Quantum L3 Model & API

Table of Contents

Quantum L3 Model & API

1 Introduction

Today's Quantum Service implements Virtual Network (L2) abstractions. Using Quantum's RESTful API, clients can create virtual networks, ports on the virtual networks, and attach VM VIFs to these ports. However, the Quantum service lacks knowledge of L3 constructs like Subnets, and the ability to configure routing between these Subnets. To overcome this limitation, and to in general, incorporate L3 semantics, this document proposes a Quantum L3 model of resources, and a specification for implementing the model.

2 Glossary

Entity	A generic term for any piece of hardware or software desiring to connect to the network services provided by Quantum. An entity may make use of Quantum by implementing a VIF.
L3-network	A network formed by the interconnection of one or more Subnets belonging to one tenant.
Route-table	A Route-table is a tenant instantiated resource that provides operations to manipulate routing entries.
Target	A Service Provider published identifier, or a tenant owned resource ID used within the Route-table to denote a gateway entity.
Subnet	A Subnet is a logical grouping of connected network devices that share a contiguous range of IP address numbers.
Association	The relationship between a Subnet and a Route-table.

3 Theory Of Operation

Quantum abstracts the physical implementation of the network, allowing plugins to configure and manage physical and virtual resources. Quantum is a standalone service, in that it requires no other project within OpenStack to function correctly. Current implementation of Quantum provides for L2 network abstraction. The proposal in this document extends Quantum service to provide L3 abstractions. These L3 abstractions will be exposed as a separate API, but part of the same Quantum service (base Quantum URI remains the same).

Further, the Quantum L3 abstractions are agnostic of the entities that realize them and use them. The L3 abstractions can be realized by a separate L3 plugin. Thus, it would be possible to deploy different combinations of L2 and L3 plugins within the same installation (even simultaneously enabling plugins from different vendors). The plugin in

turn would configure one or more physical or virtual resources (or a combination of both). On the client side, we anticipate that Nova will be the predominant consumer; however other entities can consume the L3 services just as well.

A tenant creates one or more Subnets using the Quantum L3 API (defined in Section 7). VMs on any Subnet have access to other VMs on the same Subnet. To provide access to the VMs outside the Subnet, Routes need to be created indicating the desired connectivity.

Routes are part of a Route-table. Each Route entry consists of Source, Destination, and Target IDs. For instance, if a tenant wants to set up connectivity such that a Subnet A, needs to have access to the Internet, he would perform the following steps:

1. Create Subnet A
2. Create Route-table, RT-A
3. Associate Subnet A with Route-table RT-A
4. Create a Route in the Route-table:
   <Source: Subnet A, Destination: 0.0.0.0, Target: Public>

The Route entry in step 4 indicates that traffic from any VMs on Subnet A, which is not on that Subnet needs to go the Public network.

Note: Route entries and Route-tables referred here are logical abstractions, and may result in the creation of one more route entries and/or policies in the underlying L3 infrastructure.

The Service Provider in response to these API calls, will set up the necessary infrastructure to achieve the above connectivity. Exactly how this setup is achieved will be specific to the particular plugin which implements the L3 abstractions. In the above case, the plugin could instantiate a virtual or physical gateway entity to route the traffic to and from the Subnet A to the Public network. It should be noted that in this model the tenant does not directly manage the gateway device, it's instantiated and managed by the Service Provider, the tenant merely references it by using the "Public" target.

4 Concepts

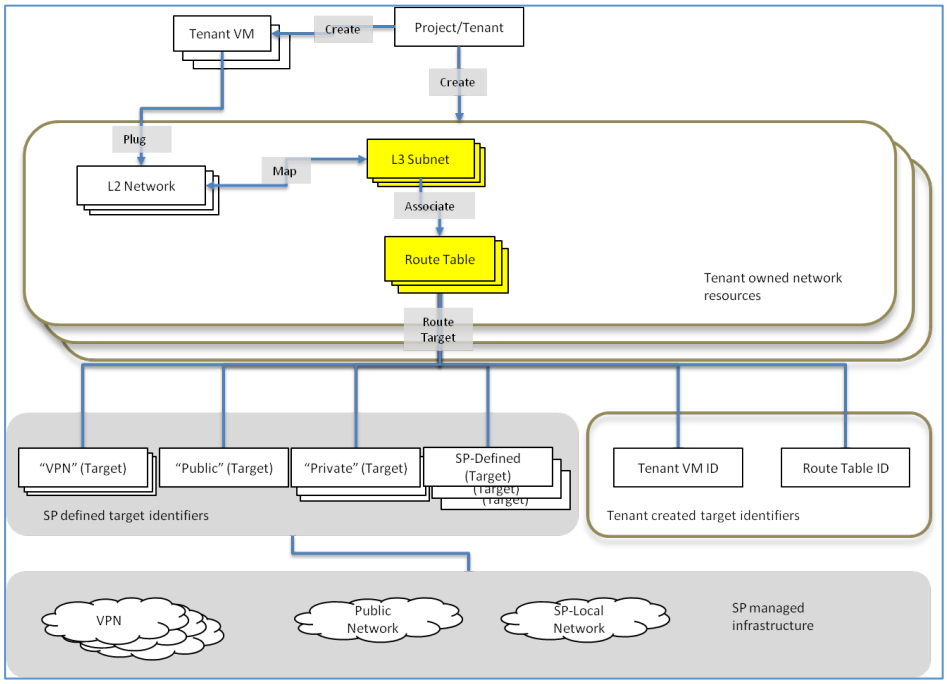

Figure 1 L3 Model Schematic

4.1 Route-table

A Route-table is a tenant instantiated resource that provides operations to manipulate routing entries. One can conceptually think of a Route-table mapping to a logical routing entity.

There is no implicit routing to/from subnets; routing will happen between entities only if a routing entry exists in the Route-table.

Each entry in the Route-table assumes the following format:

Source	Destination	Target
Source ID	Destination ID	Target ID

A Subnet resource has to be first created and associated with a Route-table (this is like physically connecting to an interface on the router).

Source ID: Subnet ID or CIDR or "*"

Destination ID: Subnet ID or CIDR

Target ID: Either a SP published identifier, or a tenant owned resource ID. The SP should be able to resolve the Target ID has to a single endpoint (e.g. IP address). In most cases the entity represented by the Target ID is a gateway device.

4.2 Targets

SP published Target Identifiers: A set of well-known identifiers (common across SPs) and also specific identifiers published by a particular SP.

Well-known Target identifiers:

"Private" – Indicates the packets from a particular Source IP address range going to the corresponding Destination IP address range should be routed on the private/local network created for this tenant.

(Note: The scope of "Private" is for that particular Route-table, i.e. routing will take place between all subnets associated with that Route-table.)

