Quantum API proposal

Contents

Introduction

This document constitutes a first attempt in defining an API for the Quantum service. The operation list cannot be deemed complete, and formats for request and response messages have not yet been defined. Moreover, authentication, authorization, and extension mechanisms, as well as the URL structure, are not defined in this document.

Since Quantum provides "*network connectivity as a service*", this API defines Layer-2 operations only. This document also proposes operations for bridging Quantum networks with external networks, such as a customer-managed network in an on-premise data centre; however, use cases in this area (bridging and federation) should be explored better.

We first introduce the operation list, and then discuss how these operations can be used to accomplish some common use cases.

Glossary

Network	A virtual network providing basic connectivity only, i.e.: collection of virtual ports sharing network connectivity. In the Quantum terminology, a network is always a Layer-2 network.
Plugin	Software component providing actual implementation for Quantum APIs.
VIF	Virtual InterFace, also known as virtual NIC or virtual Network Interface.
Attachment	Resource plugged into a port of a virtual Layer-2 network.
Logical Port	A port on the virtual network switch represented by a virtual Layer-2 network

Operation List

For each operation, both XML and JSON representation should be assumed.

List of error response codes:

Code	Name	Description
400	Bad Request	Malformed request body
401	Unauthorized	User does not have the rights to execute the operation
413	OverLimit	Maximum limit of resources has been reached
421	AttachmentsPlugged	Attachments plugged into Layer-2 network's port
423	AlreadyAttached	Attachment is already plugged into another port
424	PortInUse	There is already an attachment plugged into the port
500	Service Unavailable	Quantum plugin did not respond
510	PluginFault	The Quantum plugin could not complete the operation

1. List networks

Verb	URI	Description
GET	/networks	List summary of networks configured in Quantum

Request Body:

This operation does not require a request body.

Description:

This operation returns the list of all networks currently defined in Quantum; returned list includes at least the network's unique identifier.

Response Codes:

Normal Response code: 200

Error response code(s): 500 Service Unavailable, 401 Unauthorized

2. List network details

Verb	URI	Description
GET	/networks/*id*	List details of network identified by *id*

Request Body:

This operation does not require a request body.

Description:

This operation provides detailed output for a specific network configured in Quantum.

Response Codes:

Normal Response code: 200

Error response code(s): 500 Service Unavailable, 400 Bad Request, 401 Unauthorized

3. Create Network

Verb	URI	Description
POST	/networks	Create a new network

Request Body:
The request body for this network should contain a symbolic name for the network.

Description:
This operation asynchronously creates a Layer-2 network in Quantum based on the information provided in the request body.

Quantum validates the request, creates the network object, dispatches it to the plugin, and then returns the unique identifier of the network to the caller, who can check the progress of the operation performing a GET on `networks/id`.

Resources for the new network can be either provided at create time or when virtual interfaces are plugged into this network depending on the particular plugin implementation.

If the validation phase fails, the network object is not created at all, and a 400 error is returned to the caller.

Response Codes:
Normal Response code: 202
Error response code(s): 500 Service Unavailable, 400 Bad Request, 401 Unauthorized, 413 Over Limit

4. Update Network

Verb	URI	Description
PUT	/networks/*id*	Changes the configuration of the network identified by *id*

Request Body:
The request body for this operation will possibly contain the new symbolic name for the network.

Description:
This operation asynchronously updates a network in Quantum according to the information provided in the request body.

Response Codes:
Normal Response code: 202
Error response code(s): 500 Service Unavailable, 400 Bad Request, 401 Unauthorized, 510 PluginFault

5. Delete Network

Verb	URI	Description
DELETE	/networks/*id*	Removes the network identified by *id*

Request Body:
This operation does not require a request body.

Description:
This operation removes the network specified in the URI.

It will fail if an attachment is plugged into anyone of the network's ports (error code 511). It will not fail instead if logical ports are configured for the network, as long as nothing is plugged into them. Logical ports will be destroyed together with the network.

The operation is asynchronous. Quantum forwards it to the plugin, and upon success removes the network object. Callers can check the current status of the operation by performing a GET on `networks/id`.

This operation is not recoverable.

