IPv6 test specification

Scope

The IPv6 test area will evaluate the ability for a SUT to support IPv6 Tenant Network features and functionality. The tests in this test area will evaluate,

- network, subnet, port, router API CRUD operations
- interface add and remove operations
- security group and security group rule API CRUD operations
- IPv6 address assignment with dual stack, dual net, multiprefix in mode DHCPv6 stateless or SLAAC

References

- OPNFV IPv6 project
  - https://wiki.opnfv.org/display/ipv6/IPv6+Home
- upstream openstack API reference
  - http://developer.openstack.org/api-ref
- upstream openstack IPv6 reference (newton release)
  - https://docs.openstack.org/newton/networking-guide/config-ipv6.html
- upstream openstack IPv6 reference (mitaka release)
  - https://docs.openstack.org/mitaka/networking-guide/config-ipv6.html

Definitions and abbreviations

The following terms and abbreviations are used in conjunction with this test area

- API - Application Programming Interface
- CIDR - Classless Inter-Domain Routing
- CRUD - Create, Read, Update, and Delete
- DHCP - Dynamic Host Configuration Protocol
- DHCPv6 - Dynamic Host Configuration Protocol version 6
- ICMP - Internet Control Message Protocol
- NFVI - Network Functions Virtualization Infrastructure
- NIC - Network Interface Controller
- SDN - Software Defined Network
- SLAAC - Stateless Address Auto Configuration
- TCP - Transmission Control Protocol
- UDP - User Datagram Protocol
- VM - Virtual Machine
- vNIC - virtual Network Interface Card

System Under Test (SUT)

The system under test is assumed to be the NFVI and VIM deployed with a Pharos compliant infrastructure.

Test Area Structure

The test area is structured based on network, port and subnet operations. Each test case is able to run independently, i.e. irrelevant of the state created by a previous test.

Test Descriptions

API Used and Reference

Networks: https://developer.openstack.org/api-ref/networking/v2/index.html#networks

- show network details
- update network
- delete network
- list networks
- create netowrk
- bulk create networks

Subnets: https://developer.openstack.org/api-ref/networking/v2/index.html#subnets

- list subnets
- create subnet
- bulk create subnet
- show subnet details
- update subnet
- delete subnet

Routers and interface: https://developer.openstack.org/api-ref/networking/v2/index.html#routers-routers

- list routers
- create router
- show router details
- update router
- delete router
- add interface to router
- remove interface from router

Ports: https://developer.openstack.org/api-ref/networking/v2/index.html#ports

- show port details
- update port
- delete port
- list port
- create port
- bulk create ports

Security groups: https://developer.openstack.org/api-ref/networking/v2/index.html#security-groups-security-groups

- list security groups
- create security groups
- show security group
- update security group
- delete security group

Security groups rules: https://developer.openstack.org/api-ref/networking/v2/index.html#security-group-rules-security-group-rules

- list security group rules
- create security group rule
- show security group rule
- delete security group rule

Servers: https://developer.openstack.org/api-ref/compute/

- list servers
- create server
- create multiple servers
- list servers detailed
- show server details
- update server
- delete server

Test Case 1 - Create and Delete Bulk Network, IPv6 Subnet and Port

Short name

opnfv.ipv6.tc001.bulk_network_subnet_port_create_delete

Use case specification

This test case evaluates the SUT API ability of creating and deleting multiple networks, IPv6 subnets, ports in one request, the reference is,

tempest.api.network.test_networks.BulkNetworkOpsIpV6Test.test_bulk_create_delete_network
tempest.api.network.test_networks.BulkNetworkOpsIpV6Test.test_bulk_create_delete_subnet
tempest.api.network.test_networks.BulkNetworkOpsIpV6Test.test_bulk_create_delete_port

Test preconditions

None

Basic test flow execution description and pass/fail criteria

- Test action 1: Create 2 networks using bulk create, storing the “id” parameters returned in the response
- Test action 2: List all networks, verifying the two network id’s are found in the list
- **Test assertion 1:** The two “id” parameters are found in the network list
- Test action 3: Delete the 2 created networks using the stored network ids
- Test action 4: List all networks, verifying the network ids are no longer present
- **Test assertion 2:** The two “id” parameters are not present in the network list
- Test action 5: Create 2 networks using bulk create, storing the “id” parameters returned in the response
- Test action 6: Create an IPv6 subnets on each of the two networks using bulk create commands, storing the associated “id” parameters
- Test action 7: List all subnets, verify the IPv6 subnets are found in the list
- **Test assertion 3:** The two IPv6 subnet “id” parameters are found in the network list
- Test action 8: Delete the 2 IPv6 subnets using the stored “id” parameters
- Test action 9: List all subnets, verify the IPv6 subnets are no longer present in the list
- **Test assertion 4:** The two IPv6 subnet “id” parameters, are not present in list
- Test action 10: Delete the 2 networks created in test action 5, using the stored network ids
- Test action 11: List all networks, verifying the network ids are no longer present
- **Test assertion 5:** The two “id” parameters are not present in the network list