"Public" - Indicates the packets from a particular Source IP address range going to the corresponding Destination IP address range should be routed to the SP's network. On seeing this Target, the SP will instantiate a Gateway (if one does not already exist) to the public network and direct the packets to this Gateway. The Tenant will point to this Target when it is required to direct the traffic to the Internet or to services on the SP's shared network.

Other Target identifiers published by SP: These could be services offered by the SP, for instance NAT. The SP publishes an identifier which the tenant can refer to as a Target.

E.g.:
"VPN-A" - Indicates the packets from a particular Source IP address range going to the corresponding Destination IP address range should be routed to the tenant's network. On seeing this Target, the SP will instantiate a VPN-Gateway (if one does not already exist), or consume APIs from another service that provides VPN/Network L2/L3 Extension services abstractions, and direct the packets to that VPN. The Tenant will point to this Target when it is required to direct the traffic to its home network.

Tenant owned resources as Targets: Examples include VM IDs. For these IDs to be valid Targets, the SP has to be able to determine the route to the endpoint represented by such an ID, else the Target is not valid and the operation underlying implementation should return an error. The tenant maintains these tenant IDs.

The tenant might also want to publish certain targets for consumption (for other tenants and projects

4.3 Subnets

IP Address ranges (denoted by CIDR notation). The IP address range is defined by the tenant and each tenant has its own IP address space (i.e. support for overlapping IP address spaces within a single SP's domain).

<To Do: Define the scheme of how IP addresses are handed out to VMs within a subnet. This will help to make it consistent across plugin implementations.>

4.4 Associations

This defines the relationship of a Subnet with a Route-table. Each subnet can be associated with at the most one Route-table. The association operation can be thought of as being the logical equivalent of physically connecting to an interface on a router.

4.5 Entity Relationships

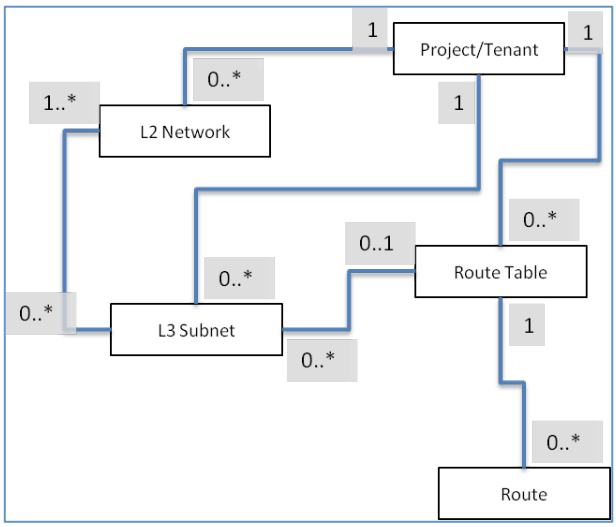

Figure 2 Entity Relationships

5 Use Cases

5.1 Public Subnet

Access to VMs on a Quantum network from the Public subnet via Floating IPs (FIP).
(Note: Here the term “Public” does not actually mean that the subnet has a public CIDR. It implies that VMs on this subnet have public floating IPs associated.)

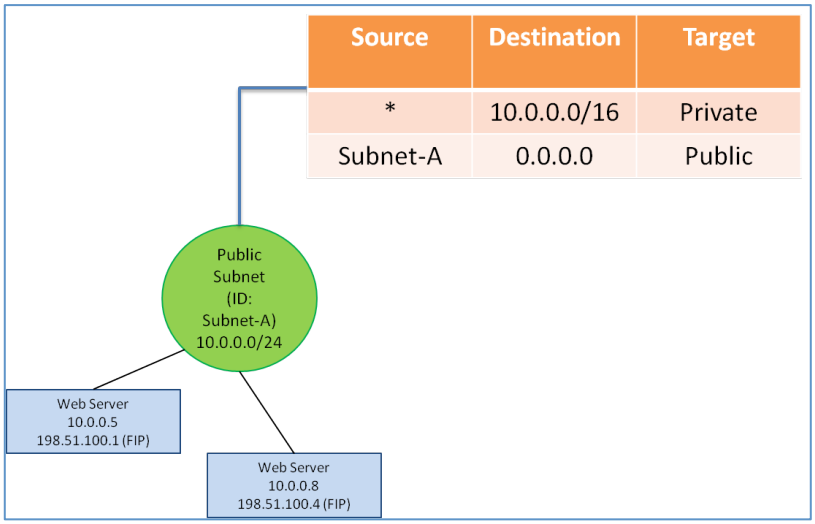

Figure 3 Quantum L3 Route-table with Public Subnet

5.2 Public and Private Subnets

Tenant has both public and private subnets. Access to VMs on a Quantum public subnet from the Public network is via Floating IPs (FIP). The VMs on the private subnet can access the Public network via a NAT instance shown in the example below.

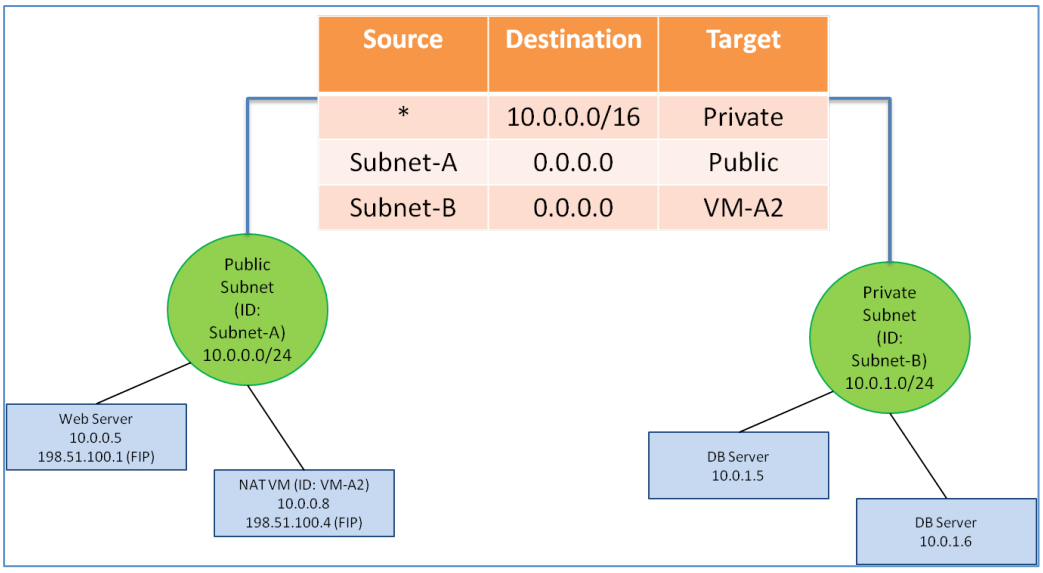

Figure 4 Quantum L3 Route-table with Public and Private Subnets

5.3 Public and Private Subnets, and VPN access

Tenant has both public and private subnets. Access to VMs on a Quantum public subnet from the Public network is via Floating IPs (FIP). The VMs on the private subnet can access the tenant's home network via a VPN target shown in the example below.

