Promise: Resource Management

Release 1.0.2

November 20, 2015

Project Promise, https://wiki.opnfv.org/promise

Editors Ashiq Khan (NTT DOCOMO), Bertrand Souville (NTT DOCOMO)

Authors Ravi Chunduru (ClearPath Networks), Peter Lee (ClearPath Networks), Gerald Kunzmann (NTT DOCOMO), Ryota Mibu (NEC), Carlos Goncalves (NEC), Arturo Martin De Nicolas (Ericsson)

History

Date	Description
04.12.2014	Project creation
20.04.2015	Initial version of the deliverable uploaded to gerrit
19.06.2015	Stable version of the Promise deliverable

Abstract Promise is an OPNFV requirement project. Its objective is to realize ETSI NFV defined resource reservation and NFVI capacity features within the scope of OPNFV. Promise provides the details of the requirements on resource reservation, NFVI capacity management at VIM, specification of the northbound interfaces from VIM relevant to these features, and implementation plan to realize these features in OPNFV.

Definition of terms

Different SDOs and communities use different terminology related to NFV/Cloud/SDN. This list tries to define an OPNFV terminology, mapping/translating the OPNFV terms to terminology used in other contexts.

Administrator Administrator of the system, e.g. OAM in Telco context.

Consumer User-side Manager; consumer of the interfaces produced by the VIM; VNFM, NFVO, or Orchestrator in ETSI NFV *[NFV003]* terminology.

NFV Network Function Virtualization

NFVI Network Function Virtualization Infrastructure; totality of all hardware and software components which build up the environment in which VNFs are deployed.

NFVO Network Functions Virtualization Orchestrator; functional block that manages the Network Service (NS) lifecycle and coordinates the management of NS lifecycle, VNF lifecycle (supported by the VNFM) and NFVI resources (supported by the VIM) to ensure an optimized allocation of the necessary resources and connectivity.

Physical resource Actual resources in NFVI; not visible to Consumer.

Resource zone A set of NFVI hardware and software resources logically grouped according to physical isolation and redundancy capabilities or to certain administrative policies for the NFVI *[NFVIFA010]*

VIM Virtualized Infrastructure Manager; functional block that is responsible for controlling and managing the NFVI compute, storage and network resources, usually within one operator's Infrastructure Domain, e.g. NFVI Point of Presence (NFVI-PoP).

Virtual Machine (VM) Virtualized computation environment that behaves very much like a physical computer/server.

Virtual network Virtual network routes information among the network interfaces of VM instances and physical network interfaces, providing the necessary connectivity.

Virtual resource A Virtual Machine (VM), a virtual network, or virtualized storage; Offered resources to "Consumer" as result of infrastructure virtualization; visible to Consumer.

Virtual Storage Virtualized non-volatile storage allocated to a VM.

VNF Virtualized Network Function. Implementation of an Network Function that can be deployed on a Network Function Virtualization Infrastructure (NFVI).

VNFM Virtualized Network Function Manager; functional block that is responsible for the lifecycle management of VNF.

CHAPTER ONE

INTRODUCTION

Resource reservation is a basic function for the operation of a virtualized telecom network. In resource reservation, VIM reserves resources for a certain period as requested by the NFVO. A resource reservation will have a start time which could be into the future. Therefore, the reserved resources shall be available for the NFVO requested purpose (e.g. for a VNF) at the start time for the duration asked by NFVO. Resources include all three resource types in an NFVI i.e. compute, storage and network.

Besides, NFVO requires abstracted NFVI resource capacity information in order to take decisions on VNF placement and other operations related to the virtual resources. VIM is required to inform the NFVO of NFVI resource state information for this purpose. Promise project aims at delivering the detailed requirements on these two features defined in ETSI NFV MAN GS *[NFVMAN]*, the list of gaps in upstream projects, potential implementation architecture and plan, and the VIM northbound interface specification for resource reservation and capacity management.

1.1 Problem description

OpenStack, a prominent candidate for the VIM, cannot reserve resources for future use. OpenStack requires immediate instantiation of Virtual Machines (VMs) in order to occupy resources intended to be reserved. Blazar can reserve compute resources for future by keeping the VMs in shelved mode. However, such reserved resources can also be used for scaling out rather than new VM instantiation. Blazar does not support network and storage resource reservation yet.

Besides, OpenStack does not provide a northbound interface through which it can notify an upper layer management entity e.g. NFVO about capacity changes in its NFVI, periodically or in an event driven way. Capacity management is a feature defined in ETSI NFV MAN GS *[NFVMAN]* and is required in network operation.

CHAPTER

TWO

USE CASES AND SCENARIOS

Resource reservation is a basic feature in any virtualization-based network operation. In order to perform such resource reservation from NFVO to VIM, NFVI capacity information is also necessary at the NFVO side. Below, four use cases to show typical requirements and solutions for capacity management and resource reservation is presented.

1. Resource capacity management
2. Resource reservation for immediate use
3. Resource reservation for future use
4. Co-existence of reservations and allocation requests without reservation

2.1 Resource capacity management

NFVO takes the first decision on in which NFVI it would instantiate a VNF. Along with NFVIs resource attributes (e.g. availability of hardware accelerators, particular CPU architectures etc.), NFVO needs to know available capacity of an NFVI in order to make an informed decision on selecting a particular NFVI. Such capacity information shall be in a coarser granularity than the respective VIM, as VIM maintains capacity information of its NFVI in fine details. However a very coarse granularity, like simply the number of available virtual CPU cores, may not be sufficient. In order to allow the NFVO to make well founded allocation decisions, an appropriate level to expose the available capacity may be per flavor. Capacity information may be required for the complete NFVI, or per partition or availability zone, or other granularities. Therefore, VIM requires to inform the NFVO about available capacity information regarding its NFVI at a pre-determined abstraction, either by a query-response, or in an event-based, or in a periodical way.

2.2 Resource reservation for immediate use

Reservation is inherently for the future. Even if some reserved resources are to be consumed instantly, there is a network latency between the issuance of a resource reservation request from the NFVO, a response from the VIM, and actual allocation of the requested resources to a VNF/VNFM. Within such latency, resource capacity in the NFVI in question could change, e.g., due to failure, allocation to a different request. Therefore, the response from a VIM to the NFVO to a resource reservation request for immediate use should have a validity period which shows until when this VIM can hold the requested resources. During this time, the NFVO should proceed to allocation if it wishes to consume the reserved requested. If allocation is not performed within the validity period, the response from VIM for a particular resource reservation request becomes invalid and VIM is not liable to provide those resources to NFVO/VNFM anymore. Reservations requests for immediate use do not have a start time but may have an end time.

2.3 Resource reservation for future use

Network operators may want to reserve extra resources for future use. Such necessity could arise from predicted congestion in telecom nodes e.g. due to local traffic spikes for concerts, natural disasters etc. In such a case, the NFVO, while sending a resource reservation request to the VIM, shall include a start time (and an end time if necessary). The start time indicates at what time the reserved resource shall be available to a designated consumer e.g. a VNF/VNFM. Here, the requirement is that the reserved resources shall be available when the start time arrives. After the start time has arrived, the reserved resources are allocated to the designated consumer(s). An explicit allocation request is needed. How actually these requested resources are held by the VIM for the period in between the arrival of the resource reservation request and the actual allocation is outside the scope of this requirement project.

2.4 Co-existence of reservations and allocation requests without reservation

In a real environment VIM will have to handle allocation requests without any time reference, i.e. time-unbound, together with time-bound reservations and allocation requests with an explicitly indicated end-time. A granted reservation for the future will effectively reduce the available capacity for any new time-unbound allocation request. The consequence is that reservations, even those far in the future, may result in denial of service for new allocation requests.

