# RIFT.ware™ version 7.2.0.108497

Release Notes

November 2019

# Notice

# Table of Contents

# RIFT.ware 7.2.0 Release Notes

This guide describes the RIFT.ware 7.2.0 release, including new features, fixed and known issues, with their workarounds.

# New and Changed Features for 7.2.0

RIFT.ware version 7.2.0 introduces enhancements to improve management.

| Feature | PFR and JIRAs |
|---|---|
| Authorization Token support for Monitoring Parameters | PFR-439 |
| Configuration Manager Changes | N/A |
| Datacenter Section Added to the UI | PFR-525 |
| Dynamic Editing of Running Network Service | PFR-562 |
| Improved AWS Support | PFR-47 |
| Mapping Existing Service Elements to Existing Resources<br>Network Service Instantiation Updates | PFR-621 |
| Or-Vnfm Account (ESTI GS NVF-SOL 003) Enhancements during Instantiation | PFR-500 |
| Os-Ma-Nfvo Interface (ESTI GS NVF-SOL 005) Enhancements during Instantiation | PFR-611 |
| RIFT RO and VIM Re-synchronization | PFR-521 |
| Support for Kubernetes as VIM | PFR-553 |
| Support for NS and VNF Heal | PFR-598 |

# Authorization Token support for Monitoring Parameters

Prior to this feature, RIFT.ware did not support the ability to include an 'authorization token' in the REST API calls made to an VNF for collecting monitoring parameters. In some cases, authorization tokens are required by VNFs to limit the execution and rate of certain REST APIs. RIFT.ware currently only supports basic authentication and authorization by using a username and password by fetching monitoring parameter data.

This feature enables users to add token support when requesting monitoring parameters if the VNF requires the tokens.



## Obtaining and Sending a Token for Monitoring Parameters

There are three options to obtain and send a token.

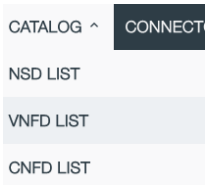
### Direct Auth Token

Users can directly add a pre-created token in the VNFD HTTP ENDPOINT section (see the Standardized Token (Oauth2 Server)) to query for monitoring parameters.

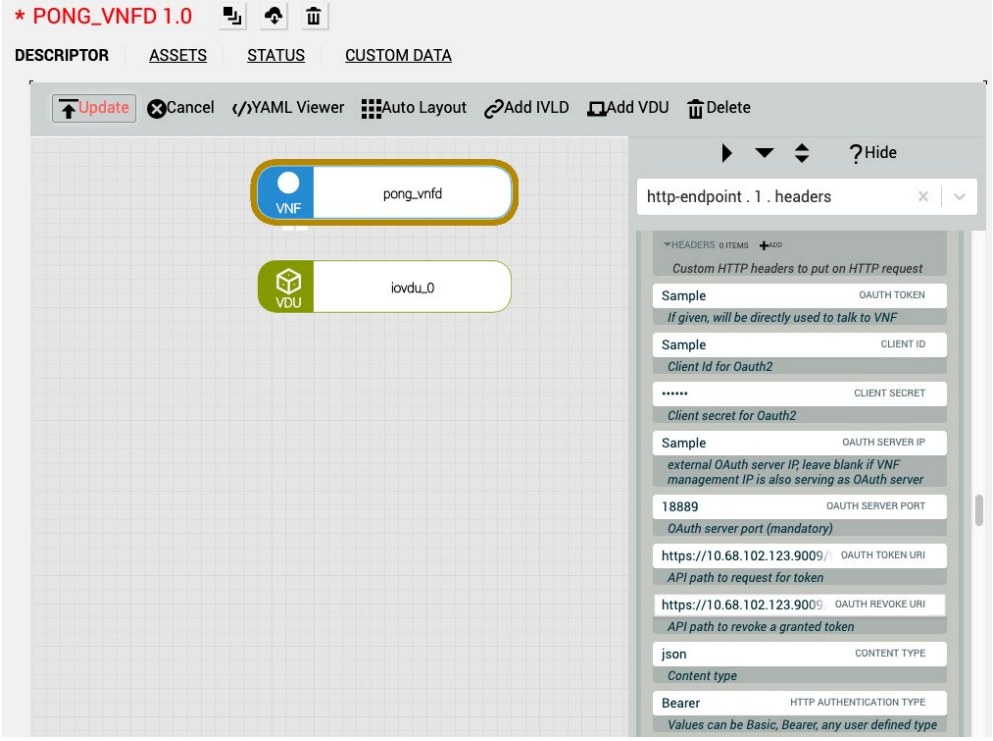
### Standardized Token (Oauth2 Server)

A token can be obtained from any OAuth2 enabled server. REST calls to the VNF to obtain monitoring parameter data will query the OAuth server using the token URL. Any additional queries will use the token for fetching monitoring parameters.