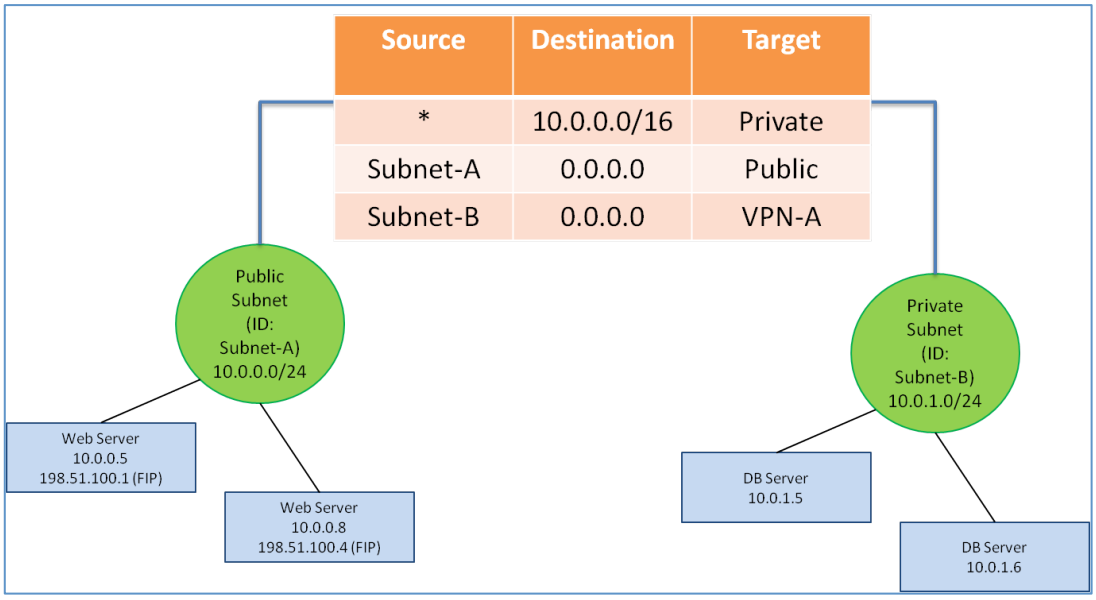

Figure 5 Quantum L3 Route-table with Public and Private Subnets, and VPN access

5.4 Private Subnet and VPN access

Tenant has one or more private subnets. The VMs on the private subnet can access the tenant's home network via a VPN target shown in the example below.

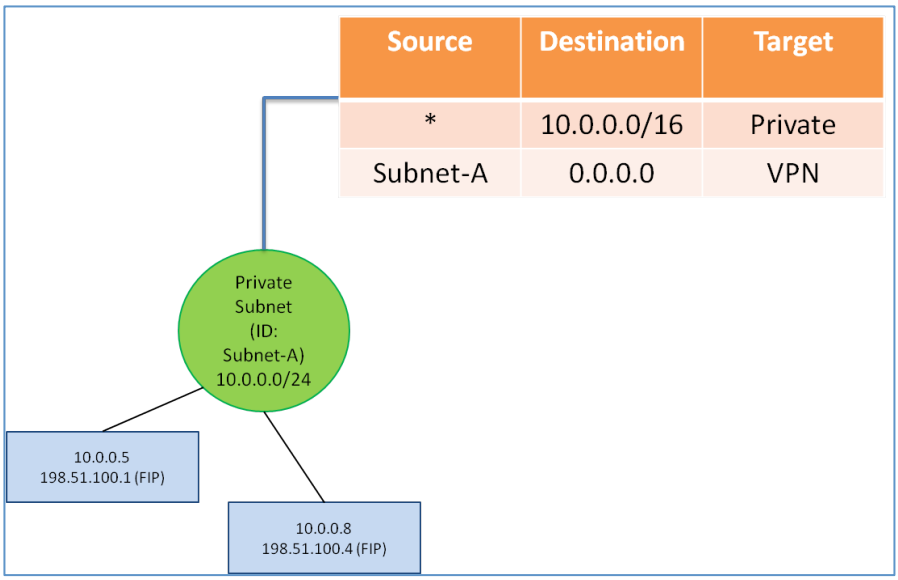

Figure 6 Quantum L3 Route-table with Private Subnet and VPN access

5.5 Three-tier app (one Public subnet and two Private Subnets)

A Web app deployed in a three-tier as shown in the figure below. Only the web tier is accessible from the Public network. The DB tier is isolated from the Web tier.