Response Codes:

Normal Response code: 202

Error response code(s): 500 Service Unavailable, 400 Bad Request, 401 Unauthorized, 510 PluginFault, 511 AttachmentsPlugged

6. List logical ports for network

Verb	URI	Description
GET	/networks/*id*/ports	Lists all the ports currently defined for a Quantum network

Request Body:

This operation does not require a request body.

Description:

This operation lists all the ports currently configured for a network. Response should include, for each port, at least its unique identifier (might just be a port number) and the identifier of the attachment plugged into it, if any.

Response Codes:

Normal Response code: 200

Error response code(s): 500 Service Unavailable, 401 Unauthorized

7. List port details

Verb	URI	Description
GET	networks/*net_id*/ports/*port_id*	Retrieves detail of the port *port_id* configured for the network *net_id*.

Request Body:

This operation does not require a request body.

Description:

This operation provides detailed output for a specific port configured for a given network. This operation will return all the attributes of the port, including plugin-specific attributes.

Response Codes:

Normal Response code: 200

Error response code(s): 500 Service Unavailable, 400 Bad Request, 401 Unauthorized

8. Create Port

Verb	URI	Description
POST	/networks/*id*/ports	Creates a logical port on the network specified in the request URI

Request Body:

The request body is not mandatory.

Description:

This operation asynchronously creates a port on a Quantum network based on the information provided in the request body.

Quantum validates the request, creates the port object and attaches it to the appropriate network object. The request is then dispatched to the plugin.

Resources for the new port can be either provided at create time or when virtual interfaces are plugged into this port depending on the particular plugin implementation. Also, this operation could not be implemented for some plugins as the number of ports available might be fixed when the network is created.

If the validation phase fails, the port object is not created at all, and a 400 error is returned to the caller.

Response Codes:

Normal Response code: 202

Error response code(s): 500 Service Unavailable, 400 Bad Request, 401 Unauthorized, 413 Over Limit

9. Delete logical port

Verb	URI	Description
DELETE	/networks/*net_id*/ports/*port_id*	Removes a port from the logical

Request Body:

This operation does not require a request body.

Description:

This operation removes a logical port from a Quantum network.

This operation might not be available for plugins in which the number of ports is fixed at network creation; in this case ports should not be deleted, just as they cannot be created.

The operation is not recoverable and will fail if an attachment is plugged into the port.

Response Codes:

Normal Response code: 202

Error response code(s): 500 Service Unavailable, 400 Bad Request, 401 Unauthorized, 510 PluginFault, 511 AttachmentsPlugged.

10. List attachment details for port

Verb	URI	Description
GET	/networks/*net_id*/ports/*port_id*/attachment	Lists complete information about the attachment currently plugged into the specified port

Request Body:
This operation does not require a request body.

Description:
This operation returns configuration details for the attachment plugged into the port specified in the request URI. This information might include:

- Type of the attachment (e.g.: virtual network interface, bridge device);
- A reference to the resource being attached (If managed by Quantum);
- A reference to an external resource being attached (If not managed by Quantum);

Response Codes:
Normal Response code: 200

Error response code(s): 500 Service Unavailable, 400 Bad Request, 401 Unauthorized

11. Attach resource to port

Verb	URI	Description
PUT	/networks/*net_id*/ports/*port_id*/attachment	Plugs a resource (e.g.: virtual network interface, bridging device) into a logical port on a virtual network

Request Body:
The request body for this network should contain a reference to the resource to plug into the logical port.

Plugged resources can either be managed by or external to Quantum. In both cases, the request body will contain an identifier for that resource.

Description:
This operation asynchronously plugs a resource, or attachment, into the logical port specified in the request URL.

A resource could be either a virtual network interface belonging to a VM instance, a bridging device used to extend the Quantum network in a different data centre, or any kind of device which might be plugged into the Quantum network in order to provide different services, such as IP addressing management. Different kinds of resources can be defined in the future.

The request will be first validated by Quantum and then dispatched to the plugin. As the request is asynchronous, control is immediately returned to the caller. Progress of the operation can then be checked by querying the attachment (or the logical port) with a GET request.

The validation can fail if:

- The attachment is already plugged somewhere else;
- There is already another attachment plugged into the specified logical port.

If the validation phase fails, the attachment object is not created at all, and a 400 error is returned to the caller.