- Test action 12: Create 2 networks using bulk create, storing the “id” parameters returned in the response
- Test action 13: Create a port on each of the two networks using bulk create commands, storing the associated “port_id” parameters
- Test action 14: List all ports, verify the port_ids are found in the list
- **Test assertion 6:** The two “port_id” parameters are found in the ports list
- Test action 15: Delete the 2 ports using the stored “port_id” parameters
- Test action 16: List all ports, verify port_ids are no longer present in the list
- **Test assertion 7:** The two “port_id” parameters, are not present in list
- Test action 17: Delete the 2 networks created in test action 12, using the stored network ids
- Test action 18: List all networks, verifying the network ids are no longer present
- **Test assertion 8:** The two “id” parameters are not present in the network list

This test evaluates the ability to use bulk create commands to create networks, IPv6 subnets and ports on the SUT API. Specifically it verifies that:

- Bulk network create commands return valid “id” parameters which are reported in the list commands
- Bulk IPv6 subnet commands return valid “id” parameters which are reported in the list commands
- Bulk port commands return valid “port_id” parameters which are reported in the list commands
- All items created using bulk create commands are able to be removed using the returned identifiers

Post conditions

N/A

Test Case 2 - Create, Update and Delete an IPv6 Network and Subnet

Short name

opnfv.ipv6.tc002.network_subnet_create_update_delete

Use case specification

This test case evaluates the SUT API ability of creating, updating, deleting network and IPv6 subnet with the network, the reference is

tempest.api.network.test_networks.NetworksIpV6Test.test_create_update_delete_network_subnet

Test preconditions

None

Basic test flow execution description and pass/fail criteria

- Test action 1: Create a network, the test passes if the network status is ACTIVE
- Test action 2: Update this network with a new_name, the test passes if the network name equals the new_name
- Test action 3: Create an IPv6 subnet within the network
- Test action 4: Update this IPv6 subnet with a new_name, the test passes if the subnet name equals the new_name
- Test action 5: Delete the subnet and network, then list all subnets and networks, the test passes if the subnet and network are not found in the list

This test case passes if all test action steps execute successfully and all assertions are affirmed. If any test steps fails to execute successfully or any of the assertions is not met, the test case fails.

Post conditions

None

Test Case 3 - Check External Network Visibility

Short name

opnfv.ipv6.tc003.external_network_visibility

Use case specification

This test case verifies user can see external networks but not subnets, the reference is,

tempest.api.network.test_networks.NetworksIpV6Test.test_external_network_visibility

Test preconditions

The SUT has at least one external network.

Basic test flow execution description and pass/fail criteria

- Test action 1: List all networks with external router, the test passes if the list is not empty.
- Test action 2: List all networks without external router in the external network list, the test passes if the list is empty.
- Test action 3: Verify the public network id found in the external networks, the test passes if the public netwrok id is found.
- Test action 4: List the subnets of the external network with the configured public network id, the test passes if the subnet list is empty.

This test case passes if all test action steps execute successfully and all assertions are affirmed. If any test steps fails to execute successfully or any of the assertions is not met, the test case fails.

Post conditions

None

Test Case 4 - List IPv6 Networks and Subnets

Short name

opnfv.ipv6.tc004.network_subnet_list

Use case specification

This test case evaluates the SUT API ability of listing netowrks, subnets after creating a network and an IPv6 subnet, the reference is

tempest.api.network.test_networks.NetworksIpV6Test.test_list_networks
tempest.api.network.test_networks.NetworksIpV6Test.test_list_subnets

Test preconditions

None

Basic test flow execution description and pass/fail criteria

- Test action 1: Create a network.
- Test action 2: List the networks, the test passes if the created network exists in the list.
- Test action 3: Create an IPv6 subnet of this network.
- Test action 4: List the subnets of this network, the test passes if the created IPv6 subnet exists in the list.
- Test action 5: Delete the created subnet and network, then list all the subnets and networks, the test passes if the subnet and network are not found in the list.