1. On the **Launchpad** menu, click **CATALOG** > **VNFD LIST** to open the list of catalog items.




2. Click next to the ping_vnfd or pong_vnfd that you want to see in more detail. The **DESCRIPTOR** tab opens.
3. Search for **http-endpoint . 1 . oauth-token** field to add the appropriate OAuth images.




4. Under **HEADERS**, provide the following information (required fields*):

- **OAUTH TOKEN**: If the token is added, then it will communicate directly to the VNF.
- **CLIENT ID**: Unique identifier for distinguishing RIFT as an Oauth2 client to the server. The Launchpad API server should be configured with this ID to authenticate RIFT.ware as a client for obtaining a token.
- **CLIENT SECRET**: Secret password for RIFT Oauth2 client to the server.
- **OAUTH SERVER IP**: External OAuth server IP.

**Note:** Do not fill in this OAuth Server IP field if the VNF management IP is also serving as the OAuth server.

- **OAUTH SERVER PORT***: OAuth server port.
- **OAUTH TOKEN URI**: API path to request a token.
- **OAUTH REVOKE URI**: API path to revoke a granted token.
- **CONTENT TYPE**: Content type. JSON is the default type.
- **HTTP AUTHENTICATION TYPE**: Values can be basic. This can be any user defined type. Bearer is the default type.

**Note:** A user can add a custom header instead of Bearer in the **HTTP AUTHENTICATION TYPE** section specific to the VNF.

5. Click Update.

> **Note:** After adding the token and instantiating the NS, the user must verify that the tokens were added by querying the monitoring parameters for the VNFDs. If a token is invalid, then monitoring parameters data will not be available.

If a user does not fill in the **OAUTH TOKEN** or **OAUTH TOKEN URI** fields, then RIFT.ware checks the scripts for the token. See the Customized Token (Config Primitive Script).

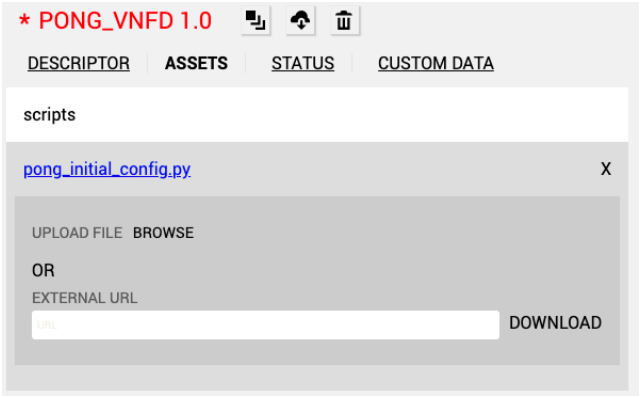
**Customized Token (Config Primitive Script)**

There is also an option to obtain a customized token using a config primitive script. A user must write a script that will retrieve the token and send the output from the token in the format specified in the script to the console. The user needs to upload the script under the Asset tab of the VNFD. Any entry must be added in the VNFD under vnf-configuration => config-primitive.

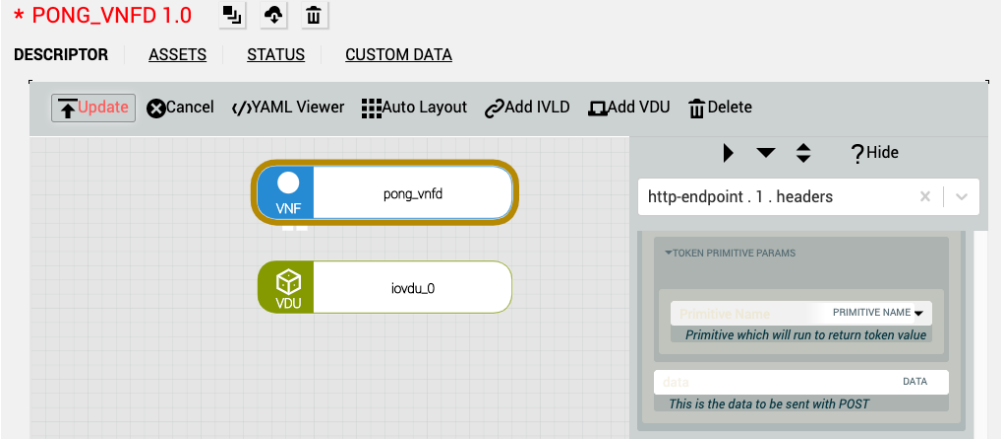
<token-start>X6tS3gVO5bv2CbmLGitzdp37bglRBzn7UAoilZ2eIJI=<token-end>

The primitive script is automatically triggered while instantiating and the corresponding token is returned.

