Policy to Legacy mapping for PoC

1. Create EPG
    a. Check if a rendering policy is present for this tenant. In PoC implementation default rendering policy is always used. Default rendering policy is defined in a configuration. This default rendering configuration will define EPG to BD mapping, BD to Neutron Network mapping, BD to RD mapping, Router creation policy, and IP address space for the RD. See below for details.
    b. Create RD with name ‘default’ if it doesn’t exist
    c. Create Neutron router, associate Router with RD
    d. Create BD in RD, with name based on EPG name, associate EPG with BD.
    e. Create Neutron Network with name based on BD name, associate Neutron Network with BD
    f. Create Subnet (default CIDR size for subnet is config driven, and is picked from the RD’s configured IP address space)
    g. Add Neutron router interface on subnet

2. Create a contract => No-op
3. An EPG provides a contract => No-op
4. An EPG consumes a specific contract =>
    a. Create security groups based on labels and addresses (if needed)
    b. Create Firewall and configure it (if needed)
    c. Add QoS params (if needed)
    d. Update routing (if needed)
5. A new VM is added to an EPG => No-op (network already exists)
6. A new subject is added => Same as (4a)
7. A new redirect is added => Same as (4b)

Implicit use case with immediate mapping

1. User creates an EPG without specifying BD or neutron_subnet_id
    a. no BD specified, so implicitly create a new BD with name based on EPG name
        i. no RD specified, so use tenant’s default RD
            1. if RD named ‘default’ doesn’t already exist for tenant, implicitly create it, using rendering policy’s default ip_supernet from config
            2. create neutron router for the RD
        ii. create neutron network for the BD
    b. create neutron subnet for the EPG
2. User creates an EP specifying the EPG from step 1, without specifying neutron_port_id

    a. create neutron port for the EP with EPG's subnet and BD's network
3. User creates a VM specifying --nic as neutron port from step 2

Split use case with deferred mapping

Rather than create neutron resources immediately as GP resources are created, this could be deferred until the neutron resources are actually needed (when an EP needs a port). This might be useful for a workflow separating application and network concerns, such as:

1. application admin creates various EPGs and their associated policy
2. network admin creates/choses RDs and BDs for the application
3. network admin associates EPGs from 1 with BDs from 2
4. application admin creates EPs in the EPGs

The deferred mapping would work as follows:

1. App admin creates EPG without specifying BD or neutron_subnet_id
    a. no BD specified, so implicitly create a new BD with name based on EPG name
        i. no RD specified, so use tenant's default RD
            1. if RD named 'default' doesn't already exist for tenant, implicitly create it, using rendering policy's default ip_supernet from config
2. Net admin creates RD
3. Net admin creates BD
4. Net admin associates BD from step 3 with EPG from step 1
5. App admin creates an EP specifying the EPG from step 1, without specifying neutron_port_id
    a. EPG doesn't have a subnet, so create one
        i. BD doesn't have a network, so create one
            1. create neutron network for BD
        ii. create neutron subnet for EPG
    b. create neutron port for the EP with EPG's subnet and BD's network
6. User creates a VM specifying --nic as neutron port from step 2

Explicit use case

1. User creates RD
2. User creates BD specifying the RD from step 1, without specifying neutron_network_id
3. User creates EPG specifying the BD from step 2, without specifying the neutron_subnet_id
4. User creates EP specifying the EPG from from step 1, without specifying the neutron_port_id

5. User creates VM specifying --nic as neutron port from step 4

Explicit use case specifying neutron resources

1. User creates RD
2. User creates BD specifying the RD from step 1, specifying neutron_network_id
3. User creates EPG specifying the BD from step 2, specifying the neutron_subnet_id
    a. Nog possible with the API as specified
4. User creates EP specifying the EPG from from step 1, specifying the neutron_port_id
5. User creates VM specifying --nic as neutron port from step 4

Mapping relationships

- An EPG maps to one or more subnets.
    - This is so SG and FW rules that apply to the EPG just match the EPG's subnet instead of the IP address of each EP in the EPG.
        - Do we ever share the same subnet across multiple EPGs?
    - If no subnet is passed in when creating EPG, a new subnet is created.
        - Do we do this immediately, or defer it until we need the subnet (when port for EP must be created)?
    - If subnet is passed in when creating EPG, raise exception if not valid.
        - Validate that subnet is on BD's network
        - Validate that no existing EPG maps to same subnet
- An EP maps to a port.
    - If no port is passed in when creating EP, a new port is created.
        - Port created in EPG's BD's network.
        - If there are multiple subnets in EPG's BD's network, need to control from which subnet the port's IP address is allocated.
    - If port is passed in when creating EP, raise exception if not valid.
        - Validate that port belongs to EPG's BD's network
        - Validate that port's IP is in one of the EPG's subnets
        - Validate that no existing EP maps to same port
- A BD maps to a network.
    - If no network is passed in when creating BD, a new network is created.
        - Do we do this immediately, or defer until we need the network?
    - If port is passed in when creating BD, raise exception if not valid.
        - Validate that no existing BD maps to the same network

Mapped resource lifecycle

- Does deleting the group policy resource result in the corresponding neutron resource being deleted?
    - Does mapping need to keep track of whether it created the neutron resource or it was passed in, and delete it only if it created it?
        - Do we need to add this to the DB models?
        - Can we delete the neutron resource iff its tenant_id is a special internal tenant used by the mapping?

Rendering policy

- Come from configuration
- Includes, with defaults:
    - default ip_supernet for RDs - 10.0.0.0/8
    - subnet size for EPGs - /24
    - name for default RDs - ‘default’

Associating VM’s vNIC with EP

Options:

1. Create EP explicitly and pass EP’s port’s ID to nova
    a. currently works with CLI’s --nic option
    b. does horizon support specifying port?
2. Create EP explicitly and pass EP’s ID to nova
    a. need to extend nova’s --nic option to take EP ID
    b. nova uses neutron API to get port’s ID from EP
3. Pass EPG’s ID to nova
    a. need to extend nova’s --nic option to take EPG ID
    b. nova uses neutron API to create new EP for the VM and get the EP’s port’s ID
4. Pass EPG’s BD’s network’s ID to nova
    a. currently works with CLI’s --nic option
    b. currently works with horizon
    c. requires intercepting neutron create_port API to wrap new port in an EP