This test case passes if all test action steps execute successfully and all assertions are affirmed. If any test steps fails to execute successfully or any of the assertions is not met, the test case fails.

Post conditions

None

Test Case 5 - Show Details of an IPv6 Network and Subnet

Short name

opnfv.ipv6.tc005.network_subnet_show

Use case specification

This test case evaluates the SUT API ability of showing the network, subnet details, the reference is,

tempest.api.network.test_networks.NetworksIpV6Test.test_show_network
tempest.api.network.test_networks.NetworksIpV6Test.test_show_subnet

Test preconditions

None

Basic test flow execution description and pass/fail criteria

- Test action 1: Create a network.
- Test action 2: List all networks, the test passes if the created network id and name exist in the list.
- Test action 3: Create an IPv6 subnet of the network.
- Test action 4: Show the details of the created subnet, the test passes if the id and CIDR in the details are equal to the sunet's id and CIDR, respectively.
- Test action 5: Delete the subnet and network, then list all the subnets and networks, the test passes if the subnet and network are not found in the list.

This test case passes if all test action steps execute successfully and all assertions are affirmed. If any test steps fails to execute successfully or any of the assertions is not met, the test case fails.

Post conditions

None

Test Case 6 - Create an IPv6 Port in Allowed Allocation Pools

Short name

opnfv.ipv6.tc006.port_create_in_allocation_pool

Use case specification

This test case evaluates the SUT API ability of creating an IPv6 subnet within allowed IPv6 address allocation pool and creating a port whose address is in the range of the pool, the reference is,

tempest.api.network.test_ports.PortsIpV6TestJSON.test_create_port_in_allowed_allocation_pools

Test preconditions

There should be an IPv6 CIDR configuration, which prefixlen is less than 126.

Basic test flow execution description and pass/fail criteria

- Test action 1: Create a network.
- Test action 2: Check the allocation pools configuration, the test passes if the prefixlen of the IPv6 CIDR configuration is less than 126.
- Test action 3: Get the allocation pool by setting the start_ip and end_ip based on the IPv6 CIDR configuration.
- Test action 4: Create an IPv6 subnet of the network within the allocation pools.
- Test action 5: Create a port of the network, the test passes if the port's id is in the range of the allocation pools.
- Test action 6: Delete the created port, subnet and network, then list all ports, subnets and networks, the test passes if the port, subnet and network are not found in the list.

This test case passes if all test action steps execute successfully and all assertions are affirmed. If any test steps fails to execute successfully or any of the assertions is not met, the test case fails.

Post conditions

None

Test Case 7 - Create an IPv6 Port with Empty Security Groups

Short name

opnfv.ipv6.tc007.port_create_empty_security_group

Use case specification

This test case evaluates the SUT API ability of creating port with empty security group, the reference is,

tempest.api.network.test_ports.PortsIpV6TestJSON.test_create_port_with_no_securitygroups

Test preconditions

None

Basic test flow execution description and pass/fail criteria

- Test action 1: Create a network.
- Test action 2: Create an IPv6 subnet of the network.
- Test action 3: Create a port of the network with an empty security group, the test passes if the security group of this port is not none but is empty.
- Test action 4: Delete the created port, subnet and network, then list all ports, subnets and networks, the test passes if the port, subnet and network are not found in the list.

This test case passes if all test action steps execute successfully and all assertions are affirmed. If any test steps fails to execute successfully or any of the assertions is not met, the test case fails.

Post conditions

None

Test Case 8 - Create, Update and Delete an IPv6 Port

Short name

opnfv.ipv6.tc008.port_create_update_delete

Use case specification

This test case evaluates the SUT API ability of creating, updating, deleting IPv6 port, the reference is,

tempest.api.network.test_ports.PortsIpV6TestJSON.test_create_update_delete_port

Test preconditions

None

Basic test flow execution description and pass/fail criteria

- Test action 1: Create a network
- Test action 2: Create a port of the network, the test passes if the port’s ‘admin_state_up’ is True.
- Test action 3: Update the port’s name with a new_name and set port’s admin_state_up to False, the test passes if the port’s name is equal to new_name and the port’s admin_state_up is False.
- Test action 4: Delete the port and network, then list all ports and networks, the test passes if the port and network are not found in the list.