Response Codes:

Normal Response code: 202

Error response code(s): 500 Service Unavailable, 400 Bad Request, 401 Unauthorized, 413 Over Limit, 423 AlreadyAttached, 423 PortInUse, 510 PluginFault

12. Remove attachment from port

Verb	URI	Description
DELETE	/networks/*net_id*/ports/*port_id*/attachment	Removes the currently attached resource

Request Body:

This operation does not require a request body.

Description:

This operation asynchronously removes an attachment from a logical port.

This operation cannot be undone.

Response Codes:

Normal Response code: 202

Error response code(s): 500 Service Unavailable, 400 Bad Request, 401 Unauthorized, 413 Over Limit, 510 PluginFault

13. Retrieve resource attached to network

Verb	URI	Description
GET	/networks/*net_id*/attachments	Retrieve all the resources (e.g.: virtual network interface, bridging device) currently attached to a network

Request Body:

This operation does not require a request body.

Description:

This operation return a list of resource currently attached to the virtual network specified in the URI. For each resource, the returned list should specify at least the identifier of the resource and the identifier of the port in which the resource is currently plugged.

Response codes:

Normal Response code: 200

Error response code(s): 500 Service Unavailable, 400 Bad Request, 401 Unauthorized

14. Attach resource to network

Verb	URI	Description
POST	/networks/*net_id*/attachments	Plugs a resource (e.g.: virtual network interface, bridging device) into the specified network, without specifying a logical port

Request Body:

The request body for this network should contain a reference to the resource to plug into the logical port.

Plugged resources can either be managed by or external to Quantum. In both cases, the request body will contain an identifier for that resource.

Description:

This operation asynchronously plugs a resource, or attachment, into the logical port specified in the request URL.

A resource could be either a virtual network interface belonging to a VM instance, a bridging device used to extend the Quantum network in a different data centre, or any kind of device which might be plugged into the Quantum network in order to provide different services, such as IP addressing management. Different kinds of resources can be defined in the future.

This operation is very similar to "*attach resource to port*", with the only exception that a logical port is not specified in this case. Quantum plugins can either use a free port on the network, or create a new logical port, and then plug the resource into it. The identifier of the logical port where the resource has been plugged should be returned to the caller.

Response Codes:

Normal Response code: 202

Error response code(s): 500 Service Unavailable, 400 Bad Request, 401 Unauthorized, 413 Over Limit, 423 AlreadyAttached, 424 PortInUse, 510 PluginFault

Use Cases

1. Create a Quantum network

Customer uses Quantum API for creating a Layer-2.

- *Customer creates Layer-2 network;*
  Quantum returns network identifier and invokes plugin;

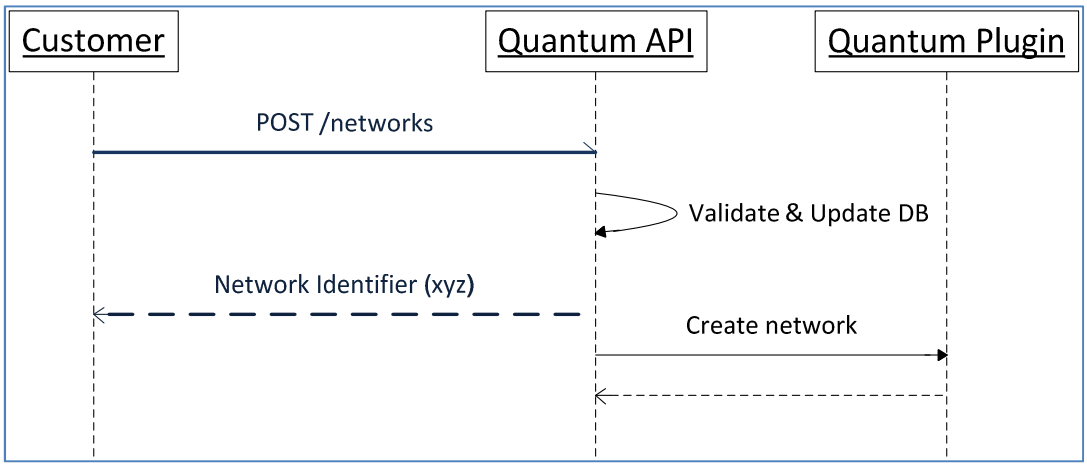

Figure 1 - Use case 1: Create a Quantum network

2. Create network and spawn VM

Customer uses Quantum for creating a network and then the Openstack compute service to create a virtual machine to be connected to this network.