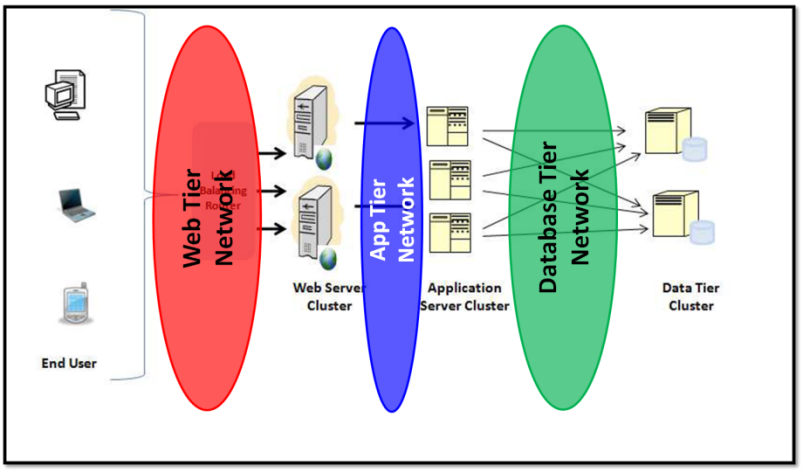

Figure 7 A Web App deployment with Web, App, and DB tiers

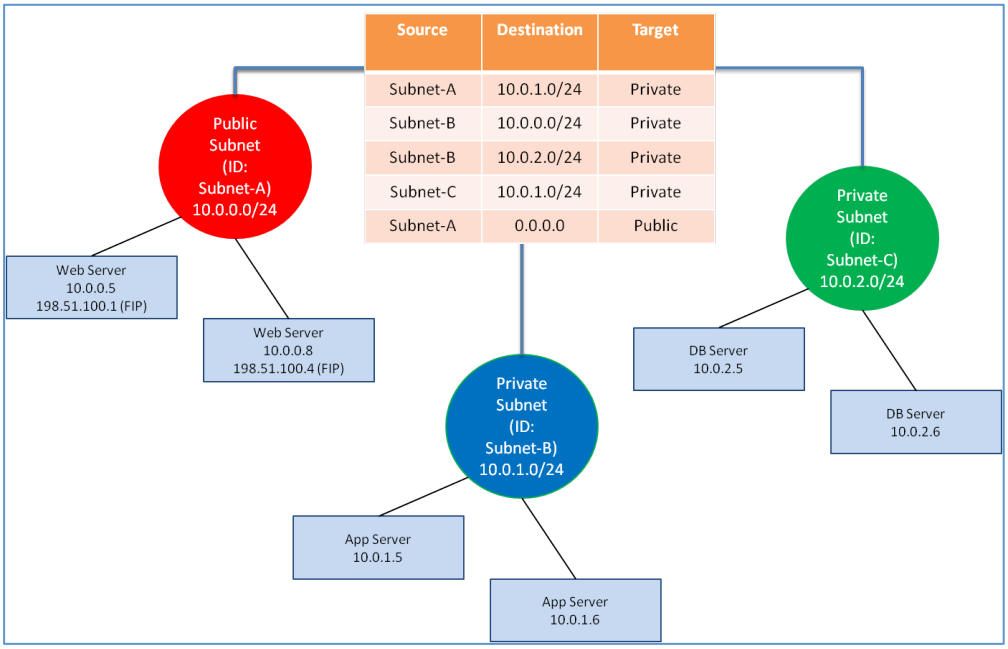

Figure 8 Quantum L3 Route-table for 3-tier app (one Public subnet and two Private Subnets)

5.6 Higher level service overlaying network containers

Consider the example in the context of the above three tier Web app. Each tier can be modeled as a separate network container that is scoped by a Route-table. The "private" scope is per Route-table. The Route-table is one of many resources that could be a part of that container.

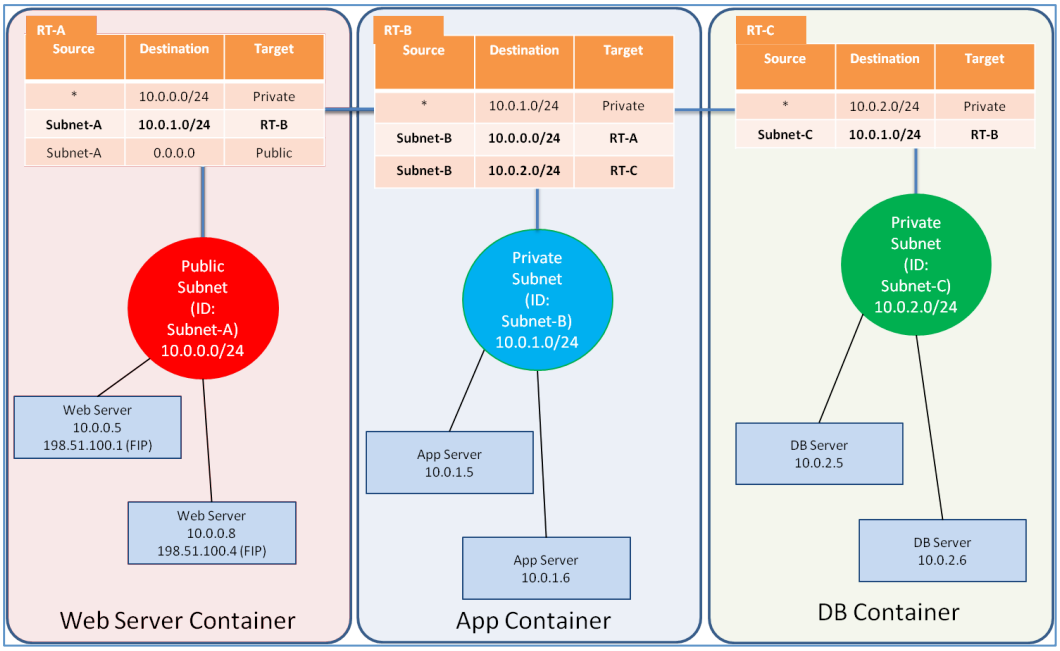

Figure 9 Higher layer service creates a network container for each tier and ties them together using the Route-tables

6 General API Information

This API allows the user to manipulate the resources associated with the above concepts in a consistent and RESTful way.

6.1 General URL structure

Each request to the OpenStack Quantum API must refer to a specific version of the API itself, and it must also identify the tenant for which the request is being sent.

This information is specified in the URI. The URI for each request to the OpenStack Quantum API should be prefixed with the API version identifier and the tenant identifier, as follows:

```
/{Quantum-version}/tenants/{tenant-id}/{Quantum-API-entity}
```

As an example, the following URI represents a request for retrieving all the subnets configured for the tenant "ABC" using the 1.0 API.

```
/v1.0/ABC/subnets
```

6.2 Content-Type Negotiation

The OpenStack Quantum API supports both the JSON and XML data serialization formats. The format for both the request and the response can be specified either using

the Content-Type header, the Accept header or adding an .xml or .json extension to the request URI.

If conflicting formats are specified in headers and/or format extensions, the latter takes precedence. XML is currently the default format for both requests and responses.

For more details see:
http://docs.openstack.org/incubation/openstack-network/developer/quantum-api-1.0/content/Request_Response_Types.html

6.3 API Versioning

The Quantum API uses a URI based versioning scheme. The first element of the URI path contains the target version identifier.

Example Request with URI versioning

```
GET              /v1.0/tenants/tenantX/networks HTTP/1.1
Host             127.0.0.1:9696
Accept
     text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q
=0.8
Content-Type     application/xml; charset=UTF-8
Content-Length   22
```

For more details, see:
http://docs.openstack.org/incubation/openstack-network/developer/quantum-api-1.0/content/Versions.html

6.4 Faults

When an error occurs at request time, the system will return an HTTP error response code denoting the type of error. The system will also return additional information about the fault in the body of the response.