This test case passes if all test action steps execute successfully and all assertions are affirmed. If any test steps fails to execute successfully or any of the assertions is not met, the test case fails.

Post conditions

None

Test Case 9 - List IPv6 Ports

Short name

opnfv.ipv6.tc009.port_list

Use case specification

This test case evaluates the SUT ability of listing the created port, the reference is,

tempest.api.network.test_ports.PortsIpV6TestJSON.test_list_ports

Test preconditions

None

Basic test flow execution description and pass/fail criteria

- Test action 1: Create a network.
- Test action 2: Create a port of the network.
- Test action 3: List all ports, the test passes if the created port exists in the list of all ports.
- Test action 4: Delete the port and network, then list all ports and networks, the test passes if the port and network are not found in the list.

This test case passes if all test action steps execute successfully and all assertions are affirmed. If any test steps fails to execute successfully or any of the assertions is not met, the test case fails.

Post conditions

None

Test Case 10 - Show Key/Valus Details of an IPv6 Port

Short name

opnfv.ipv6.tc010.port_show_details

Use case specification

This test case evaluates the SUT ability of showing the port details, the reference is,

tempest.api.network.test_ports.PortsIpV6TestJSON.test_show_port

Test preconditions

None

Basic test flow execution description and pass/fail criteria

- Test action 1: Create a network.
- Test action 2: Create a port of the network.
- Test action 3: Show the details of this created port, the test passes if the created port’s id exists in the details.
- Test action 4: Check the details of the created port, the test passes if the values in the details are the same as the values to create the port.

- Test action 5: Delete the port and network, then list all ports and networks, the test passes if the deleted port and network are not found in the list.

This test case passes if all test action steps execute successfully and all assertions are affirmed. If any test steps fails to execute successfully or any of the assertions is not met, the test case fails.

Post conditions

None

Test Case 11 - Add Multiple Interfaces for an IPv6 Router

Short name

opnfv.ipv6.tc011.router_add_multiple_interface

Use case specification

This test case evaluates the SUT ability of adding multiple interface to a router, the reference is,

tempest.api.network.test_routers.RoutersIpV6Test.test_add_multiple_router_interfaces

Test preconditions

None

Basic test flow execution description and pass/fail criteria

- Test action 1: Create 2 networks named network01 and network02
- Test action 2: Create an IPv6 subnet01 in network01, an IPv6 subnet02 in network02
- Test action 3: Create a router
- Test action 4: Create interface01 with subnet01 and the router, the test passes if the router_id equals to the interface01's 'device_id' and subnet01_id equals to the interface01's 'subnet_id'
- Test action 5: Create interface02 with subnet02 and the router, the test passes if the router_id equals to the interface02's 'device_id' and subnet02_id equals to the interface02's 'subnet_id'
- Test action 6: Delete the networks, subnets, router and interfaces, then list all interfaces, ports, subnets, networks, the test passes if the deleted ones are not found in the list.

This test case passes if all test action steps execute successfully and all assertions are affirmed. If any test steps fails to execute successfully or any of the assertions is not met, the test case fails.

Post conditions

None

Test Case 12 - Add and Remove an IPv6 Router Interface with port_id

Short name

opnfv.ipv6.tc012.router_interface_add_remove_with_port

Use case specification

This test case evaluates the SUT abiltiy of adding, removing router interface to a port, the subnet_id and port_id of the interface will be checked, the port's device_id will be checked if equals to the router_id or not. The reference is,

tempest.api.network.test_routers.RoutersIpV6Test.test_add_remove_router_interface_with_port_id

Test preconditions

None

Basic test flow execution description and pass/fail criteria

- Test action 1: Create a network
- Test action 2: Create an IPv6 subnet of the network
- Test action 3: Create a router
- Test action 4: Create a port of the network
- Test action 5: Add router interface to the port created
- Test action 6: Check the interface's keys, the test passes if the keys include 'subnet_id' and 'port_id'
- Test action 7: Check the 'device_id' in port details, the test passes if it equals to the router_id
- Test action 8: Delete the network, subnet, router and port, then list all ports, routers, subnets and networks, the test passes if the deleted ones are not found in the list.

This test case passes if all test action steps execute successfully and all assertions are affirmed. If any test steps fails to execute successfully or any of the assertions is not met, the test case fails.