To alleviate this problem several approaches can be taken. They imply an implicit or explicit priority scheme:

- Allocation requests without reservation and which are time-unbound will be granted resources in a best-effort way: if there is instant capacity, but the resources may be later withdrawn due to the start time of a previously granted reservation
- Both allocation requests and reservation requests contain a priority which may be related to SLAs and contractual conditions between the tenant and the NFVI provider. Interactions may look like:
  - A reservation request for future use may cancel another, not yet started, reservation with lower priority
  - An allocation request without reservations and time-unbound [1] may be granted resources and prevent a future reservation with lower priority from getting resources at start time
  - A reservation request may result in terminating resources allocated to a request with no reservation, if the latter has lower priority

[1] In this case, the consumer (VNFM or NFVO) requests to immediately instantiate and assign virtualized resources without having reserved the resources beforehand

CHAPTER

THREE

HIGH LEVEL ARCHITECTURE AND GENERAL FEATURES

3.1 Architecture Overview

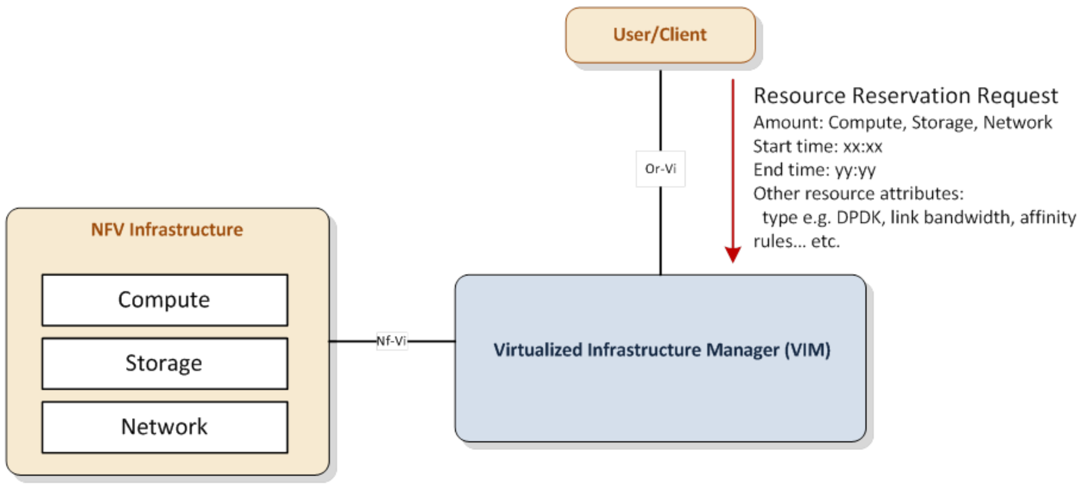

User/Client
Resource Reservation Request
Amount: Compute, Storage, Network
Start time: xx:xx
End time: yy:yy
Other resource attributes:
type e.g. DPDK, link bandwidth, affinity
rules... etc.
Or-Vi
NFV Infrastructure
Compute
Storage
Network
Nf-Vi
Virtualized Infrastructure Manager (VIM)

Fig. 3.1: Resource Reservation Architecture

Fig. 3.1 shows the high level architecture for the resource reservation use cases. Reserved resources are guaranteed for a given user/client for the period expressed by start and end time. User/client represents the requestor and the consequent consumer of the reserved resources and correspond to the NFVO or VNFM in ETSI NFV terminology.

Note: in this document only reservation requests from NFVO are considered.

3.2 General Features

This section provides a list of features that need to be developed in the Promise project.

- Resource capacity management
  - Discovery of available resource capacity in resource providers
  - Monitoring of available resource capacity in resource providers
  - Update available resource capacity as a result of new or expired reservations, addition/removal of resources. Note: this is a VIM internal function, not an operation in the VIM northbound interface.
- Resource reservation

  - Set start time and end time for allocation
  - Increase/decrease reserved resource's capacity
  - Update resource reservations, e.g. add/remove reserved resources
  - Terminate an allocated resource due to the end time of a reservation

- VIM northbound interfaces
  - Receive/Reply resource reservation requests
  - Receive/Reply resource capacity management requests
  - Receive/Reply resource allocation requests for reserved resources when start time arrives
  - Subscribe/Notify resource reservation event
    * Notify reservation error or process completion prior to reservation start
    * Notify remaining time until termination of a resource due to the end time of a reservation
    * Notify termination of a resource due to the end time of a reservation
  - Receive/Reply queries on available resource capacity
  - Subscribe/Notify changes in available resource capacity

3.3 High level northbound interface specification

3.3.1 Resource Capacity Management

Fig. 3.2 shows a high level flow for a use case of resource capacity management. In this example, the VIM notifies the NFVO of capacity change after having received an event regarding a change in capacity (e.g. a fault notification) from the NFVI. The NFVO can also retrieve detailed capacity information using the Query Capacity Request interface operation.

Fig. 3.3 shows a high level flow for another use case of resource capacity management. In this example, the NFVO queries the VIM about the actual capacity to instantiate a certain resource according to a certain template, for example a VM according to a certain flavor. In this case the VIM responds with the number of VMs that could be instantiated according to that flavor with the currently available capacity.

3.3.2 Resource Reservation

Fig. 3.4 shows a high level flow for a use case of resource reservation. The main steps are:

- The NFVO sends a resource reservation request to the VIM using the Create Resource Reservation Request interface operation.
- The NFVO gets a reservation identifier reservation associated with this request in the reply message
- Using the reservation identifier reservation, the NFVO can query/update/terminate a resource reservation using the corresponding interface operations
- The NFVO is notified that the resource reservation is terminated due to the end time of the reservation

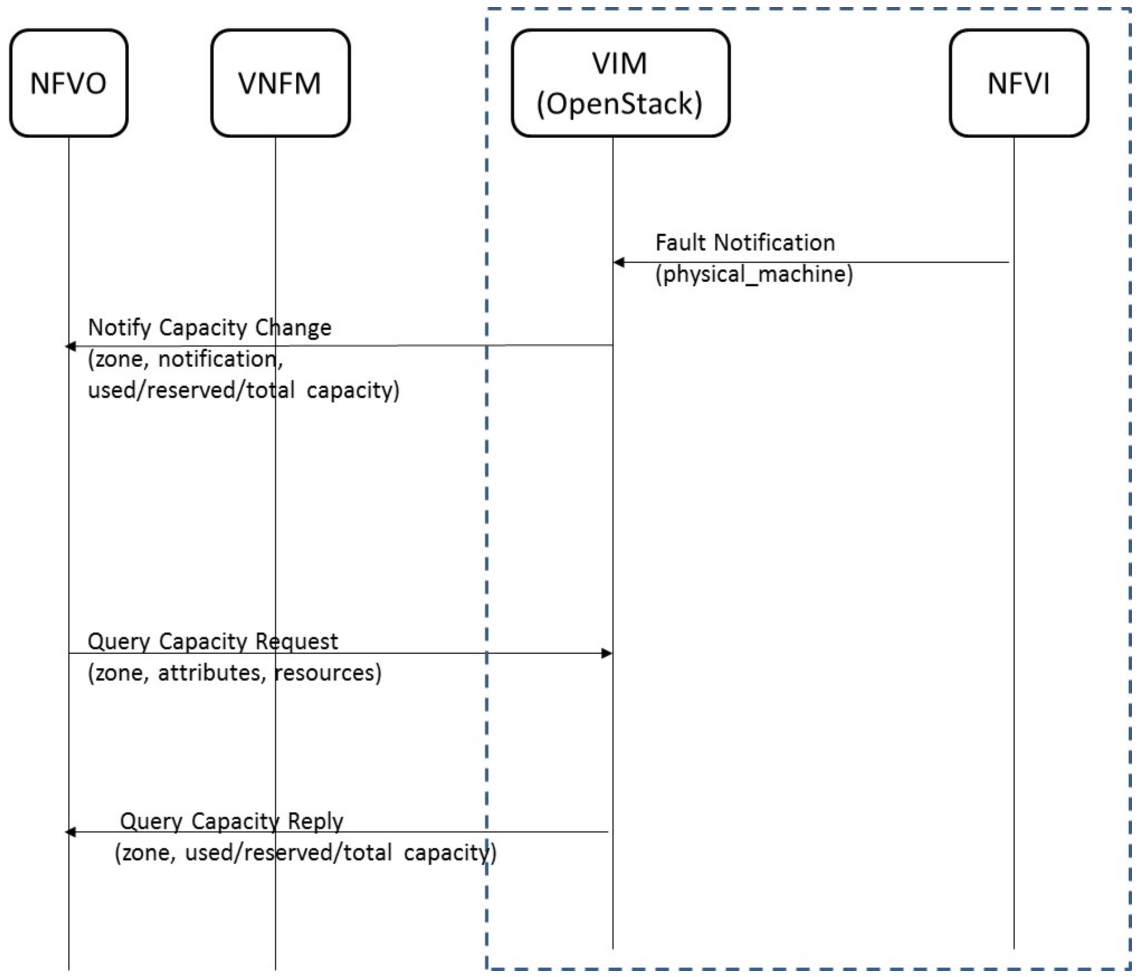

NFVO
VNFM
VIM (OpenStack)
NFVI
Fault Notification (physical_machine)
Notify Capacity Change (zone, notification, used/reserved/total capacity)
Query Capacity Request (zone, attributes, resources)
Query Capacity Reply (zone, used/reserved/total capacity)