Example "Subnet not found" fault Response (XML)

```
<subnetNotFound code="450"
xmlns="http://netstack.org/quantum/api/v1.0">
   <message>
       Unable to find a subnet with the specified identifier.
   </message>
   <detail>
       Subnet 8de6af7c-ef95-4ac1-9d37-172f8df33a1f could not be
found
   </detail>
</subnetNotFound>
```

Example "Subnet not found" fault Response (JSON)

```
{
   "subnetNotFound": {
```

```
       "message": "Unable to find a subnet with the specified
identifier.",
        "code": 420,
        "detail": "Subnet 8de6af7c-ef95-4ac1-9d37-172f8df33a1f
could not be found"
   }
}
```

The root element of the fault (e.g. subnetNotFound) may change depending on the type of error. The following is a list of possible elements along with their associated error codes:

Fault Element	Error Code	Description
BadRequest	400	Malformed request body. The Quantum service is unable to parse the contents of the request body.
Unauthorized	401	User has not provided authentication credentials. If authentication is provided by the Keystone identity service, this might mean that either no authentication token has been supplied in the request, or that the token itself is either invalid or expired.
Forbidden	403	The user does not have the necessary rights to execute the requested operation
ItemNotFound	404	The requested resource does not exist on the Quantum API server
SubnetNotFound	450	The specified subnet has not been created or has been removed.
InvalidCIDR	451	CIDR incorrectly specified.
DuplicateCIDR	452	CIDR has already been used for another subnet in this project.
SubnetAlreadyAssociated	453	Subnet has already been associated with a route-table (needs to be dis-associated before associating again).
RoutetableNotFound	460	The specified route-table has not been created or has been removed.
RouteNotFound	465	The specified route has not been created or has been removed.
RouteSourceInvalid	466	Plugin cannot resolve this route source identifier.
RouteDestinationInvalid	467	Plugin cannot resolve this route destination identifier.
RouteTargetInvalid	468	Plugin cannot resolve this route target identifier.
ServiceUnavailable	470	API not supported.
PluginFault	471	Generic fault within the plugin.

7 API Operations

7.1 API at a glance

Subnets		
GET	`/tenants/{tenant_id}/subnets`	List summary of subnets configured in Quantum for a given tenant, identified by tenant-id
GET	`/tenants/{tenant_id}/subnets/{subnet_id}`	List information for a specific subnet, identified by subnet-id, for a given tenant, identified by tenant-id.
POST	`/tenants/{tenant_id}/subnets`	Creates a new subnet for the tenant identified by tenant-id
PUT	`/tenants/{tenant_id}/subnets/{subnet_id}`	Changes the CIDR and/or the virtual network association for the subnet identified by subnet_id.
DELETE	`/tenants/{tenant_id}/subnets/{subnet_id}`	Destroys the subnet identified by subnet_id for the tenant identified by tenant_id
Routetables		

GET	/tenants/{tenant_id}/routetables	List summary of routetables configured in Quantum for a given tenant, identified by tenant-id
GET	/tenants/{tenant_id}/routetables/{routetable_id}	List information for a specific routetable, identified by routetable_id, for a given tenant, identified by tenant-id.
POST	/tenants/{tenant_id}/routetables	Creates a new routetable for the tenant identified by tenant-id
PUT	/tenants/{tenant_id}/routetables/{routetable_id}	This currently changes the label and description of the routetable
DELETE	/tenants/{tenant_id}/routetables/{routetable_id}	Destroys the routetable identified by routetable_id for the tenant identified by tenant-id
Routes		
GET	/tenants/{tenant_id}/routetables/{routetable_id}/routes	List summary of routes configured in Quantum for a given tenant, and routetable
GET	/tenants/{tenant_id}/routetables/{routetable_id}/routes/{route_id}	List information for a specific route, identified by route_id, for a given tenant and routetable identified by their respective ids.
POST	/tenants/{tenant_id}/routetables/{routetable_id}/routes	Creates a new route for the tenant and routetable identified by their respective ids
PUT	/tenants/{tenant_id}/routetables/{routetable_id}/routes/{route_id}	This changes the source and/or destination and/or target of the route
DELETE	/tenants/{tenant_id}/routetables/{routetable_id}/routes/{route_id}	Destroys the route identified by route_id for the tenant and routetable identified by their respective ids.
Targets		
GET	/tenants/{tenant_id}/routetables/{routetable_id}/targets	List summary of targets configured in Quantum for a given tenant, and routetable.
Associations		
GET	/tenants/{tenant_id}/subnets/{subnet_id}/association	Get the id of the routetable with which this Subnet is associated.
PUT	/tenants/{tenant_id}/subnets/{subnet_id}/association	Associates the Subnet with a Routetable.
DELETE	/tenants/{tenant_id}/subnets/{subnet_id}/association	Destroys the subnet association of the subnet_id with a routetable for the tenant identified by tenant_id

The following sections provide details of each operation referenced in the table above.

7.2 Subnets

7.2.1 List Subnets

Verb	URI	Description
GET	/tenants/{tenant_id}/subnets	List summary of subnets configured in Quantum for a given tenant, identified by tenant-id

This operation does not require a request body.

Normal Response Code(s):	200
Error Response Code(s):	Unauthorized (401), Forbidden (403)

This operation returns the list of all subnets currently defined in Quantum for the tenant specified in the request URI. The returned list will provide the unique identifier of each subnet configured for the tenant specified in the resource URI.

Example: Subnet List Request/Response (XML)

Request:

```
GET /tenants/XYZ/subnets.xml
```

Response:

```
<subnets>
    <subnet id="8bec1293-16bd-4568-ba75-1f58bec0b4c3"/>
    <subnet id="2a39409c-7146-4501-8429-3579e03e9b56"/>
</subnets>
```

Example: Subnets List Request/Response (JSON)

Request:

```
GET /tenants/XYZ/subnets.json
```

Response:

```
{
    "subnets":
        [
            {
                "id": "8bec1293-16bd-4568-ba75-1f58bec0b4c3"
            },
            {
                "id": "2a39409c-7146-4501-8429-3579e03e9b56"
            }
        ]
}
```

7.2.2 Show Subnet

Verb	URI	Description
GET	`/tenants/{tenant_id}/subnets/{subnet_id}`	List information for a specific subnet, identified by subnet-id, for a given tenant, identified by tenant-id.

This operation does not require a request body.

Normal Response Code(s):	200

Error Response Code(s):	Unauthorized (401), Forbidden (403), SubnetNotFound(450)

This operation returns the identifier, the CIDR and the virtual network identifier (L2 network id) for a specific subnet configured in Quantum.

Example: Show Subnet Request/Response (XML)

Request:

```
GET /tenants/XYZ/subnets/8bec1293-16bd-4568-ba75-1f58bec0b4c.xml
```

Response:

```
```

Example: Show Subnet Request/Response (JSON)

Request:

```
GET /tenants/XYZ/subnets/8bec1293-16bd-4568-ba75-1f58bec0b4c.json
```

Response:

```
{
    "subnet":
            {
                "id": "8bec1293-16bd-4568-ba75-1f58bec0b4c3",
                "network_id": "1883dacb-ae95-413a-b918-
860a93665c7e",
                "cidr": "10.0.0.0/24"

            }
}
```

7.2.3 Create Subnet

Verb	URI	Description
POST	/tenants/{tenant_id}/subnets	Creates a new subnet for the tenant identified by tenant-id

The body for this request must contain a Subnet object specifying a CIDR for this subnet. It may optionally contain a Quantum L2 virtual network ID.
[If no virtual network ID is provided, should the plugin instantiate a Quantum L2 virtual network and associate it's ID with this subnet?]