Post conditions

None

Test Case 13 - Add and Remove an IPv6 Router Interface with subnet_id

Short name

opnfv.ipv6.tc013.router_interface_add_remove

Use case specification

This test case evaluates the SUT API abilities of adding and removing a router interface with the IPv6 subnet id, the reference is

tempest.api.network.test_routers.RoutersIpV6Test.test_add_remove_router_interface_with_subnet_id

Test preconditions

None

Basic test flow execution description and pass/fail criteria

- Test action 1: Create a network and a router
- Test action 2: Create an IPv6 subnet with the network created
- Test action 3: Add a router interface with ids of the router and IPv6 subnet created

- Test action 4: Verify the key 'subnet_id' is included in the added interface's keys
- Test action 5: Verify the key 'port_id' is included in the added interface's keys
- Test action 6: Show the port info with the interface's port id
- Test action 7: Verify the router id is equal to the device id shown in the port info
- Test action 8: Delete the network, subnet and router, then list networks, subnets and routers, the test passes if they are not found in the list after deleting

This test case passes if all test action steps execute successfully and all assertions are affirmed. If any test steps fails to execute successfully or any of the assertions is not met, the test case fails.

Post conditions

None

Test Case 14 - Create, Show, List, Update and Delete an IPv6 router

Short name

opnfv.ipv6.tc014.router_create_show_list_update_delete

Use case specification

This test case evaluates the SUT API abilities of creating, showing, listing, updating and deleting routers, the reference is

tempest.api.network.test_routers.RoutersIpV6Test.test_create_show_list_update_delete_router

Test preconditions

There should exist an OpenStack external network.

Basic test flow execution description and pass/fail criteria

- Test action 1: Create a router, set the admin_state_up to be False and external_network_id to be public network id
- Test action 2: Verify the created router's admin_state_up is False
- Test action 3: Verify the created router's external network id equals to the public network id
- Test action 4: Show details of the created router
- Test action 5: Verify the router's name shown is the same as the router created
- Test action 6: List all routers and verify if created router is in response message
- Test action 7: Update the name of router and verify if it is updated
- Test action 8: Delete the router, then list routers, the test passes if the router is not found in the list after deleting

This test case passes if all test action steps execute successfully and all assertions are affirmed. If any test steps fails to execute successfully or any of the assertions is not met, the test case fails.

Post conditions

None

Test Case 15 - Create, List, Update, Show and Delete an IPv6 security group

Short name

opnfv.ipv6.tc015.security_group_create_list_update_show_delete

Use case specification

This test case evaluates the SUT API abilities of creating, listing, updating, showing and deleting security groups, the reference is

tempest.api.network.test_security_groups.SecGroupIPv6Test.test_create_list_update_show_delete_security_group

Test preconditions

None

Basic test flow execution description and pass/fail criteria

- Test action 1: Create a security group
- Test action 2: List security groups and verify if created security group is there in response
- Test action 3: Update the name and description of this security group
- Test action 4: Verify if the security group's name and description are updated
- Test action 5: Show details of the updated security group, the test passes if the name and description in shown details are equal to the name and description used to update.
- Test action 6: Delete the security group, then list security groups, the test passes if it is not found in the list after deleting

This test case passes if all test action steps execute successfully and all assertions are affirmed. If any test steps fails to execute successfully or any of the assertions is not met, the test case fails.

Post conditions

None

Test Case 16 - Create, Show and Delete IPv6 security group rule

Short name

opnfv.ipv6.tc016.security_group_rule_create_show_delete

Use case specification

This test case evaluates the SUT API abilities of creating, showing, listing and deleting security group rules, the reference is

tempest.api.network.test_security_groups.SecGroupIPv6Test.test_create_show_delete_security_group_rule

Test preconditions

None

Basic test flow execution description and pass/fail criteria

- Test action 1: Create a security group
- Test action 2: Create a rule of the security group with protocol tcp, udp and icmp, respectively
- Test action 3: Show details of the created security rule, the test passes if the created security rule values equal to the rule values used to create.
- Test action 4: List rules, the test passes if the created rule is in the listed rules.
- Test action 5: Delete the security group and security group rules, then list security groups and security group rules, the test passes if they are not found in the list after deleting.

This test case passes if all test action steps execute successfully and all assertions are affirmed. If any test steps fails to execute successfully or any of the assertions is not met, the test case fails.

Post conditions

None

Test Case 17 - List IPv6 Security Groups

Short name

opnfv.ipv6.tc017.security_group_list

Use case specification

This test case evaluates the SUT API abilities of listing security groups, the reference is

tempest.api.network.test_security_groups.SecGroupIPv6Test.test_list_security_groups

Test preconditions

There should exist a default security group.