1. Click the **Asset** tab to upload the script.




2. Either click **BROWSE** to upload a local file or click **DOWNLOAD** to download an external file.
3. Return to the **Descriptor** tab.
4. Search for **http-endpoint . 1 . oauth-token** field again and scroll down to the **TOKEN PRIMITIVE PARAMS** section.




5. Under **token-primitive-params** provide the following information:
   - **PRIMITIVE NAME**: The primitive that will run and return the token value. You must upload a script in the Asset tab in order to add the primitive name to this section.

> **Note:** After adding the token and instantiating the NS, the user must verify that the tokens were added by querying the monitoring parameters for the VNFDs. If a token is invalid, then monitoring parameters data will not be available.

See the Monitor the Network Service section.

## Model Enhancement

The following changes have been made to the VNFD model to allow a user to obtain a token.
VNFD => http-endpoint

**OAuth-Token Fields**

| Name | Type | Cardinality | Description |
|---|---|---|---|
| username | string | 1 | Username for OAuth2 |
| password | string | 1 | Password for OAuth2 |
| oauth-token | string | 1 | If present, will be directly used to query monitoring params from VNF |
| client-id | string | 1 | Client id for OAuth2 |
| client-secret | string | 1 | Client secret for OAuth2 |
| oauth-server-ip | string | 1 | External OAuth server IP, leave blank if VNF management IP is also serving as OAuth server |
| oauth-server-port | uint32 | 1 | OAuth server port (mandatory) |
| oauth-token-uri | string | 1 | API path to request for token |

| | | | |
|---|---|---|---|
| oauth-revoke-uri | string | 1 | API path to revoke a granted token |
| http-authentication-type | string | 1 | Values can be Basic, Bearer, any user defined type |

**Script Input Related Changes**

| Leaf name | Type | Cardinality | Description |
|---|---|---|---|
| primitive-name | string | 1 | Primitive which will run to return the token value. |

## Configuration Manager Enhancements

This feature provides a rw-config-util python3 module that can be used to create configuration scripts for NSD and VNFD. This new module provides a standard interface for all configuration scripts. The script no longer has to parse different input data based on the Life Cycle Management event that triggered the script. Also, the configuration utility automatically creates a log file at the standard location in Launchpad. The configuration scripts are now run as a rwcfgmgr user instead of root or ubuntu users.

This feature adds support for the following rwconfigutil APIs:

- Module rwconfigutil
  - Module rwconfigutil.agent
  - Module rwconfigutil.base
  - Module rwconfigutil.cp
  - Module rwconfigutil.lcm
  - Module rwconfigutil.main
  - Module rwconfigutil.parameter
  - Module rwconfigutil.vdu
  - Module rwconfigutil.vnf

**Module rwconfigutil**

rwconfigutil python3 module helps with configuration scripts used during Life Cycle Management for network services. The main class is ConfigUtil and it can be imported as:

```
from rwconfigutil import ConfigUtil
```

The class uses an input_dict or input_file as input and provides an instance. Then, an operator can use class methods to access the various data in the input.

```
    inp = ConfigUtil(input_dict=yaml_cfg, log=logger)

    # Get the required and optional parameters

    ping_vnfr = inp.affected_vnfs[0]

    mgmt_ip = ping_vnfr.management_ip

    mgmt_port = ping_vnfr.management_port

    pong_vnfr = inp.get_vnf_by_index(2)

    pong_svc_ip = pong_vnfr.get_cp_by_name('pong_vnfd/cp1').port_ips[0]
```

The rwconfigutil.main can be run as a python script to parse an input file and print out all the parsed data.

```
python3 -m rwconfigutil.main -f <input_file>
```

Sub-modules

- Module rwconfigutil.agent
- Module rwconfigutil.base
- Module rwconfigutil.cp
- Module rwconfigutil.lcm
- Module rwconfigutil.main
- Module rwconfigutil.parameter
- Module rwconfigutil.vdu
- Module rwconfigutil.vnf

**Module rwconfigutil.agent**

This API handles the config agent section of input.