Normal Response Code(s):	200
Error Response Code(s):	BadRequest (400) Unauthorized (401), Forbidden (403), InvalidCIDR (451), DuplicateCIDR(452)

Quantum validates the request, and dispatches it to the plugin, and then returns the unique identifier of the subnet to the caller.

Example: Create Subnet Request/Response (XML)

Request:

```
POST /tenants/XYZ/subnets.xml
```

```
<subnet
    cidr="10.0.0.0/24"/>
```

Response:

```
 <subnet
    id="158233b0-ca9a-40b4-8614-54a4a99d47d1"/>
```

Example: Create Subnet Request/Response (JSON)

Request:

```
POST /tenants/XYZ/subnets.json
```

```
{
   "subnet":
       {
            "cidr": "10.0.0.0/24"
       }
}
```

Response:

```
{
    "subnet":
            {
                "id": "158233b0-ca9a-40b4-8614-54a4a99d47d1",
            }
}
```

7.2.4 Update Subnet

Verb	URI	Description
PUT	`/tenants/{tenant_id}/subnets/{subnet_id}`	Changes the CIDR and/or the virtual network_id for the subnet identified by subnet_id for the tenant identified by tenant-id

The body for this request must contain a Subnet object specifying a CIDR for this subnet, and/or a Quantum L2 virtual network ID.

Normal Response Code(s):	`204`
Error Response Code(s):	`BadRequest (400) Unauthorized (401), Forbidden (403), SubnetNotFound(450), InvalidCIDR (451), DuplicateCIDR(452)`

This operation changes the CIDR of a Quantum subnet using the data provided in the request body.

Example: Update Subnet Request/Response (XML)

Request:

```
PUT /tenants/XYZ/subnets/158233b0-ca9a-40b4-8614-54a4a99d47d1.xml
```

```
<subnet
    cidr="10.1.0.0/24"/>
```

Response:

```
 None
```

Example: Update Subnet Request/Response (JSON)

Request:

```
PUT /tenants/XYZ/subnets/158233b0-ca9a-40b4-8614-
54a4a99d47d1.json
```

```
{
   "subnet":
       {
            "cidr": "10.1.0.0/24"
```

```
        }
}
```

Response:

```
None
```

7.2.5 Delete Subnet

Verb	URI	Description
DELETE	/tenants/{tenant_id}/subnets/{subnet_id}	Destroys the subnet identified by subnet_id for the tenant identified by tenant_id

This operation does not require a request body.

Normal Response Code(s):	204
Error Response Code(s):	Unauthorized (401), Forbidden (403), SubnetNotFound(450)

No data is returned in the response body.

Example: Delete Subnet Request/Response (XML)

Request:

```
DELETE /tenants/XYZ/subnets/8bec1293-16bd-4568-ba75-
1f58bec0b4c.xml
```

Response:

```
None
```

Example: Delete Subnet Request/Response (JSON)

Request:

```
DELETE /tenants/XYZ/subnets/8bec1293-16bd-4568-ba75-
1f58bec0b4c.json
```

Response:

```
None
```

7.3 Routetables

7.3.1 List Routetables

Verb	URI	Description
GET	/tenants/{tenant_id}/routetables	List summary of routetables configured in Quantum for a given tenant, identified by tenant-id

This operation does not require a request body.

Normal Response Code(s):	200
Error Response Code(s):	Unauthorized (401), Forbidden (403)

This operation returns the list of all routetables currently defined in Quantum for the tenant specified in the request URI. The returned list will provide the unique identifier of each routetable configured for the tenant specified in the resource URI.

Example: Routetable List Request/Response (XML)

Request:

```
GET /tenants/XYZ/routetables.xml
```

Response:

```
<routetables>
    <routetable id="8bec1293-16bd-4568-ba75-1f58bec0b4c3"/>
    <routetable id="2a39409c-7146-4501-8429-3579e03e9b56"/>
</routetables>
```

Example: Routetables List Request/Response (JSON)

Request:

```
GET /tenants/XYZ/routetables.json
```

Response:

```
{
    "routetables":
```

```
        [
            {
                "id": "8bec1293-16bd-4568-ba75-1f58bec0b4c3"
            },
            {
                "id": "2a39409c-7146-4501-8429-3579e03e9b56"
            }
        ]
}
```

7.3.2 Show Routetable <this is currently not doing much, ideally it should return all the routes in this routetable, and all the associated subnets>

Verb	URI	Description
GET	/tenants/{tenant_id}/routetables/{routetable_id}	List information for a specific routetable, identified by routetable_id, for a given tenant, identified by tenant-id.

This operation does not require a request body.

Normal Response Code(s):	200
Error Response Code(s):	Unauthorized (401), Forbidden (403), RoutetableNotFound(460)

This operation returns the identifier for a specific routetable configured in Quantum.

Example: Show Routetable Request/Response (XML)

Request:

```
GET /tenants/XYZ/routetables/8bec1293-16bd-4568-ba75-
1f58bec0b4c.xml
```

Response:

```

```

Example: Show Routetable Request/Response (JSON)

Request:

```
GET /tenants/XYZ/routetables/8bec1293-16bd-4568-ba75-
1f58bec0b4c.json
```

Response:

```
{
    "routetable":
            {
                "id": "8bec1293-16bd-4568-ba75-1f58bec0b4c3",
            }
}
```

7.3.3 Create Routetable

Verb	URI	Description
POST	/tenants/{tenant_id}/routetables	Creates a new routetable for the tenant identified by tenant-id

The body for this request is empty.

Normal Response Code(s):	200
Error Response Code(s):	BadRequest (400) Unauthorized (401), Forbidden (403)

Quantum validates the request, and dispatches it to the plugin, and then returns the unique identifier of the routetable to the caller.

Example: Create Routetable Request/Response (XML)

Request:

```
POST /tenants/XYZ/routetables.xml
```

Response:

```
 <routetable
    id="158233b0-ca9a-40b4-8614-54a4a99d47d1"/>
```

Example: Create Routetable Request/Response (JSON)

Request:

```
POST /tenants/XYZ/routetables.json
```

Response:

```
{
    "routetable":
            {
                "id": "158233b0-ca9a-40b4-8614-54a4a99d47d1",
            }
}
```

7.3.4 Update Routetable

Verb	URI	Description
PUT	/tenants/{tenant_id}/routetables/{routetable_id}	This currently changes the label and description of the routetable

The body for this request may contain label and/or description.