Basic test flow execution description and pass/fail criteria

- Test action 1: List all security groups
- Test action 2: Verify the default security group exists in the list, the test passes if the default security group exists

This test case passes if all test action steps execute successfully and all assertions are affirmed. If any test steps fails to execute successfully or any of the assertions is not met, the test case fails.

Post conditions

None

Test Case 18 - IPv6 Address Assignment - Dual Stack, DHCPv6 Stateless

Short name

opnfv.ipv6.tc018.dhcpv6_stateless

Use case specification

This test case evaluates IPv6 address assignment in mode ‘dhcpv6_stateless’, and verify the ping6 available VM can ping all of the v4 addresses and other’s v6 addresses as well as the router’s in the same network, the reference is

tempest.scenario.test_network_v6.TestGettingAddress.test_dhcp6_stateless_from_os

Test preconditions

There should exist a public router or a public network.

Basic test flow execution description and pass/fail criteria

- Test action 1: Create one network and one IPv4 subnet of this network
- Test action 2: If there exists a public router, use it as the router. Otherwise, use the public network to create a router
- Test action 3: Connect the IPv4 subnet to the router
- Test action 4: Create one IPv6 subnet of the network created in action 1 in mode ‘dhcpv6_stateless’
- Test action 5: Connect the IPv6 subnet to the router
- Test action 6: Boot two VMs on this network
- Test action 7: Verify the vNICs of all VMs gets all addresses actually assigned
- Test action 8: Verify each VM can ping the other’s v4 private address
- Test action 9: Verify the ping6 available VM can ping all of the other’s v6 addresses as well as the router’s
- Test action 10: Delete the network and VMs, then list networks and VMs, the test passes if the VMs and the network are not found in the list after deleting.

This test case passes if all test action steps execute successfully and all assertions are affirmed. If any test steps fails to execute successfully or any of the assertions is not met, the test case fails.

Post conditions

None

Test Case 19 - IPv6 Address Assignment - Dual Net, Dual Stack, DHCPv6 Stateless

Short name

opnfv.ipv6.tc019.dualnet_dhcpv6_stateless

Use case specification

This test case evaluates IPv6 address assignment in mode ‘dhcpv6_stateless’, and verify the ping6 available VM can ping all of the v4 addresses in another network and other’s v6 addresses as well as the router’s in the same network, the reference is

tempest.scenario.test_network_v6.TestGettingAddress.test_dualnet_dhcp6_stateless_from_os

Test preconditions

There should exists a public router or a public network.

Basic test flow execution description and pass/fail criteria

- Test action 1: Create one network and one IPv4 subnet of this network

- Test action 2: If there exists a public router, use it as the router. Otherwise, use the public network to create a router
- Test action 3: Connect the IPv4 subnet to the router
- Test action 4: Create another network and one IPv6 subnet of this network in mode 'dhcpv6_stateless'
- Test action 5: Connect the IPv6 subnet to the router
- Test action 6: Boot two VMs on these two networks
- Test action 7: Turn on 2nd NIC of each VM
- Test action 8: Verify the vNICs of all VMs gets all addresses actually assigned
- Test action 9: Verify each VM can ping the other's v4 private address
- Test action 10: Verify the ping6 available VM can ping all of the other's v6 addresses as well as the router's
- Test action 11: Delete the network and VMs, then list networks and VMs, the test passes if the VMs and the network are not found in the list after deleting.

This test case passes if all test action steps execute successfully and all assertions are affirmed. If any test steps fails to execute successfully or any of the assertions is not met, the test case fails.

Post conditions

None

Test Case 20 - IPv6 Address Assignment - Multiple Prefixes, Dual Stack, DHCPv6 Stateless

Short name

opnfv.ipv6.tc020.multiple_prefixes_dhcpv6_stateless

Use case specification

This test case evaluates IPv6 address assignment in mode 'dhcpv6_stateless', and verify the ping6 available VM can ping all of the v4 addresses and other's v6 addresses as well as the router's in the same network, the reference is

tempest.scenario.test_network_v6.TestGettingAddress.test_multi_prefix_dhcpv6_stateless

Test preconditions

There should exist a public router or a public network.