The Openstack compute Service, while creating the VM interacts with Quantum for creating a logical port and plugging the virtual network interface into this port. IP Configuration is not provided by Quantum.

Moreover, compute service can either explicitly create a port or just instruct Quantum to plug the VIF into the Network. In this case Quantum will either use an available logical port or create a new port (see Figure 3).

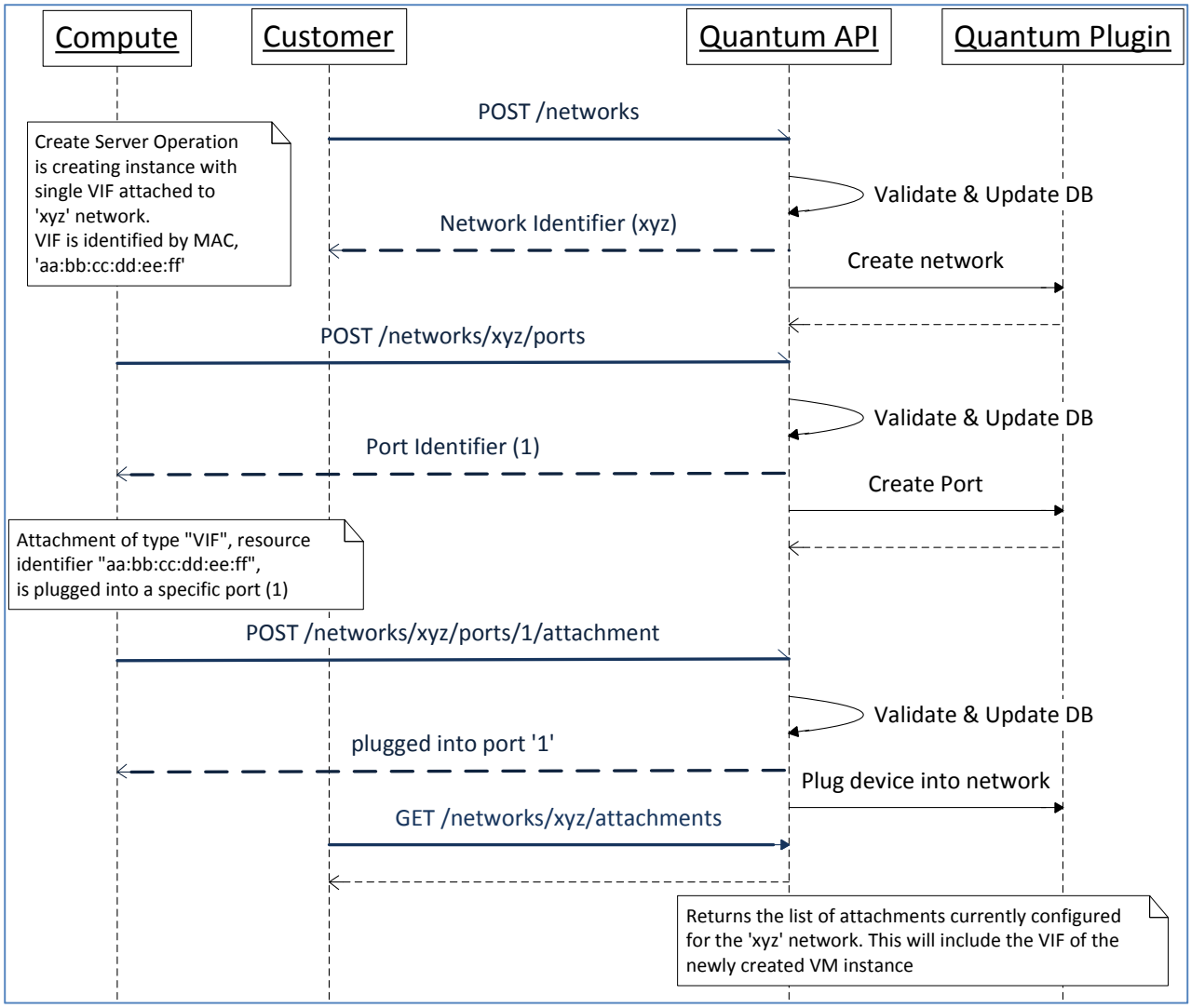

Figure 2 - Use Case 3: create Quantum network, spawn virtual machine, and attach its virtual network interface to previously created network.