Normal Response Code(s):	204
Error Response Code(s):	BadRequest (400) Unauthorized (401), Forbidden (403), RoutetableNotFound(460)

This operation changes the label and or description of a Quantum routetable using the data provided in the request body.

Example: Update Routetable Request/Response (XML)

Request:

```
PUT /tenants/XYZ/routetables/158233b0-ca9a-40b4-8614-
54a4a99d47d1.xml
```

```
<routetable
    label="some-new-lable"/>
```

Response:

```
 None
```

Example: Update Routetable Request/Response (JSON)

Request:

```
PUT /tenants/XYZ/routetables/158233b0-ca9a-40b4-8614-
54a4a99d47d1.json
```

```
{
   "routetable":
       {
            "label": "some-new-label"
       }
}
```

Response:

```
None
```

7.3.5 Delete Routetable

Verb	URI	Description
DELETE	/tenants/{tenant_id}/routetables/{routetable_id}	Destroys the routetable identified by routetable_id for the tenant identified by tenant-id

This operation does not require a request body.

Normal Response Code(s):	204
Error Response Code(s):	Unauthorized (401), Forbidden (403), RoutetableNotFound(460)

This operation does not return any data.

Example: Delete Routetable Request/Response (XML)

Request:

```
DELETE /tenants/XYZ/routetables/8bec1293-16bd-4568-ba75-
1f58bec0b4c.xml
```

Response:

```
None
```

Example: Delete Routetable Request/Response (JSON)

Request:

```
DELETE /tenants/XYZ/routetables/8bec1293-16bd-4568-ba75-
1f58bec0b4c.json
```

Response:

```
None
```

7.4 Routes

7.4.1 List Routes

Verb	URI	Description
GET	`/tenants/{tenant_id}/routetables/{routetable_id}/routes`	List summary of routes configured in Quantum for a given tenant, and routetable

This operation does not require a request body.

Normal Response Code(s):	`200`
Error Response Code(s):	`Unauthorized (401), Forbidden (403), RoutetableNotFound(460)`

This operation returns the list of all routes currently defined in Quantum for the tenant and the routetable specified in the request URI. The returned list will provide the unique identifier of each route along with the details of the route entry, i.e. the source, destination, and target fields, and also the routetable identifier.

Example: Routes List Request/Response (XML)

Request:

```
GET /tenants/XYZ/routetables/8bec1293-16bd-4568-ba75-
1f58bec0b4c3/routes.xml
```

Response:

```
```

Example: Routes List Request/Response (JSON)

Request:

```
GET /tenants/XYZ/routetables/8bec1293-16bd-4568-ba75-
1f58bec0b4c3/routes.json
```

Response:

```
{
    "routes":
        [
            {
                "id": "8bec1293-16bd-4568-ba75-1f58bec0b4c3",
                "routetable-id": "bffafd87-be7f-4d3a-9f8d-
2a0f6bd6a109",
                "source": "subnet-A",
                "destination": "10.0.0.0/24",
                "target": "Private"
            },
            {
                "id": "2a39409c-7146-4501-8429-3579e03e9b56",
                "routetable-id": "59ee1ca1-e54c-40df-bea0-
c70a3742ff00",
                "source": "subnet-B",
                "destination": "10.0.0.0/24",
                "target": "Private"

            }
        ]
}
```

7.4.2 Show Route

Verb	URI	Description
GET	/tenants/{tenant_id}/routetables/{routetable_id}/routes/{route_id}	List information for a specific route, identified by route_id, for a given tenant and routetable identified by their respective ids.

This operation does not require a request body.

Normal Response Code(s):	200
Error Response Code(s):	Unauthorized (401), Forbidden (403), RoutetableNotFound(460), RouteNotFound(465)

This operation returns the identifier, source, destination, target, and routetable_id for a specific route configured in Quantum.

Example: Show Route Request/Response (XML)

Request:

```
GET /tenants/XYZ/routetables/8bec1293-16bd-4568-ba75-
1f58bec0b4c/routes/59ee1ca1-e54c-40df-bea0-c70a3742ff00.xml
```

Response:

```
```

Example: Show Route Request/Response (JSON)

Request:

```
GET /tenants/XYZ/routetables/8bec1293-16bd-4568-ba75-
1f58bec0b4c/routes/59ee1ca1-e54c-40df-bea0-c70a3742ff00.json
```

Response:

```
{
    "route":
            {
                "id": " 59ee1ca1-e54c-40df-bea0-c70a3742ff00",
                "routetable_id": " 8bec1293-16bd-4568-ba75-
1f58bec0b4c3",
                "source": " subnet-A",
                "destination": "10.0.0.0/24",
                "target": "Private"
            }
}
```

7.4.3 Create Route

Verb	URI	Description
POST	/tenants/{tenant_id}/routetables/{routetable_id}/routes	Creates a new route for the tenant and routetable identified by their respective ids

The body for this request is empty.

Normal Response Code(s):	200

Error Response Code(s):	BadRequest (400) Unauthorized (401), Forbidden (403), RoutetableNotFound(460), RouteSourceInvalid(466), RouteDestinationInvalid(467), RouteTargetInvalid(468)

Quantum validates the request, and dispatches it to the plugin, and then returns the unique identifier of the route to the caller.

Example: Create Route Request/Response (XML)

Request:

```
POST /tenants/XYZ/routetables/bffafd87-be7f-4d3a-9f8d-
2a0f6bd6a109/routes.xml
```

Response:

```
 <route
    id="158233b0-ca9a-40b4-8614-54a4a99d47d1"/>
```

Example: Create Route Request/Response (JSON)

Request:

```
POST /tenants/XYZ/routetables/bffafd87-be7f-4d3a-9f8d-
2a0f6bd6a109/routes.json
```

Response:

```
{
    "route":
            {
                "id": "158233b0-ca9a-40b4-8614-54a4a99d47d1",
            }
}
```

7.4.4 Update Route

Verb	URI	Description
PUT	/tenants/{tenant_id}/routetables/{routetable_id}/routes/{route_id}	This changes the source and/or destination and/or target of the route

The body for this request may contain the updated source and/or destination

and/or target.	
Normal Response Code(s):	204
Error Response Code(s):	BadRequest (400) Unauthorized (401), Forbidden (403), RoutetableNotFound(460), RouteNotFound(465), RouteSourceInvalid(466), RouteDestinationInvalid(467), RouteTargetInvalid(468)

This operation changes the source and/or destination and/or target of a Quantum route using the data provided in the request body.