Basic test flow execution description and pass/fail criteria

- Test action 1: Create one network and one IPv4 subnet of this network
- Test action 2: If there exists a public router, use it as the router. Otherwise, use the public network to create a router
- Test action 3: Connect the IPv4 subnet to the router
- Test action 4: Create two IPv6 subnets of the network created in action 1 in mode 'dhcpv6_stateless'
- Test action 5: Connect the two IPv6 subnets to the router
- Test action 6: Boot two VMs on this network
- Test action 7: Verify the vNICs of all VMs gets all v6 addresses actually assigned
- Test action 8: Verify each VM can ping the other's v4 private address
- Test action 9: Verify the ping6 available VM can ping all of the other's v6 addresses as well as the router's
- Test action 10: Delete the network and VMs, then list networks and VMs, the test passes if the VMs and the network are not found in the list after deleting.

This test case passes if all test action steps execute successfully and all assertions are affirmed. If any test steps fails to execute successfully or any of the assertions is not met, the test case fails.

Post conditions

None

Test Case 21 - IPv6 Address Assignment - Dual Net, Multiple Prefixes, Dual Stack, DHCPv6 Stateless

Short name

opnfv.ipv6.tc021.dualnet_multiple_prefixes_dhcpv6_stateless

Use case specification

This test case evaluates IPv6 address assignment in mode 'dhcpv6_stateless', and verify the ping6 available VM can ping all of the v4 addresses in another network and other's v6 addresses as well as the router's in the same network, the reference is

tempest.scenario.test_network_v6.TestGettingAddress.test_dualnet_multi_prefix_dhcpv6_stateless

Test preconditions

There should exist a public router or a public network.

Basic test flow execution description and pass/fail criteria

- Test action 1: Create one network and one IPv4 subnet of this network
- Test action 2: If there exists a public router, use it as the router. Otherwise, use the public network to create a router
- Test action 3: Connect the IPv4 subnet to the router
- Test action 4: Create another network and two IPv6 subnets of this network in mode 'dhcpv6_stateless'
- Test action 5: Connect the two IPv6 subnets to the router
- Test action 6: Boot two VMs on these two networks
- Test action 7: Turn on 2nd NIC of each VM
- Test action 8: Verify the vNICs of all VMs gets all addresses actually assigned
- Test action 9: Verify each VM can ping the other's v4 private address
- Test action 10: Verify the ping6 available VM can ping all of the other's v6 addresses as well as the router's
- Test action 11: Delete the network and VMs, then list networks and VMs, the test passes if the VMs and the network are not found in the list after deleting.

This test case passes if all test action steps execute successfully and all assertions are affirmed. If any test steps fails to execute successfully or any of the assertions is not met, the test case fails.

Post conditions

None

Test Case 22 - IPv6 Address Assignment - Dual Stack, SLAAC

Short name

opnfv.ipv6.tc022.slaac

Use case specification

This test case evaluates IPv6 address assignment in mode 'slaac', and verify the ping6 available VM can ping all of the v4 addresses and other's v6 addresses as well as the router's in the same network, the reference is

tempest.scenario.test_network_v6.TestGettingAddress.test_slaac_from_os

Test preconditions

There should exist a public router or a public network.

Basic test flow execution description and pass/fail criteria

- Test action 1: Create one network and one IPv4 subnet of this network
- Test action 2: If there exists a public router, use it as the router. Otherwise, use the public network to create a router
- Test action 3: Connect the IPv4 subnet to the router
- Test action 4: Create one IPv6 subnet of the network created in action 1 in mode 'slaac'
- Test action 5: Connect the IPv6 subnet to the router
- Test action 6: Boot two VMs on this network
- Test action 7: Verify the vNICs of all VMs gets all addresses actually assigned
- Test action 8: Verify each VM can ping the other's v4 private address
- Test action 9: Verify the ping6 available VM can ping all of the other's v6 addresses as well as the router's
- Test action 10: Delete the network and VMs, then list networks and VMs, the test passes if the VMs and the network are not found in the list after deleting.

This test case passes if all test action steps execute successfully and all assertions are affirmed. If any test steps fails to execute successfully or any of the assertions is not met, the test case fails.

Post conditions

None

Test Case 23 - IPv6 Address Assignment - Dual Net, Dual Stack, SLAAC

Short name

opnfv.ipv6.tc023.dualnet_slaac

Use case specification

This test case evaluates IPv6 address assignment in mode 'slaac', and verify the ping6 available VM can ping all of the v4 addresses in another network and other's v6 addresses as well as the router's in the same network, the reference is

tempest.scenario.test_network_v6.TestGettingAddress.test_dualnet_slaac_from_os

Test preconditions

There should exist a public router or a public network.