- *Customer Creates a Layer-2 network;*
  Quantum returns network identifier, 'xyz';
- *Customer Creates a VM using Cloud Controller API (e.g.: POST /Servers);*
  Customers specifies a network for the VM's VIFs in the create server request;
  The Cloud Controller dispatches the call to the Compute Service which creates the VM;
- *Compute Service tells Quantum to create a logical port on the 'xyz' network;*
  Quantum creates the logical port and returns the port number;
- *Compute Service instructs Quantum to plug the VM's VIF into the previously created port;*
  VIF's unique identifier is provided in the request body.
  Quantum invokes the plugin to attach the VIF to the logical network. This operation might require the plugin to contact the hypervisor on which the VM has been created in order to setup networking according to the technology adopted by the plugin

- *Customer can verify that the attachment has been plugged into the network by querying Quantum for attachments for the 'xyz' network.*
  Quantum returns a list of currently configured attachments, together with their state (e.g.: in Progress, Attached/DOWN, Attached/UP, Failed).

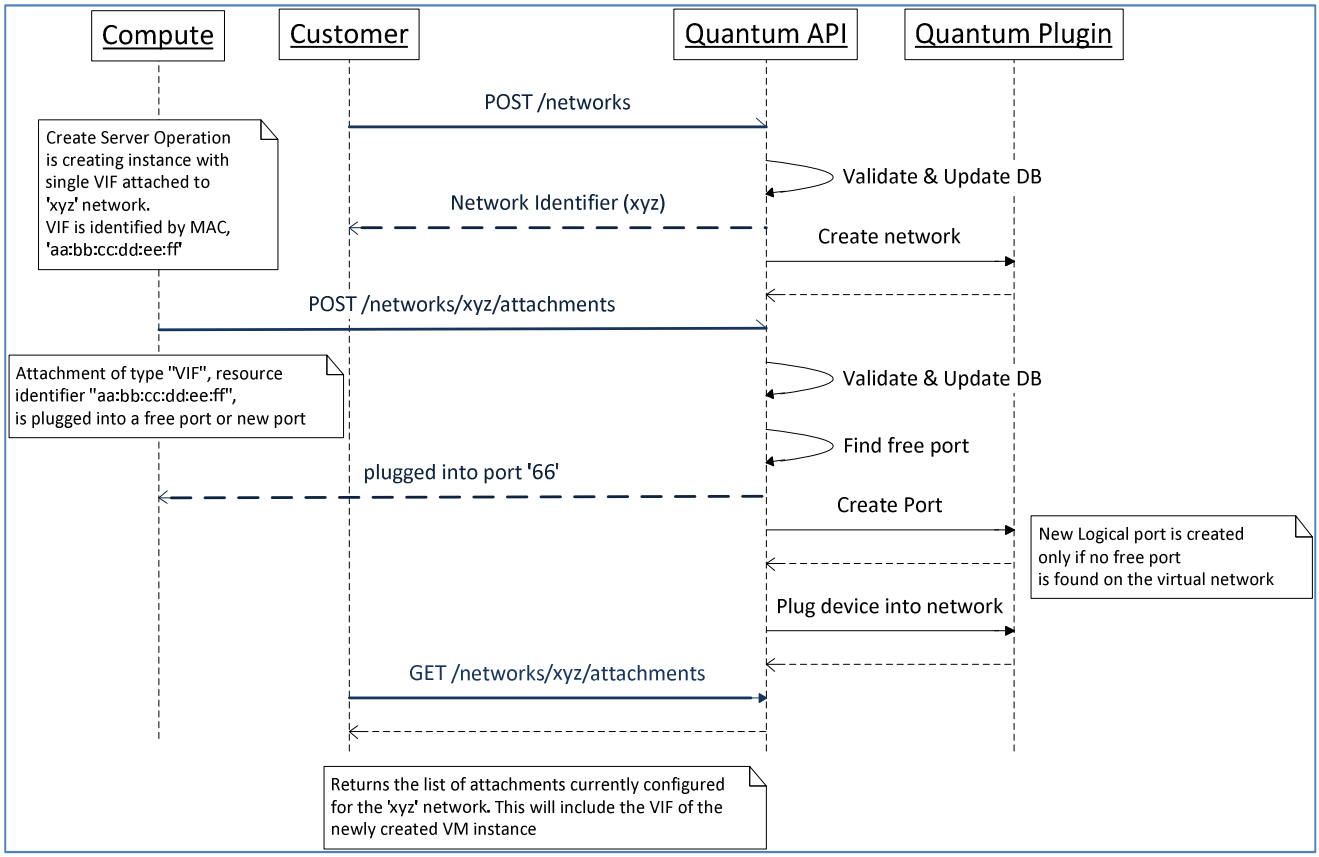

Figure 3 - Create a quantum network and attach a VM to it, without explicitly specifying a port

3. Spawn VM with multiple VIFs

This use case is very similar to the previous one, with the only difference that the VM being spawned has two VIFs. The compute service repeats the following sequence of operations for each VIF:

- Create logical port;
- Plug VIF into logical port;

The sequence diagram below explains how the compute service interacts with Quantum to do so.