Classes

**Class AgentType**

class AgentType (*args, **kwargs)

An enumeration.

**Ancestors (in MRO)**

- enum.Enum

Class variables

*Variable JUJU*

*Variable SCRIPT*

**Class ConfigAgent**

class ConfigAgent(log, input_dict)

Class to manage Configuration Agent related data.

**Ancestors (in MRO)**

- rwconfigutil.base.BaseSection

Instance variables

*Variable type*

str: Type of the configuration agent.

**Module rwconfigutil.base**

This base class handles config input sections.

Classes

**Class BaseSection**

class BaseSection(log, input_dict)

**Descendants**

- rwconfigutil.agent.ConfigAgent
- rwconfigutil.lcm.Lcm
- rwconfigutil.parameter.Parameter
- rwconfigutil.cp.Cp
- rwconfigutil.vdu.Vdu
- rwconfigutil.vnf.Vnf

**Instance variables**

*Variable log*

**Module rwconfigutil.cp**

This handles the Connection Points section of input.

Classes

**Class Cp**

class Cp(log, input_dict, cp_type='external')

Class to manage connection points related data.

**Ancestors (in MRO)**

- rwconfigutil.base.BaseSection

**Instance variables**

Variable *floating_ips*

list of str: Provide the list of floating IPs for this CP.

*Variable name*

str: Name of this connection point.

*Variable port_ips*

list of str: Provide the list of port IPs for this CP.

**Methods**

*Method get_ips*

```
def get_ips(self)
```

list of set of str: Returns the list of set of port IPs and corresponding floating IPs.

**Module rwconfigutil.lcm**

This handles the LCM section of input.

Classes

**Class Lcm**

class Lcm(log, input_dict)

Class to manage the LCM section of the data.

**Ancestors (in MRO)**

- rwconfigutil.base.BaseSection

**Instance variables**

*Variable config*

dict: Config related to this event.

*Variable event*

LcmEvent: The type of LCM event.

*Variable retry*

bool: If this is retry of LCM event.

*Variable trigger*

str: Trigger for this LCM event like pre, post, etc.

*Variable vnf_level*

bool: If the LCM event is a VNF level event.

**Methods**

*Method get_value*

```
def get_value(self, key)
```

str: Return value of any key that is part of the LCM input.

**Class LcmEvent**

class LcmEvent(*args, **kwargs)

An enumeration.

**Ancestors (in MRO)**

- enum.Enum

Class variables

*Variable HEAL*
*Variable INSTANTIATE*
*Variable MANUAL*
*Variable SCALE*
*Variable TERMINATE*
*Variable UNKNOWN*
*Variable UPDATE*
*Variable VNF_HEAL*
*Variable VNF_INSTANTIATE*
*Variable VNF_MANUAL*
*Variable VNF_TERMINATE*

**Module rwconfigutil.main**

This is the main module for the RIFT configuration utility. The class is ConfigUtil. This module can be run from the command line by providing an input file.

Functions

**Function main**

```
def main(args=None)
```

Helper function to run this utility from the command line.

**Classes**

**Class ConfigUtil**

class ConfigUtil(input_dict=None, input_file=None, log=None, debug=False)

Configuration utility to parse the input for configuration script during NS LCM event.

Constructor for the ConfigUtil

**Args**

**input_dict : dict, optional**
Input for the config as a script.

**input_file : dict, optional**
Input file for the config.

**log : logging.logger, optional**
Pass the logger to use. If None, default logger used.

**debug : str,optional**
Enable debug logs in the stderr. Default is off.

Either input_dict or input_file should be provided.

**Instance variables**

*Variable affected_vnfs*
list of Vnf: Returns the list of affected VNFs.
The affected VNFs are the VNFS which got added or removed as part of the LCM event In case of VNF initial config, this provides the VNF that is being configured, for scale, it provides the list of VNFs being added or deleted.

*Variable config_agent*
ConfigAgent: Returns the config agent section of the input.

*Variable lcm*
Lcm: Returns LCM section of the input.

*Variable log*
logging.Logger - Get the logger being used.

*Variable log_file*
str - Get the log filename, if available.

*Variable ns_name*
str: Returns the name for the NS.

*Variable other_vnfs*
list of Vnf: Returns the list of the other VNFs which are not affected by the LCM.