Fig. 3.2: Resource capacity management message flow: notification of capacity change

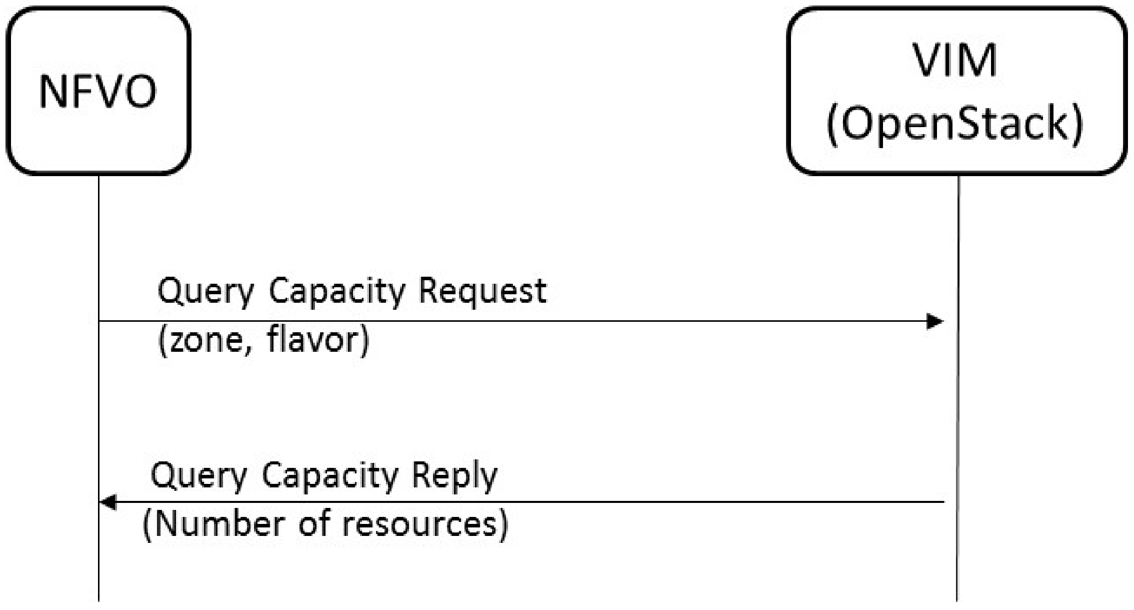

NFVO
VIM (OpenStack)
Query Capacity Request (zone, flavor)
Query Capacity Reply (Number of resources)

Fig. 3.3: Resource capacity management message flow: query of capacity density

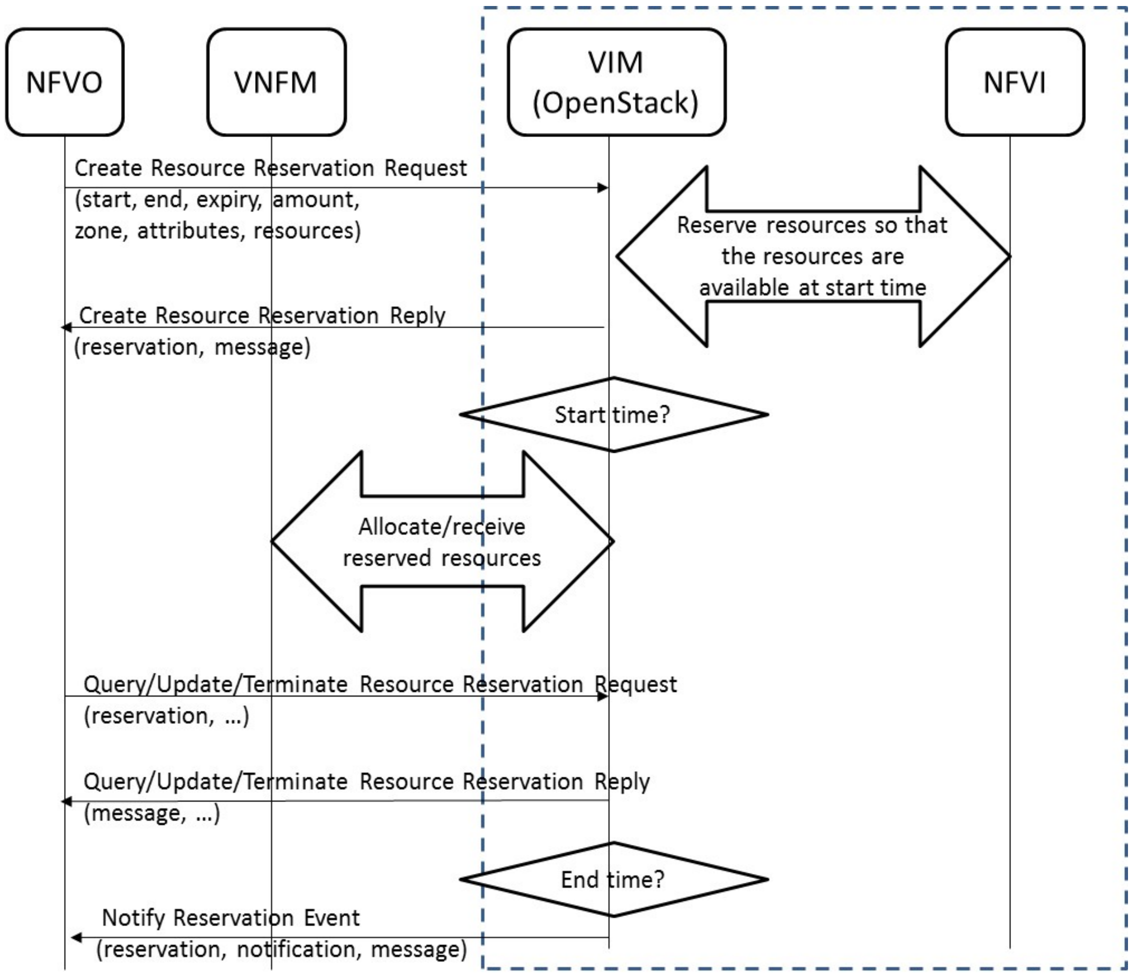

NFVO
VNFM
VIM (OpenStack)
NFVI
Create Resource Reservation Request (start, end, expiry, amount, zone, attributes, resources)
Reserve resources so that the resources are available at start time
Create Resource Reservation Reply (reservation, message)
Start time?
Allocate/receive reserved resources
Query/Update/Terminate Resource Reservation Request (reservation, …)
Query/Update/Terminate Resource Reservation Reply (message, …)
End time?
Notify Reservation Event (reservation, notification, message)

Fig. 3.4: Resource reservation flow

3.4 Information elements

3.4.1 Resource Capacity Management

Notify Capacity Change Event

The notification change message shall include the following information elements:

Name	Type	Description
Notification	Identifier	Identifier issued by the VIM for the capacity change event notification
Zone	Identifier	Identifier of the zone where capacity has changed
Used/Reserved/Total Capacity	List	Used, reserved and total capacity information regarding the resource items subscribed for notification for which capacity change event occurred

Query Resource Capacity Request

The capacity management query request message shall include the following information elements:

Name	Type	Description
Zone	Identifier	Identifier of the zone where capacity is requested
Attributes	List	Attributes of resource items to be notified regarding capacity change events
Resources	List	Identifiers of existing resource items to be queried regarding capacity info (such as images, flavors, virtual containers, networks, physical machines, etc.)

