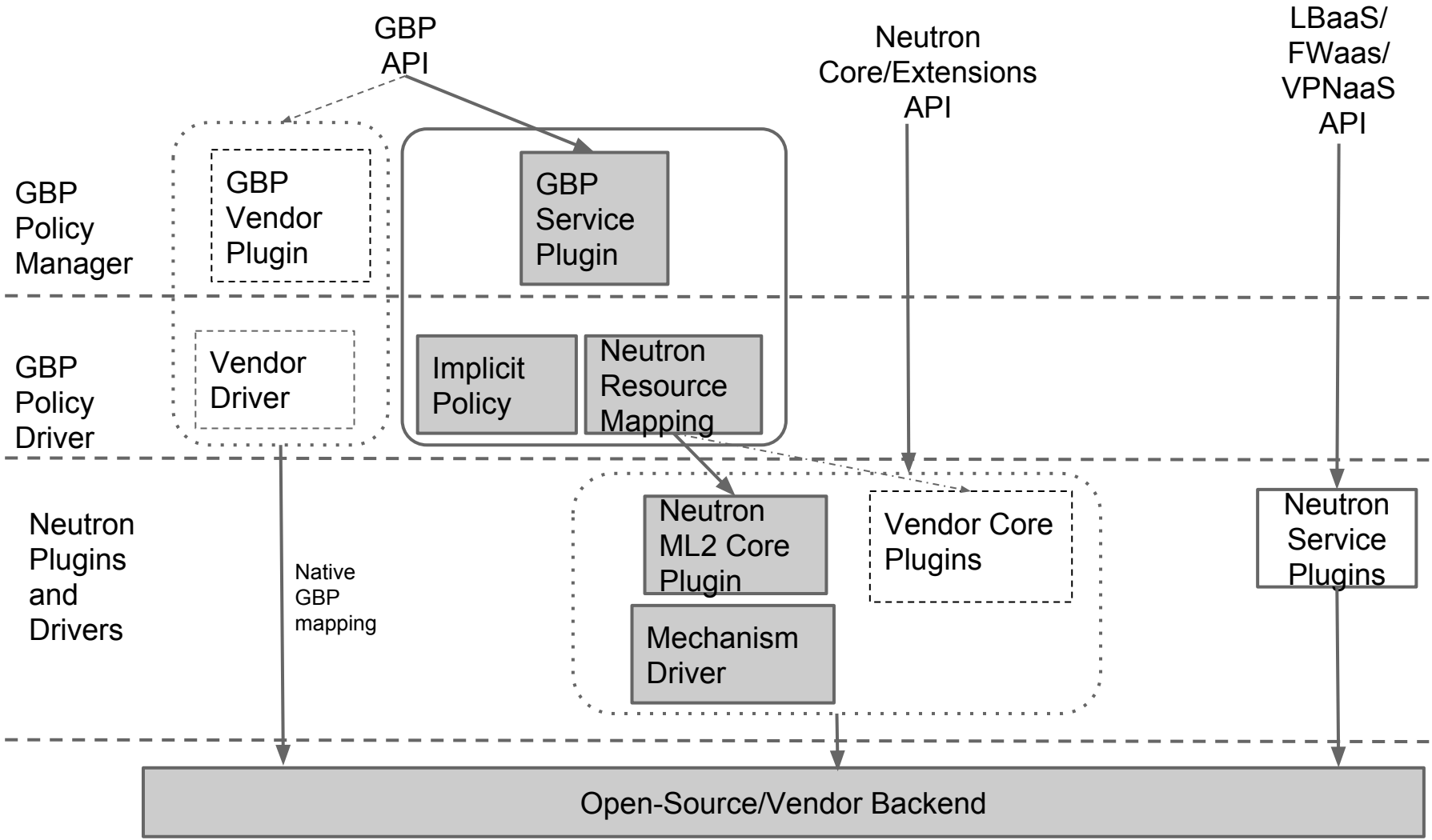
Each network service is treated as a function with n input and m output endpoints.