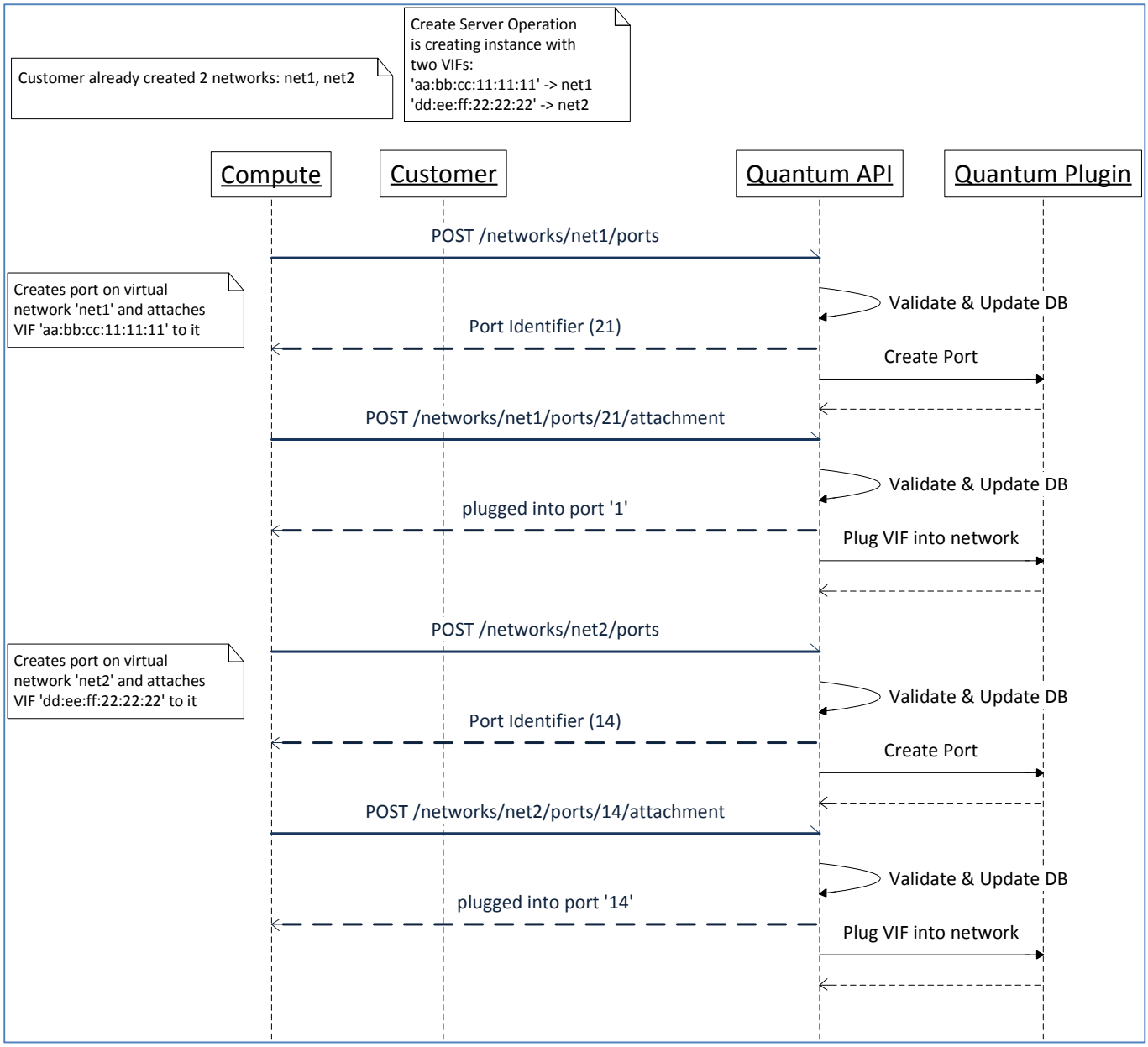

Figure 4 - Use Case 4: spawn a VM with multiple virtual network interfaces

4. Change network for VIF

The customer wants to remove a VIF from an IP subnet and attach it to another IP subnet defined for a different network.

In the diagram reported below, the VIF is moved from subnet ‘sub1’, defined for network ‘net1’, to subnet ‘sub2’, defined for network ‘net2’.

This will involve the following Quantum operations:

- Unplug VIF from network ‘net1’;
- Plug VIF into network ‘net2’;

NOTE: The plugin might not allow users to unplug VIF attachments belonging to VM which are already running. In this case an error code will be returned by the operation.