The capacity management query request message may also include the following information element:

Name	Type	Description
Flavor	Identifier	Identifier that is passed in the request to obtain information of the number of virtual resources that can be instantiated according to this flavor with the available capacity

Query Resource Capacity Reply

The capacity management query reply message shall include the following information elements:

Name	Type	Description
Zone	Identifier	Identifier of the zone where capacity is requested
Used/Reserved/Total Capacity	List	Used, reserved and total capacity information regarding each of the resource items requested to check for capacity

The detailed specification of the northbound interface for Capacity Management in provided in section 5.1.1.

3.4.2 Resource Reservation

Create Resource Reservation Request

The create resource reservation request message shall include the following information elements:

Name	Type	Description
Start	Timestamp	Start time for consumption of the reserved resources
End	Timestamp	End time for consumption of the reserved resources
Expiry	Timestamp	If not all reserved resources are allocated between start time and expiry, the VIM shall release the corresponding resources [1]
Amount	Number	Amount of the resources per resource item type (i.e. compute/network/storage) that need to be reserved
Zone	Identifier	The zone where the resources need(s) to be reserved
Attributes	List	Attributes of the resources to be reserved such as DPDK support, hypervisor, network link bandwidth, affinity rules, etc.
Resources	List	Identifiers of existing resource items to be reserved (such as images, flavors, virtual containers, networks, physical machines, etc.)

Create Resource Reservation Reply

The create resource reservation reply message shall include the following information elements:

Name	Type	Description
Reservation	Identifier	Identification of the reservation instance. It can be used by a consumer to modify the reservation later, and to request the allocation of the reserved resources.
Message	Text	Output message that provides additional information about the create resource reservation request (e.g. may be a simple ACK if the request is being background processed by the VIM)

Notify Reservation Event

The notification reservation event message shall include the following information elements:

Name	Type	Description
Reservation	Identifier	Identification of the reservation instance triggering the event
Notification	Identifier	Identification of the resource event notification issued by the VIM
Message	Text	Message describing the event

The detailed specification of the northbound interface for Resource Reservation is provided in section 5.1.2.

[1] Expiry is a period around start time within which, the allocation process must take place. If allocation process does not start within the expiry period, the reservation becomes invalid and VIM should release the resources

CHAPTER

FOUR

GAP ANALYSIS IN UPSTREAM PROJECTS

This section provides a list of gaps in upstream projects for realizing resource reservation and management. The gap analysis work focuses on the current OpenStack Blazar project *[BLAZAR]* in this first release.

4.1 OpenStack

4.1.1 Resource reservation for future use

- Category: Blazar
- Type: 'missing' (lack of functionality)
- Description:
  - To-be: To reserve a whole set of compute/storage/network resources in the future
  - As-is: Blazar currently can do only compute resource reservation by using "Shelved VM"
- Related blueprints:
  - https://blueprints.launchpad.net/blazar/+spec/basic-volume-plugin
  - https://blueprints.launchpad.net/blazar/+spec/basic-network-plugin
  - It was planned in Blazar to implement volume and network/fixed ip reservations

4.1.2 Resource reservation update

- Category: Blazar
- Type: 'missing' (lack of functionality)
- Description:
  - To-be: Have the possibility of adding/removing resources to an existing reservation, e..g in case of NFVI failure
  - As-is: Currently in Blazar, a reservation can only be modified in terms of start/end time
- Related blueprints: N/A

4.1.3 Give me an offer

- Category: Blazar
- Type: 'missing' (lack of functionality)
- Description:
  - To-be: To have the possibility of giving a quotation to a requesting user and an expiration time. Reserved resources shall be released if they are not claimed before this expiration time.
  - As-is: Blazar can already send notification e.g. to inform a given user that a reservation is about to expire
- Related blueprints: N/A

4.1.4 StormStack StormForge

Stormify

- Stormify enables rapid web applications construction
- Based on Ember.js style Data stores
- Developed on Node.js using coffeescript/javascript
- Auto RESTful API generation based on Data Models
- Development starts with defining Data Models
- Code hosted at github : http://github.com/stormstack/stormify

StormForge

- Data Model driven management of Resource Providers
- Based on Stormify Framework and implemented as per the OPNFV Promise requirements
- Data Models are auto generated and RESTful API code from YANG schema
- Currently planned key services include Resource Capacity Management Service and Resource Reservation Service
- List of YANG schemas for Promise project is attached in the Appendix
- Code hosted at github: http://github.com/stormstack/stormforge

Resource Discovery

- Category: StormForge
- Type: 'planning' (lack of functionality)
- Description
  - To-be: To be able to discover resources in real time from OpenStack components. Planning to add OpenStack Project to interface with Promise for real time updates on capacity or any failures
  - As-is: Currently, resource capacity is learnt using NB APIs related to quota
- Related Blueprints: N/A

CHAPTER

FIVE

DETAILED ARCHITECTURE AND MESSAGE FLOWS

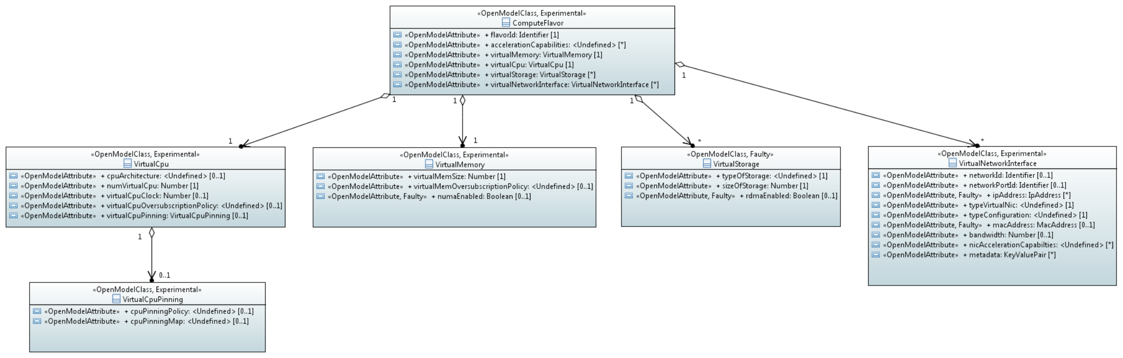
5.1 Detailed northbound interface specification

Note: This is Work in Progress.

5.1.1 ETSI NFV IFA Information Models

Compute Flavor

A compute flavor includes information about number of virtual CPUs, size of virtual memory, size of virtual storage, and virtual network interfaces *[NFVIFA005]*


«OpenModelClass, Experimental»
ComputeFlavor
«OpenModelAttribute» + flavorId: Identifier [1]
«OpenModelAttribute» + accelerationCapabilities: <Undefined> [*]
«OpenModelAttribute» + virtualMemory: VirtualMemory [1]
«OpenModelAttribute» + virtualCpu: VirtualCpu [1]
«OpenModelAttribute» + virtualStorage: VirtualStorage [*]
«OpenModelAttribute» + virtualNetworkInterface: VirtualNetworkInterface [*]
1
1
1
1
1
1
1
*
*
«OpenModelClass, Experimental»
VirtualCpu
«OpenModelAttribute» + cpuArchitecture: <Undefined> [0..1]
«OpenModelAttribute» + numVirtualCpu: Number [1]
«OpenModelAttribute» + virtualCpuClock: Number [0..1]
«OpenModelAttribute» + virtualCpuOversubscriptionPolicy: <Undefined> [0..1]
«OpenModelAttribute» + virtualCpuPinning: VirtualCpuPinning [0..1]
«OpenModelClass, Experimental»
VirtualMemory
«OpenModelAttribute» + virtualMemSize: Number [1]
«OpenModelAttribute» + virtualMemOversubscriptionPolicy: <Undefined> [0..1]
«OpenModelAttribute, Faulty» + numaEnabled: Boolean [0..1]
«OpenModelClass, Faulty»
VirtualStorage
«OpenModelAttribute» + typeOfStorage: <Undefined> [1]
«OpenModelAttribute» + sizeOfStorage: Number [1]
«OpenModelAttribute, Faulty» + rdmaEnabled: Boolean [0..1]
«OpenModelClass, Experimental»
VirtualNetworkInterface
«OpenModelAttribute» + networkId: Identifier [0..1]
«OpenModelAttribute» + networkPortId: Identifier [0..1]
«OpenModelAttribute, Faulty» + ipAddress: IpAddress [*]
«OpenModelAttribute» + typeVirtualNic: <Undefined> [1]
«OpenModelAttribute» + typeConfiguration: <Undefined> [1]
«OpenModelAttribute, Faulty» + macAddress: MacAddress [0..1]
«OpenModelAttribute» + bandwidth: Number [0..1]
«OpenModelAttribute» + nicAccelerationCapabilties: <Undefined> [*]
«OpenModelAttribute» + metadata: KeyValuePair [*]
1
0..1
«OpenModelClass, Experimental»
VirtualCpuPinning
«OpenModelAttribute» + cpuPinningPolicy: <Undefined> [0..1]
«OpenModelAttribute» + cpuPinningMap: <Undefined> [0..1]