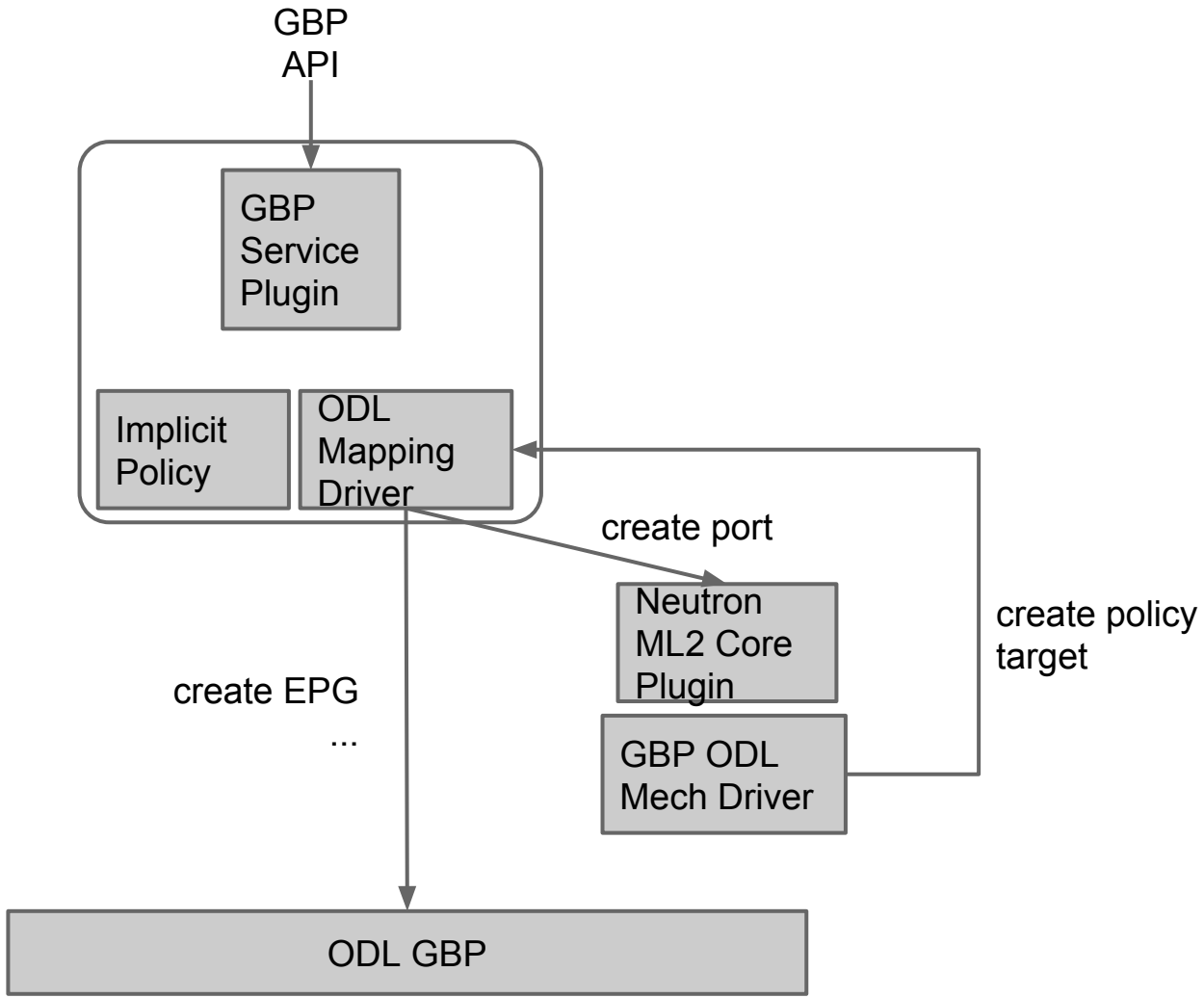
A network service chain is also treated as a single service.

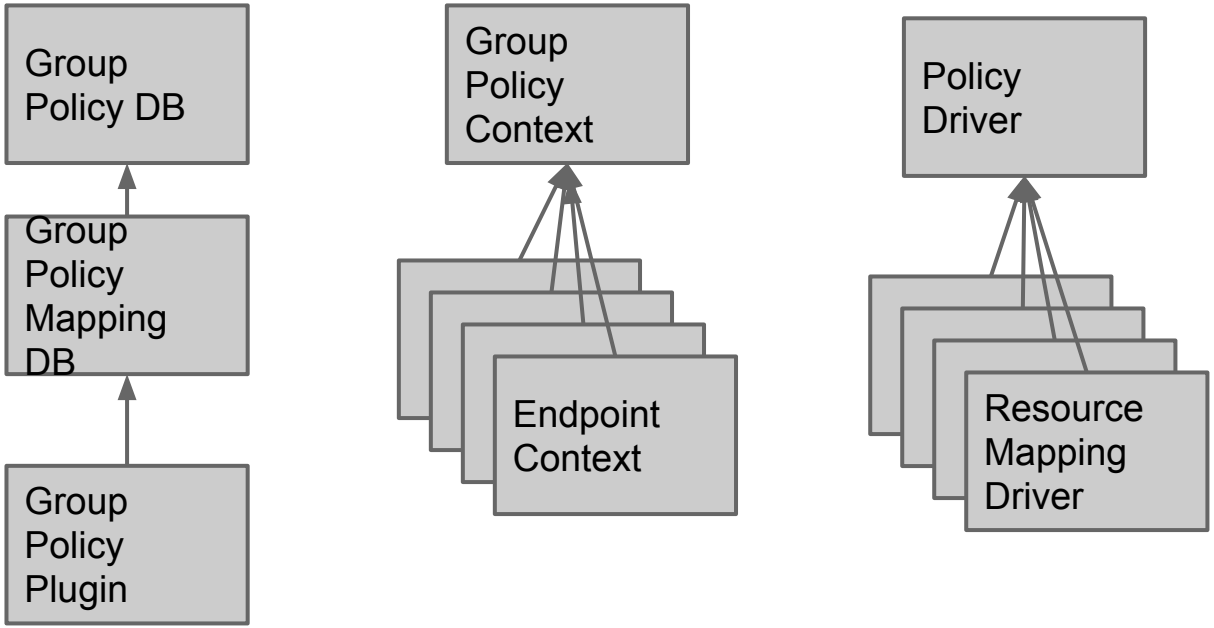
Open Issues

- Policy Label RBAC
-

Implementation


GBP API
Neutron Core/Extensions API
LBaaS/ FWaas/ VPNaaS API
GBP Policy Manager
GBP Vendor Plugin
GBP Service Plugin
GBP Policy Driver
Vendor Driver
Implicit Policy
Neutron Resource Mapping
Neutron Plugins and Drivers
Native GBP mapping
Neutron ML2 Core Plugin
Vendor Core Plugins
Neutron Service Plugins
Mechanism Driver
Open-Source/Vendor Backend


GBP
API
GBP
Service
Plugin
Implicit
Policy
ODL
Mapping
Driver
create port
Neutron
ML2 Core
Plugin
create policy
target
create EPG
...
GBP ODL
Mech Driver
ODL GBP


Group
Policy DB
Group
Policy
Mapping
DB
Group
Policy
Plugin
Group
Policy
Context
Endpoint
Context
Policy
Driver
Resource
Mapping
Driver