Basic test flow execution description and pass/fail criteria

- Test action 1: Create one network and one IPv4 subnet of this network
- Test action 2: If there exists a public router, use it as the router. Otherwise, use the public network to create a router
- Test action 3: Connect the IPv4 subnet to the router
- Test action 4: Create another network and one IPv6 subnet of this network in mode 'slaac'
- Test action 5: Connect the IPv6 subnet to the router
- Test action 6: Boot two VMs on these two networks
- Test action 7: Turn on 2nd NIC of each VM
- Test action 8: Verify the vNICs of all VMs gets all addresses actually assigned
- Test action 9: Verify each VM can ping the other's v4 private address
- Test action 10: Verify the ping6 available VM can ping all of the other's v6 addresses as well as the router's
- Test action 11: Delete the network and VMs, then list networks and VMs, the test passes if the VMs and the network are not found in the list after deleting.

This test case passes if all test action steps execute successfully and all assertions are affirmed. If any test steps fails to execute successfully or any of the assertions is not met, the test case fails.

Post conditions

None

Test Case 24 - IPv6 Address Assignment - Multiple Prefixes, Dual Stack, SLAAC

Short name

opnfv.ipv6.tc024.multiple_prefixes_slaac

Use case specification

This test case evaluates IPv6 address assignment in mode 'slaac', and verify the ping6 available VM can ping all of the v4 addresses and other's v6 addresses as well as the router's in the same network, the reference is

tempest.scenario.test_network_v6.TestGettingAddress.test_multi_prefix_slaac

Test preconditions

There should exists a public router or a public network.

Basic test flow execution description and pass/fail criteria

- Test action 1: Create one network and one IPv4 subnet of this network
- Test action 2: If there exists a public router, use it as the router. Otherwise, use the public network to create a router
- Test action 3: Connect the IPv4 subnet to the router
- Test action 4: Create two IPv6 subnets of the network created in action 1 in mode 'slaac'
- Test action 5: Connect the two IPv6 subnets to the router
- Test action 6: Boot two VMs on this network
- Test action 7: Verify the vNICs of all VMs gets all v6 addresses actually assigned

- Test action 8: Verify each VM can ping the other’s v4 private address
- Test action 9: Verify the ping6 available VM can ping all of the other’s v6 addresses as well as the router’s
- Test action 10: Delete the network and VMs, then list networks and VMs, the test passes if the VMs and the network are not found in the list after deleting.

This test case passes if all test action steps execute successfully and all assertions are affirmed. If any test steps fails to execute successfully or any of the assertions is not met, the test case fails.

Post conditions

None

Test Case 25 - IPv6 Address Assignment - Dual Net, Dual Stack, Multiple Prefixes, SLAAC

Short name

opnfv.ipv6.tc025.dualnet_multiple_prefixes_slaac

Use case specification

This test case evaluates IPv6 address assignment in mode ‘slaac’, and verify the ping6 available VM can ping all of the v4 addresses in another network and other’s v6 addresses as well as the router’s in the same network, the reference is

tempest.scenario.test_network_v6.TestGettingAddress.test_dualnet_multi_prefix_slaac

Test preconditions

There should exist a public router or a public network.

Basic test flow execution description and pass/fail criteria

- Test action 1: Create one network and one IPv4 subnet of this network
- Test action 2: If there exists a public router, use it as the router. Otherwise, use the public network to create a router
- Test action 3: Connect the IPv4 subnet to the router
- Test action 4: Create another network and two IPv6 subnets of this network in mode ‘slaac’
- Test action 5: Connect the two IPv6 subnets to the router
- Test action 6: Boot two VMs on these two networks
- Test action 7: Turn on 2nd NIC of each VM
- Test action 8: Verify the vNICs of all VMs gets all addresses actually assigned
- Test action 9: Verify each VM can ping the other’s v4 private address
- Test action 10: Verify the ping6 available VM can ping all of the other’s v6 addresses as well as the router’s
- Test action 11: Delete the network and VMs, then list networks and VMs, the test passes if the VMs and the network are not found in the list after deleting.

This test case passes if all test action steps execute successfully and all assertions are affirmed. If any test steps fails to execute successfully or any of the assertions is not met, the test case fails.

Post conditions

None