5.1.2 Virtualised Compute Resources

Compute Capacity Management

Subscribe Compute Capacity Change Event

Subscription from Consumer to VIM to be notified about compute capacity changes

`POST /capacity/compute/subscribe`

Example request:

```
 POST /capacity/compute/subscribe HTTP/1.1
 Accept: application/json

 {
    "zoneId": "12345",
    "resourceDescriptor": [
        {
           "computeResourceTypeId": "vcInstances"
        }
    ],
    "threshold": [
        {
           "capacity_info": "available",
           "condition": "lt",
           "value": 5
        }
    ]
}
```

Example response:

```
HTTP/1.1 201 CREATED
Content-Type: application/json

{
   "created": "2015-09-21T00:00:00Z",
   "capacityChangeSubscriptionId": "abcdef-ghijkl-123456789"
}
```

Status Codes

- 400 Bad Request – resourceDescriptor is missing

Query Compute Capacity

Request to find out about available, reserved, total and allocated compute capacity.

GET /capacity/compute/query

Example request:

```
GET /capacity/compute/query HTTP/1.1
Accept: application/json

{
  "zoneId": "12345",
  "resourceDescriptor":  {
       "computeResourceTypeId": "vcInstances"
  },
  "timePeriod":  {
       "startTime": "2015-09-21T00:00:00Z",
       "stopTime": "2015-09-21T00:05:30Z"
  }
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json
```

```
{
   "zoneId": "12345",
   "lastUpdate": "2015-09-21T00:03:20Z",
   "capacityInformation": {
      "available": 4,
      "reserved": 17,
      "total": 50,
      "allocated": 29
   }
}
```

Query Parameters

- **`limit`** – Default is 10.

Status Codes

- 404 Not Found – resource zone unknown

Notify Compute Capacity Change Event

Notification about compute capacity changes

`POST /capacity/compute/notification`

Example notification:

```
Content-Type: application/json

{
     "zoneId": "12345",
     "notificationId": "zyxwvu-tsrqpo-987654321",
     "capacityChangeTime": "2015-09-21T00:03:20Z",
     "resourceDescriptor": {
        "computeResourceTypeId": "vcInstances"
     },
     "capacityInformation": {
        "available": 4,
        "reserved": 17,
        "total": 50,
        "allocated": 29
     }
}
```

Compute Resource Reservation

Create Compute Resource Reservation

Request the reservation of compute resource capacity

`POST /reservation/compute/create`

Example request:

```
POST /reservation/compute/create HTTP/1.1
Accept: application/json

{
    "startTime": "2015-09-21T01:00:00Z",
    "computePoolReservation": {
        "numCpuCores": 20,
        "numVcInstances": 5,
        "virtualMemSize": 10
    }
}
```

Example response:

```
HTTP/1.1 201 CREATED
Content-Type: application/json

{
   "reservationData": {
      "startTime": "2015-09-21T01:00:00Z",
      "reservationStatus": "initialized",
      "reservationId": "xxxx-yyyy-zzzz",
      "computePoolReserved": {
          "numCpuCores": 20,
          "numVcInstances": 5,
          "virtualMemSize": 10,
          "zoneId": "23456"
      }
   }
}
```

and/or virtualized containers

`POST reservation/compute/create`

Example request:

```
POST /reservation/compute/create HTTP/1.1
Accept: application/json

{
  "startTime": "2015-10-05T15:00:00Z",
  "virtualizationContainerReservation": [
    {
       "containerId": "myContainer",
       "containerFlavor": {
          "flavorId": "myFlavor",
          "virtualCpu": {
             "numVirtualCpu": 2,
             "cpuArchitecture": "x86"
          },
          "virtualMemory": {
              "numaEnabled": "False",
              "virtualMemSize": 16
          },
          "virtualStorage": {
              "typeOfStorage": "volume",
              "sizeOfStorage": 16
          }
       }
    }
  ]
}
```

Example response:

```
HTTP/1.1 201 CREATED
Content-Type: application/json

{
   "reservationData": {
      "startTime": "2015-10-05T15:00:00Z",
      "reservationId": "aaaa-bbbb-cccc",
      "reservationStatus": "initialized",
      "virtualizationContainerReserved": [
          {
             "containerId": "myContainer",
             "containerFlavor": {
                 "flavorId": "myFlavor",
                 "virtualCpu": {
                    "numVirtualCpu": 2,
                    "cpuArchitecture": "x86"
                 },
                 "virtualMemory": {
                    "numaEnabled": "False",
                    "virtualMemSize": 16
                 },
                 "virtualStorage": {
                     "typeOfStorage": "volume",
                     "sizeOfStorage": 16
                 }
             }
          }
      ]
   }
}
```

Query Compute Resource Reservation

Request to find out about reserved compute resources that the consumer has access to.

GET /reservation/compute/query

Example request:

```
GET /reservation/compute/query HTTP/1.1
Accept: application/json

{
   "queryReservationFilter": [
       {
           "reservationId": "xxxx-yyyy-zzzz"
       }
   ]

}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
   "reservationData":
   {
      "startTime": "2015-09-21T01:00:00Z",
      "reservationStatus": "active",
      "reservationId": "xxxx-yyyy-zzzz",
      "computePoolReserved":
      {
          "numCpuCores": 20,
          "numVcInstances": 5,
          "virtualMemSize": 10,
          "zoneId": "23456"
      }
   }
}
```

Status Codes

- 404 Not Found – reservation id unknown

Update Compute Resource Reservation

Request to update compute resource reservation

`POST /reservation/compute/update`

Example request:

```
POST /reservation/compute/update HTTP/1.1
Accept: application/json

{
    "startTime": "2015-09-14T16:00:00Z",
    "reservationId": "xxxx-yyyy-zzzz"
}
```

Example response:

```
HTTP/1.1 201 CREATED
Content-Type: application/json

{
  "reservationData": {
      "startTime": "2015-09-14TT16:00:00Z",
      "reservationStatus": "active",
      "reservationId": "xxxx-yyyy-zzzz",
      "computePoolReserved": {
          "numCpuCores": 20,
          "numVcInstances": 5,
          "virtualMemSize": 10,
          "zoneId": "23456"
      }
   }
}
```

Terminate Compute Resource Reservation

Request to terminate a compute resource reservation

DELETE /reservation/compute/*(reservation_id)*

5.1.3 Virtualised Network Resources

Network Capacity Management

Subscribe Network Capacity Change Event

Susbcription from Consumer to VIM to be notified about network capacity changes

POST /capacity/network/subscribe

Example request:

```
POST /capacity/network/subscribe HTTP/1.1
Accept: application/json

{
   "resourceDescriptor": [
        {
            "networkResourceTypeId": "publicIps"
        }
   ],
   "threshold": [
       {
          "capacity_info": "available",
          "condition": "lt",
          "value": 5
       }
   ]
```

}

Example response:

```
HTTP/1.1 201 CREATED
Content-Type: application/json

{
   "created": "2015-09-28T00:00:00Z",
   "capacityChangeSubscriptionId": "bcdefg-hijklm-234567890"
}
```

Query Network Capacity

Request to find out about available, reserved, total and allocated network capacity.