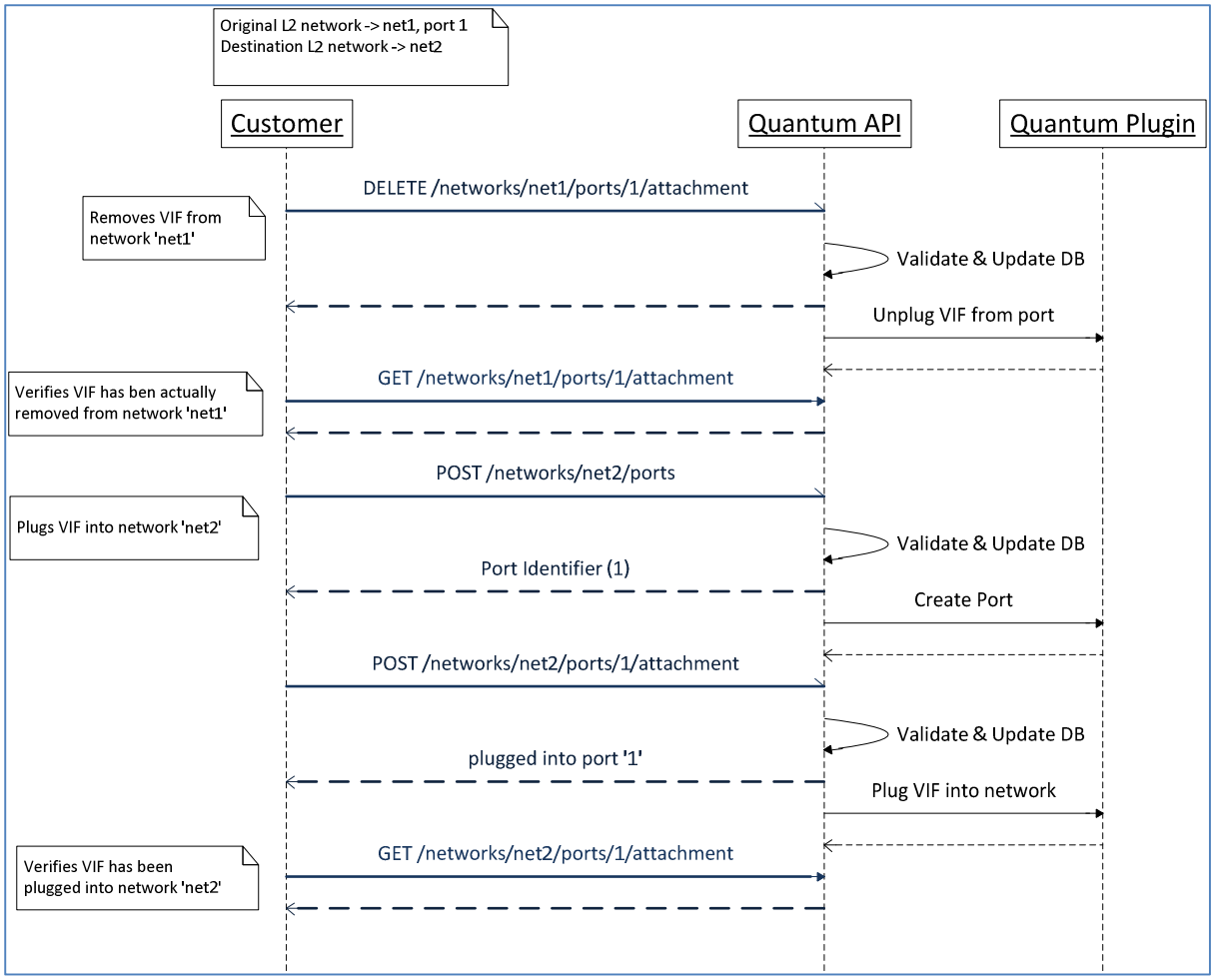

Figure 5 - Use case 5: change network assignment for a virtual network interface

5. Destroy network

The customer wants to destroy a Quantum network and all the resources associated with it.