Example: Update Route Request/Response (XML)

Request:

```
PUT /tenants/XYZ/routetables/158233b0-ca9a-40b4-8614-
54a4a99d47d1/routes/59ee1ca1-e54c-40df-bea0-c70a3742ff00.xml
```

```
<route
    source="subnet-B"/>
```

Response:

```
 None
```

Example: Update Route Request/Response (JSON)

Request:

```
PUT /tenants/XYZ/routetables/158233b0-ca9a-40b4-8614-
54a4a99d47d1/routes/59ee1ca1-e54c-40df-bea0-c70a3742ff00.json
```

```
{
   "route":
       {
            "source": "subnet-B",
            "destination": "192.168.0.0/24",
            "target": "Private"
       }
}
```

Response:

```
None
```

7.4.5 Delete Route

Verb	URI	Description
DELETE	`/tenants/{tenant_id}/routetables/{routetable_id}/routes/{route_id}`	Destroys the route identified by route_id for the tenant and routetable identified by their respective ids.

This operation does not require a request body.

Normal Response Code(s):	`204`
Error Response Code(s):	`Unauthorized (401), Forbidden (403), RoutetableNotFound(460), RouteNotFound(465)`

This operation does not return any data.

Example: Delete Route Request/Response (XML)

Request:

```
DELETE /tenants/XYZ/routetables/8bec1293-16bd-4568-ba75-
1f58bec0b4c/routes/59ee1ca1-e54c-40df-bea0-c70a3742ff00.xml
```

Response:

```
None
```

Example: Delete Route Request/Response (JSON)

Request:

```
DELETE /tenants/XYZ/routetables/8bec1293-16bd-4568-ba75-
1f58bec0b4c/routes/59ee1ca1-e54c-40df-bea0-c70a3742ff00.json
```

Response:

```
None
```

7.5 Targets

7.5.1 List Targets

Verb	URI	Description
GET	`/tenants/{tenant_id}/routetables/{routetable_id}/targets`	List summary of targets configured in Quantum for a given tenant, and routetable.

This operation does not require a request body.

Normal Response Code(s):	`200`
Error Response Code(s):	`Unauthorized (401), Forbidden (403), RoutetableNotFound(460)`

This operation returns the list of all the targets which are available to this particular tenant. These will contain a list of system (i.e. SP-published targets), and there might be others which are published by other tenants.

Example: Targets List Request/Response (XML)

Request:

```
GET /tenants/XYZ/routetables/8bec1293-16bd-4568-ba75-
1f58bec0b4c3/targets.xml
```

Response:

```
```

Example: Targets List Request/Response (JSON)

Request:

```
GET /tenants/XYZ/routetables/8bec1293-16bd-4568-ba75-
1f58bec0b4c3/routes.json
```

Response:

```
{
    "targets":
        [
            {
                "tag": "Private",
                "description": "System"
            },
```

```
            {
                "tag": "Public",
                "description": "System"
            }
            {
                "tag": "VPN",
                "description": "System"
            }
        ]
}
```

7.6 Associations

7.6.1 Show Subnet Association

Verb	URI	Description
GET	`/tenants/{tenant_id}/subnets/{subnet_id}/association`	Get the id of the routetable with which this Subnet is associated.

This operation does not require a request body.

Normal Response Code(s):	`200`
Error Response Code(s):	`Unauthorized (401), Forbidden (403), SubnetNotFound(450)`

This operation returns the routetable identifier with which this subnet is associated. An empty string is returned if no routetable is associated.

Example: Show Subnet Association Request/Response (XML)

Request:

```
GET /tenants/XYZ/subnets/8bec1293-16bd-4568-ba75-
1f58bec0b4c/association.xml
```

Response:

```
```

Example: Show Subnet Association Request/Response (JSON)

Request:

```
GET /tenants/XYZ/subnets/8bec1293-16bd-4568-ba75-
1f58bec0b4c/association.json
```

Response:

```
{
    "association":
            {
                "routetable_id": "1883dacb-ae95-413a-b918-
860a93665c7e"
            }
}
```

7.6.2 Create Subnet Association

Verb	URI	Description
PUT	/tenants/{tenant_id}/subnets/{subnet_id}/association	Associates the Subnet with a Routetable.

The body for this request must contain a association object specifying a routetable id.

Normal Response Code(s):	204
Error Response Code(s):	BadRequest (400) Unauthorized (401), Forbidden (403), SubnetNotFound(450), SubnetAlreadyAssociated(453)

This operation will attempt to associate the subnet with the routetable id provided in the request body.

Example: Create Subnet Association Request/Response (XML)

Request:

```
PUT /tenants/XYZ/subnets/158233b0-ca9a-40b4-8614-
54a4a99d47d1/association.xml
```

```
<association
    routetable_id="1883dacb-ae95-413a-b918-860a93665c7e "/>
```

Response:

```
 None
```

Example: Create Subnet Association Request/Response (JSON)

Request:

```
PUT /tenants/XYZ/subnets/158233b0-ca9a-40b4-8614-
54a4a99d47d1/association.json
```

```
{
   "association":
       {
            "routetable_id": "1883dacb-ae95-413a-b918-
860a93665c7e"
       }
}
```

Response:

```
None
```

7.6.3 Delete Subnet Association

Verb	URI	Description
DELETE	/tenants/{tenant_id}/subnets/{subnet_id}/association	Destroys the subnet association of the subnet_id with a routetable for the tenant identified by tenant_id

This operation does not require a request body.

Normal Response Code(s):	204
Error Response Code(s):	Unauthorized (401), Forbidden (403), SubnetNotFound(450)

Returns the routetable_id that is disassociated.

Example: Delete Subnet Association Request/Response (XML)

Request:

```
DELETE /tenants/XYZ/subnets/8bec1293-16bd-4568-ba75-
1f58bec0b4c/association.xml
```

Response:

```
None
```

Example: Delete Subnet Association Request/Response (JSON)

Request:

```
DELETE /tenants/XYZ/subnets/8bec1293-16bd-4568-ba75-
1f58bec0b4c/association.json
```

Response:

```
{
   "association":
       {
            "routetable_id": "1883dacb-ae95-413a-b918-
860a93665c7e"
       }
}
```

8 Work in progress

- IPAM/Melange Linkage
- ACL Resource Definitions
- NAT service policy definitions
- L3 level Services(example : FW, SLB and so on) Insertion Models.

Open Questions/Work in progress

1. Do we need explicit Gateway as a resource?
2. API extensions for L3 abstractions - We may have to identify the model
3. API Model – need to be modeled after the current essex Quantum API – tags and status and so on.
4. Monitoring and Debugging APIs?