GET /capacity/network/query

Example request:

```
 GET /capacity/network/query HTTP/1.1
 Accept: application/json

 {
   "resourceDescriptor":  {
       "networkResourceTypeId": "publicIps"
   },
   "timePeriod":  {
       "startTime": "2015-09-28T00:00:00Z",
       "stopTime": "2015-09-28T00:05:30Z"
   }
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "lastUpdate": "2015-09-28T00:02:10Z",
  "capacityInformation": {
      "available": 4,
      "reserved": 10,
      "total": 64,
      "allocated": 50
  }
}
```

Notify Network Capacity Change Event

Notification about network capacity changes

POST /capacity/network/notification

Example notification:

```
 Content-Type: application/json

 {
    "notificationId": "yxwvut-srqpon-876543210",
    "capacityChangeTime": "2015-09-28T00:02:10Z",
    "resourceDescriptor": {
       "networkResourceTypeId": "publicIps"
    },
    "capacityInformation": {
      "available": 4,
      "reserved": 10,
      "total": 64,
      "allocated": 50
    }
}
```

Network Resource Reservation

Create Network Resource Reservation

Request the reservation of network resource capacity and/or virtual networks, network ports

POST /reservation/network/create

Example request:

```
POST /reservation/network/create HTTP/1.1
Accept: application/json
```

```
{
   "startTime": "2015-09-28T01:00:00Z",
   "networkReservation": {
      "numPublicIps": 2
   }
}
```

Example response:

```
HTTP/1.1 201 CREATED
Content-Type: application/json

{
   "reservationData": {
       "startTime": "2015-09-28T01:00:00Z",
       "reservationStatus": "initialized",
       "reservationId": "wwww-xxxx-yyyy",
       "networkReserved": {
           "publicIps": [
               "10.2.91.60",
               "10.2.91.61"
           ]
       }
   }
}
```

Query Network Resource Reservation

Request to find out about reserved network resources that the consumer has access to.

GET /reservation/network/query

Example request:

```
GET /reservation/network/query HTTP/1.1
Accept: application/json

{
  "queryReservationFilter": [
     {
        "reservationId": "wwww-xxxx-yyyy"
     }
  ]
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
   "reservationData": {
      "startTime": "2015-09-28T01:00:00Z",
      "reservationStatus": "active",
      "reservationId": "wwww-xxxx-yyyy",
      "networkReserved": "publicIps": [
        "10.2.91.60",
        "10.2.91.61"
      ]
```

```
        }
    }
```

Update Network Resource Reservation

Request to update network resource reservation

`POST /reservation/network/update`

Example request:

```
POST /reservation/network/update HTTP/1.1
Accept: application/json

{
   "startTime": "2015-09-21T16:00:00Z",
   "reservationId": "wwww-xxxx-yyyy"
}
```

Example response:

```
HTTP/1.1 201 CREATED
Content-Type: application/json

{
   "reservationData": {
       "startTime": "2015-09-21T16:00:00Z",
       "reservationStatus": "active",
       "reservationId": "wwww-xxxx-yyyy",
       "networkReserved": {
           "publicIps": [
               "10.2.91.60",
               "10.2.91.61"
           ]
       }
   }
}
```

Terminate Network Resource Reservation

Request to terminate a network resource reservation

`DELETE /reservation/network/`(*reservation_id*)

5.1.4 Virtualised Storage Resources

Storage Capacity Management

Subscribe Storage Capacity Change Event

Subscription from Consumer to VIM to be notified about storage capacity changes

`POST /capacity/storage/subscribe`

Example request:

```
POST /capacity/storage/subscribe HTTP/1.1
Accept: application/json

{
   "resourceDescriptor": [
      {
         "storageResourceTypeId": "volumes"
      }
   ],
   "threshold": [
      {
        "capacity_info": "available",
        "condition": "lt",
        "value": 3
      }
   ]
}
```

Example response:

```
HTTP/1.1 201 CREATED
Content-Type: application/json

{
   "created": "2015-09-28T12:00:00Z",
   "capacityChangeSubscriptionId": "cdefgh-ijklmn-345678901"
}
```

Query Storage Capacity

Request to find out about available, reserved, total and allocated storage capacity.

GET /capacity/storage/query

Example request:

```
GET /capacity/storage/query HTTP/1.1
Accept: application/json

{
   "resourceDescriptor": {
      "storageResourceTypeId": "volumes"
   },
   "timePeriod":  {
     "startTime": "2015-09-28T12:00:00Z",
     "stopTime": "2015-09-28T12:04:45Z"
   }
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "lastUpdate": "2015-09-28T12:01:35Z",
    "capacityInformation": {
       "available": 2,
       "reserved": 4,
            "total": 10,
            "allocated": 4
        }
    }
```

Notify Storage Capacity Change Event

Notification about storage capacity changes

POST /capacity/storage/notification

Example notification:

```
Content-Type: application/json

{
   "notificationId": "xwvuts-rqponm-765432109",
   "capacityChangeTime": "2015-09-28T12:01:35Z",
   "resourceDescriptor": {
       "storageResourceTypeId": "volumes"
   },
   "capacityInformation": {
      "available": 2,
      "reserved": 4,
      "total": 10,
      "allocated": 4
   }
}
```

Storage Resource Reservation

Create Storage Resource Reservation

Request the reservation of storage resource capacity

POST /reservation/storage/create

Example request:

```
POST /reservation/storage/create HTTP/1.1
Accept: application/json

{
    "startTime": "2015-09-28T13:00:00Z",
    "storagePoolReservation": {
       "storageSize": 10,
       "numSnapshots": 3,
       "numVolumes": 2
    }
}
```

Example response:

```
HTTP/1.1 201 CREATED
Content-Type: application/json

{
    "reservationData": {
        "startTime": "2015-09-28T13:00:00Z",
        "reservationStatus": "initialized",
        "reservationId": "vvvv-wwww-xxxx",
        "storagePoolReserved": {
           "storageSize": 10,
           "numSnapshots": 3,
           "numVolumes": 2
        }
    }
}
```

Query Storage Resource Reservation

Request to find out about reserved storage resources that the consumer has access to.

GET /reservation/storage/query

Example request:

```
GET /reservation/storage/query HTTP/1.1
Accept: application/json

{
   "queryReservationFilter": [
       {
          "reservationId": "vvvv-wwww-xxxx"
       }
   ]
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
   "reservationData": {
      "startTime": "2015-09-28T13:00:00Z",
      "reservationStatus": "active",
      "reservationId": "vvvv-wwww-xxxx",
      "storagePoolReserved": {
           "storageSize": 10,
           "numSnapshots": 3,
           "numVolumes": 2
      }
   }
}
```

Update Storage Resource Reservation

Request to update storage resource reservation

POST /reservation/storage/update

Example request:

```
POST /reservation/storage/update HTTP/1.1
Accept: application/json
```

```
{
   "startTime": "2015-09-20T23:00:00Z",
   "reservationId": "vvvv-wwww-xxxx"
}
```

Example response:

```
HTTP/1.1 201 CREATED
Content-Type: application/json

{
   "reservationData": {
       "startTime": "2015-09-20T23:00:00Z",
       "reservationStatus": "active",
       "reservationId": "vvvv-wwww-xxxx",
       "storagePoolReserved": {
           "storageSize": 10,
           "numSnapshots": 3,
           "numVolumes": 2
       }
   }
}
```

Terminate Storage Resource Reservation

Request to terminate a storage resource reservation

DELETE /reservation/storage/(*reservation_id*)

5.2 Detailed Message Flows

Fig. 5.1 shows a detailed message flow between the consumers and the functional blocks inside the VIM and has the following steps:

Step 1: The consumer subscribes to capacity change notifications

Step 2: The Capacity Manager monitors the capacity information for the various types of resources by querying the various Controllers (e.g. Nova, Neutron, Cinder), either periodically or on demand and updates capacity information in the Capacity Map

Step 3: Capacity changes are notified to the consumer

Step 4: The consumer queries the Capacity Manager to retrieve capacity detailed information

Fig. 5.2 shows a detailed message flow between the consumers and the functional blocks inside the VIM and has the following steps:

Step 1: The consumer creates a resource reservation request for future use by setting a start and end time for the allocation

Step 2: The consumer gets an immediate reply with a reservation status message "reservationStatus" and an identifier to be used with this reservation instance "reservationID"

Step 3: The consumer subscribes to reservation notification events

Step 4: The Resource Reservation Manager checks the feasibility of the reservation request by consulting the Capacity Manager

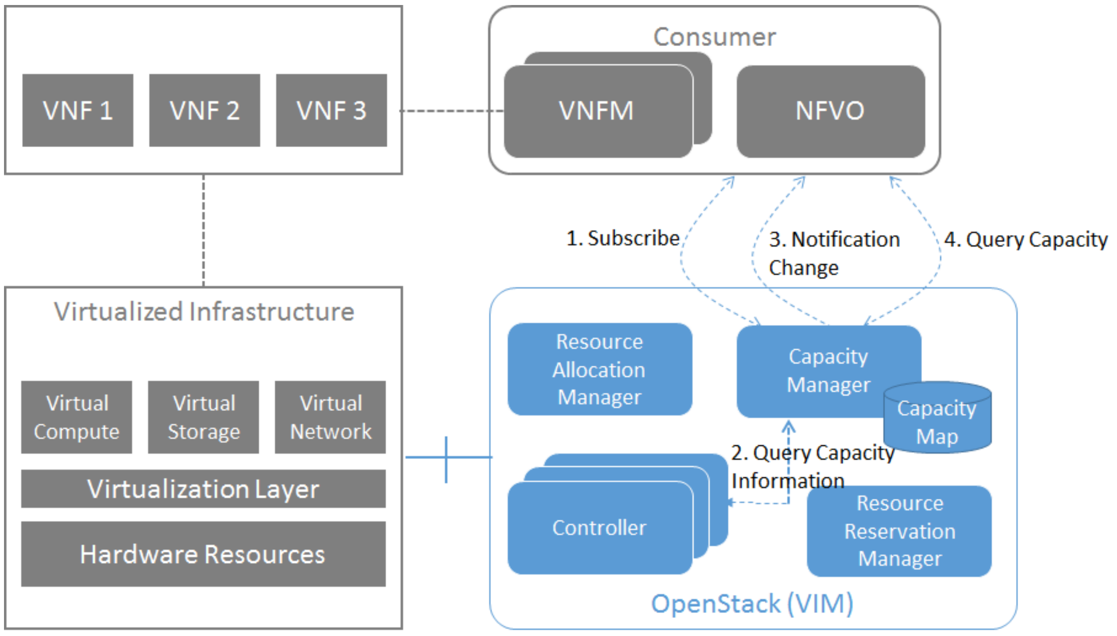


Fig. 5.1: Capacity Management Scenario

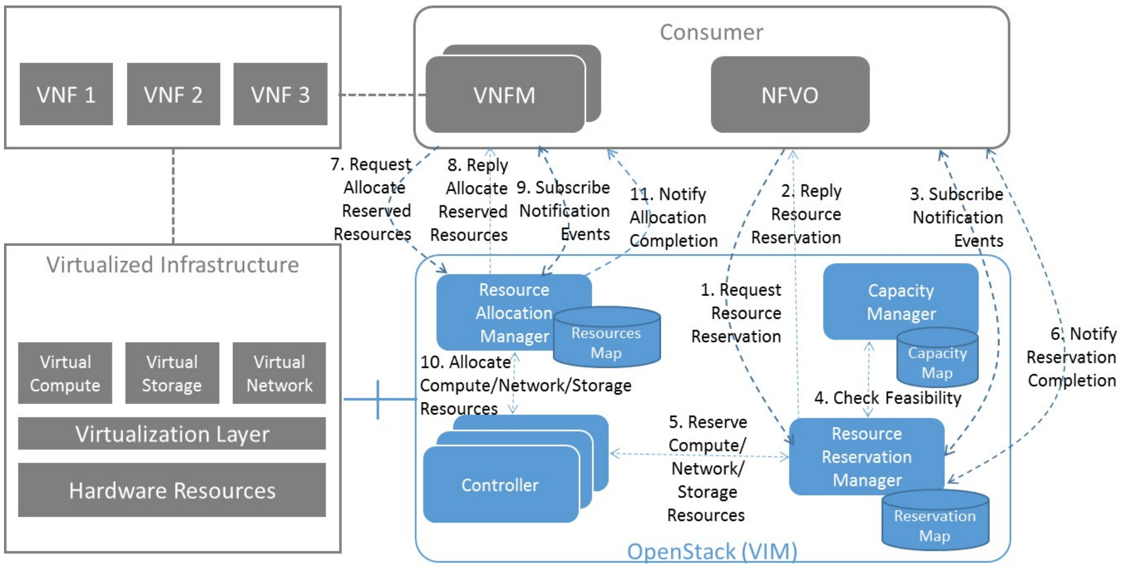

Consumer
VNF 1
VNF 2
VNF 3
VNFM
NFVO
7. Request Allocate Reserved Resources
8. Reply Allocate Reserved Resources
9. Subscribe Notification Events
11. Notify Allocation Completion
2. Reply Resource Reservation
3. Subscribe Notification Events
6. Notify Reservation Completion
Virtualized Infrastructure
Virtual Compute
Virtual Storage
Virtual Network
Virtualization Layer
Hardware Resources
Resource Allocation Manager
Resources Map
10. Allocate Compute/Network/Storage Resources
1. Request Resource Reservation
Capacity Manager
Capacity Map
4. Check Feasibility
5. Reserve Compute/ Network/ Storage Resources
Controller
Resource Reservation Manager
Reservation Map
OpenStack (VIM)

Fig. 5.2: Resource Reservation for Future Use Scenario

Step 5: The Resource Reservation Manager reserves the resources and stores the list of reservations IDs generated by the Controllers (e.g. Nova, Neutron, Cinder) in the Reservation Map

Step 6: Once the reservation process is completed, the VIM sends a notification message to the consumer with information on the reserved resources

Step 7: When start time arrives, the consumer creates a resource allocation request.

Step 8: The consumer gets an immediate reply with an allocation status message "allocationStatus".

Step 9: The consumer subscribes to allocation notification events

Step 10: The Resource Allocation Manager allocates the reserved resources. If not all reserved resources are allocated before expiry, the reserved resources are released and a notification is sent to the consumer

Step 11: Once the allocation process is completed, the VIM sends a notification message to the consumer with information on the allocated resources

CHAPTER

SIX

SUMMARY AND CONCLUSION

Resource Reservation and Resource Capacity Management are features to be supported by the VIM and exposed to the consumer via the VIM NBI. These features have been specified by ETSI NFV.

This document has described several use cases and corresponding high level flows where Resource Reservation and Capacity Management are of great benefit for the consumer of the virtualised resource management interface: the NFVO or the VNFM. The use cases include:

- Notification of changes in capacity in the NFVI
- Query of available resource capacity
- Reservation of a resource or set of resources for immediate use
- Reservation of a resource or set of resources for future use

The Promise project has performed a gap analysis in order to fulfil the required functionality. Based on the gap analysis an implementation plan and way forward has been proposed, including a possible design architecture and high level information model. Immediate next steps of this project is to deliver a working Proof-of-Concepts (PoC) and engage upstream communities to fill out the gaps identified by Promise.