This will require using Quantum for the following operations:

- For each attachment connected to the network, unplug it;
- Destroy the network object.

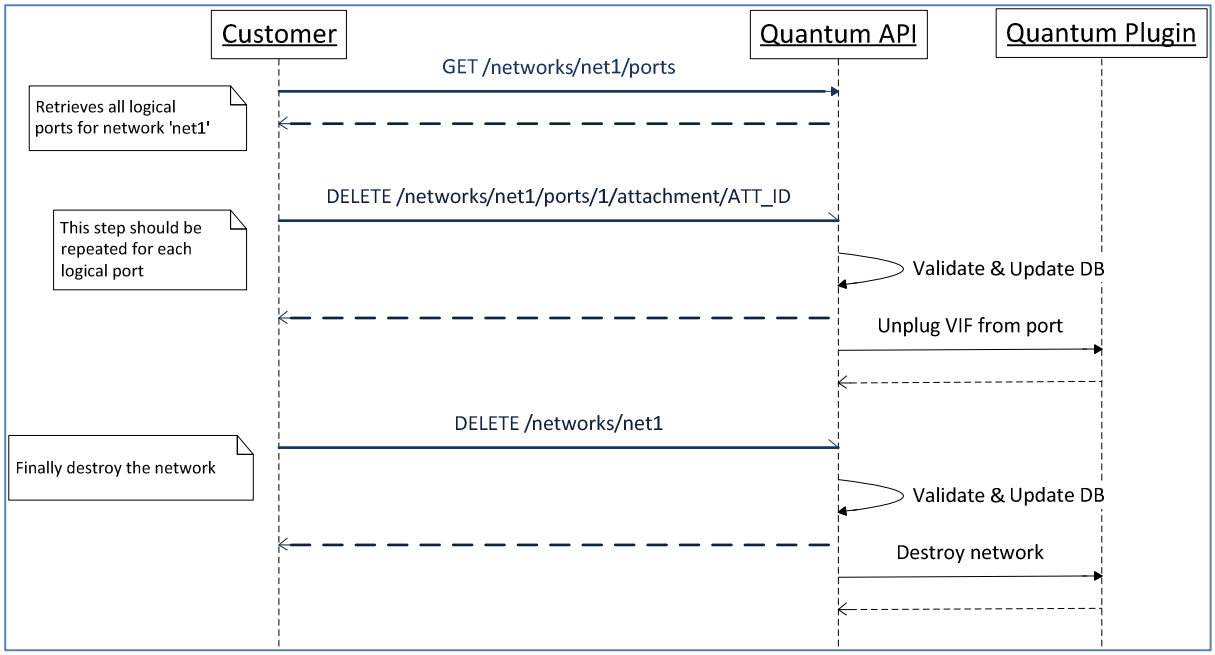

Figure 6 - Use case 6, remove a Quantum network