CHAPTER

SEVEN

ANNEX A: PROMISE YANG SCHEMA BASED ON YANGFORGE

```
module opnfv-promise {
namespace "urn:opnfv:promise";
prefix promise;

import complex-types { prefix ct; }
import iana-crypt-hash { prefix ianach; }
import ietf-inet-types { prefix inet; }
import ietf-yang-types { prefix yang; }
import opnfv-promise-vim { prefix vim; }

feature multi-provider {
  description "";
}

description
  "OPNFV Promise Resource Reservation/Allocation controller module";

revision 2015-04-16 {
  description "Initial revision.";
}

revision 2015-08-06 {
  description "Updated to incorporate YangForge framework";
}

grouping resource-capacity {
  container capacity {
    container quota { description 'Conceptual container that should be extended'; }
    container usage { description 'Conceptual container that should be extended';
                      config false; }
    container reserved { description 'Conceptual container that should be extended';
                      config false; }
    container available { description 'Conceptual container that should be extended';
                      config false; }
  }
}

grouping compute-capacity {
  leaf cores { type number; }
  leaf ram { type number; }
  leaf instances { type number; }
}

grouping networking-capacity {
  leaf network { type number; }
  leaf port { type number; }
  leaf router { type number; }
  leaf subnet { type number; }
  leaf address { type number; }
}

ct:complex-type ResourceReservation {
  ct:extends vim:ResourceElement;

  description
    "Contains the capacities of various resource services being reserved
     along with any resource elements needed to be available at
     the time of allocation(s).";

  reference "OPNFV-PROMISE, Section 3.4.1";

  leaf start { type yang:date-and-time; }
  leaf end   { type yang:date-and-time; }
  leaf expiry {
    description "Duration in seconds from start when unallocated reserved resources
                 will be released back into the pool";
    type number; units "seconds";
  }
  leaf zone { type instance-identifier { ct:instance-type vim:AvailabilityZone; } }
  container capacity {
    uses vim:compute-capacity;
    uses vim:networking-capcity;
    uses vim:storage-capacity;
  }
  leaf-list resources {
    description
      "Reference to a collection of existing resource elements required by
       this reservation. It can contain any instance derived from
       ResourceElement, such as ServerInstances or even other
       ResourceReservations. If the ResourceReservation request is
       accepted, the ResourceElement(s) listed here will be placed
       into 'protected' mode as to prevent accidental delete.";
    type instance-identifier {
      ct:instance-type vim:ResourceElement;
    }
    // following 'must' statement applies to each element
    must "boolean(/provider/elements/*[@id=id])" {
      error-message "One or more of the ResourceElement(s) does not exist in
                     the provider to be reserved";
    }
  }

  leaf provider {
    if-feature multi-provider;
    config false;

    description
      "Reference to a specified existing provider from which this reservation
       will be drawn if used in the context of multi-provider
       environment.";
    type instance-identifier {
      ct:instance-type vim:ResourceProvider;
      require-instance true;
    }
  }

  container remaining {
    config false;
    description
      "Provides visibility into total remaining capacity for this
       reservation based on allocations that took effect utilizing
       this reservation ID as a reference.";

    uses vim:compute-capacity;
    uses vim:networking-capcity;
    uses vim:storage-capacity;
  }

  leaf-list allocations {
    config false;
    description
      "Reference to a collection of consumed allocations referencing
       this reservation.";
    type instance-identifier {
      ct:instance-type ResourceAllocation;
    }
  }
}

ct:complex-type ResourceAllocation {
  ct:extends vim:ResourceElement;

  description
     "Contains a list of resources to be allocated with optional reference
     to an existing reservation.

     If reservation is specified but this request is received prior
     to reservation start timestamp, then it will be rejected unless
     'allocate-on-start' is set to true.  'allocate-on-start' allows
     the allocation to be auto-initiated and scheduled to run in the
     future.

     The 'priority' state indicates the classification for dealing
     with resource starvation scenarios. Lower priority allocations
     will be forcefully terminated to allow for higher priority
     allocations to be fulfilled.

     Allocations without reference to an existing reservation will
     receive the lowest priority.";

  reference "OPNFV-PROMISE, Section 3.4.3";

  leaf reservation {
    description "Reference to an existing reservation identifier";

    type instance-identifier {
      ct:instance-type ResourceReservation;
      require-instance true;
    }
  }
  leaf allocate-on-start {
    description
     "If 'allocate-on-start' is set to true, the 'planned' allocations will
     take effect automatically at the reservation 'start' date/time.";
    type boolean; default false;
  }

  ct:instance-list resources {
    description "Contains list of new ResourceElements that will be allocated";
    ct:instance-type vim:ResourceElement;
  }

  leaf priority {
    description
      "Reflects current priority level of the allocation according to classification rules";
    type number;
    config false;
  }
}

// MAIN CONTAINER
container promise {
  ct:instance-list providers {
    description "Aggregate collection of all registered ResourceProvider instances";
    ct:instance-type vim:ResourceProvider;
    config false;

   // augment compute container with capacity elements
   augment "compute" {
     uses resource-capacity {
       augment "capacity/quota" { uses compute-capacity; }
       augment "capacity/usage" { uses compute-capacity; }
       augment "capacity/reserved" { uses compute-capacity; }
       augment "capacity/available" { uses compute-capacity; }
     }
   }

   // augment networking container with capacity elements
   augment "networking" {
     uses resource-capacity {
       if-feature has-networking-capacity;
       augment "capacity/quota" { uses networking-capacity; }
       augment "capacity/usage" { uses networking-capacity; }
       augment "capacity/reserved" { uses networking-capacity; }
       augment "capacity/available" { uses networking-capacity; }
     }
   }

   // track references to reservations for this resource provider
   leaf-list reservations {
     type instance-identifier {
       ct:instance-type ResourceReservation;
     }
   }
  }

  ct:instance-list reservations {
    description "Aggregate collection of all registered ResourceReservation instances";
    ct:instance-type ResourceReservation;
  }

  ct:instance-list allocations {
    description "Aggregate collection of all active ResourceAllocation instances";
    ct:instance-type ResourceAllocation;
  }
}

rpc add-provider {
  description "This operation allows you to register a new ResourceProvider
               into promise management service";
  input {
    leaf provider {
      description "Select a specific resource provider";
      mandatory true;
      type enumeration {
        enum openstack;
        enum hp;
        enum rackspace;
        enum amazon {
          status planned;
        }
        enum joyent {
          status planned;
        }
        enum azure {
          status planned;
        }
      }
    }
    leaf username {
      type string;
      mandatory true;
    }
    leaf password {
      type ianach:crypt-hash;
      mandatory true;
    }
    leaf endpoint {
      type inet:uri;
      description "The target URL endpoint for the resource provider";
      mandatory true;
    }
    leaf region {
      type string;
      description "Optional specified regsion for the provider";
    }
  }
  output {
    leaf id {
      description "Unique identifier for the newly added provider found in /promise/providers";
      type instance-identifier {
        ct:instance-type ResourceProvider;
      }
    }
    leaf result {
      type enumeration {
        enum success;
        enum error;
      }
    }
  }
}
rpc remove-provider;
rpc list-providers;

rpc check-capacity;

rpc list-reservations;
rpc create-reservation;
rpc update-reservation;
rpc cancel-reservation;

rpc list-allocations;
rpc create-allocation;

notification reservation-event;
notification capacity-event;
notification allocation-event;
}
```

CHAPTER

EIGHT

ANNEX B: DOCUMENT REVISION

Version	Description
1.0.1	**JIRA: PROMISE-23** • Reference to YangForge Framework • Corrections to figure 3.1
1.0.2	**JIRA: PROMISE-11** • OPNFV Logo Added • Alignment to ETSI NFV Specs

[PROMISE] OPNFV, “Promise” requirements project, [Online]. Available at https://wiki.opnfv.org/promise

[BLAZAR] OpenStack Blazar Project, [Online]. Available at https://wiki.openstack.org/wiki/Blazar

[PROMOSS] Promise reference implementation, [Online]. Available at https://github.com/opnfv/promise

[YANGFO] Yangforge Project, [Online]. Available at https://github.com/opnfv/yangforge

[NFVMAN] ETSI GS NFV MAN 001, “Network Function Virtualisation (NFV); Management and Orchestration”

[NFV003] ETSI GS NFV 003, “Network Function Virtualisation (NFV); Terminology for Main Concepts in NFV”

[NFVIFA010] ETSI GS NFV IFA 010, “Network Function Virtualisation (NFV); Management and Orchestration; Functional Requirements Specification”

[NFVIFA005] ETSI GS NFV IFA 005, “Network Function Virtualisation (NFV); Management and Orchestration; Or-Vi reference point - Interface and Information Model Specification”

[NFVIFA006] ETSI GS NFV IFA 006 “Network Function Virtualisation (NFV); Management and Orchestration; Vi-Vnfm reference point - Interface and Information Model Specification”

[ETSINFV] ETSI NFV, [Online]. Available at http://www.etsi.org/technologies-clusters/technologies/nfv