*Variable parameters*

list of Parameter: Returns the parameters section of the input.

*Variable vnfs*

list of Vnf: Returns list of all VNFs.

**Methods**

*Method get_asset_path*

```
def get_asset_path(self, filepath)
```

Get the path to the asset.

Args

**filepath : str**

File path relative to the script.

Returns

**str**

Absolute path to the asset.

Raises

**ValueError**

If the absolute path is not under the script directory.

*Method get_vnf_by_name*

```
def get_vnf_by_name(self, name, exact_match=True,
affected_only=False)
```

Get the first VNF matching the name.

Args

**name : str**

Name to match.

**exact_match : bool, optional**

Whether to do an exact match or check if name is in the VNF name. Default is true, do exact match.

**affected_only : bool, optional**

Search in affected VNFs list or all VNFs. Default false, search in all VNFs.

Returns

Vnf: VNF matching the name.

*Method get_vnfs_by_name*

```
def get_vnfs_by_name(self, name, affected_only=False)
```

Get the list of VNFs containing the name.

Args

**name : str**

Name to match

**affected_only : bool, optional**
Search in affected VNFs list or all VNFs. Default is False, i.e., search in all VNFs.

Returns

list of Vnf: VNF matching the name

**Module rwconfigutil.parameter**

This handles the parameters section of input.

Classes

Class DataType

class DataType(*args, **kwargs)

An enumeration.

**Ancestors (in MRO)**

- enum.Enum

**Class variables**

*Variable BOOLEAN*
*Variable INTEGER*
*Variable STRING*

Class Parameter

```
class Parameter(log, input_dict)
```

Class to manage parameters related data.

**Ancestors (in MRO)**

- rwconfigutil.base.BaseSection

Methods

*Method get_parameter*

```
def get_parameter(self, name, default=None)
```

Get the value for the parameter with the matching name.

Args

**name : str**
Name of the parameter.

default (str, optional): Value to return if parameter not found. Default is None.

Returns

**str**
Value of the parameter.

**Module rwconfigutil.vdu**

This handles the VDUR section of input.

Classes

**Class Vdu**

class Vdu(log, input_dict)

Class to manage VDU related data.

Ancestors (in MRO)

- rwconfigutil.base.BaseSection

Instance variables

Variable *id*
str: ID of this VDU.
Variable *management_floating_ip*
str: Management floating IP for this VDU.
Variable *management_ip*
str: Management port IP for this VDU.
Variable *name*
str: Name of this VDU.
Variable *vdu_id*
str: ID of this VDU in the descriptor.

Methods

*Method get_external_cp_by_name*
```
def get_external_cp_by_name(self, name)
```
Cp: Return the connection point matching the name.

*Method get_external_cps*
```
def get_external_cps(self)
```
list of Cp: List of external connection points on the VDU.

**Module rwconfigutil.vnf**

This handles the VNFR section of input.

Classes

**Class Vnf**

```
class Vnf(log, input_dict, member_index=None)
```

Class to manage VNF related data.

**Ancestors (in MRO)**

- rwconfigutil.base.BaseSection

**Instance variables**

*Variable dashboard_url*

str: Dashbosrd URL for the VNF.

*Variable datacenter*

str: Name of the datacenter where this VNF is deployed.

*Variable management_ip*

str: First management IP address for this VNF.

*Variable management_ip_list*

list of str: List of management IP addresses for this VNF.

*Variable management_port*

int: Management port for the VNF.

*Variable member_index*

int: Member index of this VNF in the NSD.

*Variable name*

str: Name of this VNF.

**Methods**

Method *get_cp_by_name*

```
def get_cp_by_name(self, name)
```

Cp: Return the connection point matching the name.

*Method get_cps*

```
def get_cps(self)
```

list of Cp: List of connection points exposed on the VNF.

*Method get_vdu_by_descriptor_id*

```
def get_vdu_by_descriptor_id(self, vdu_id)
```

list of Vdu: List of VDUs which match the VDU descriptor ID.

*Method get_vdu_by_name*

```
def get_vdu_by_name(self, name, exact_match=True)
```

Vdu: First VDU which matches the name.

**Args**

**name : str**

Name to match.

exact_match (bool, optional) - Whether to do an exact match or check if name is in the VDU name. Default is true, do exact match.

**Returns**

list of Vdu: First VDU matching the name.

*Method get_vdus*

def get_vdus(self)

list of Vdu: List of VDUs that are part of the VNF.

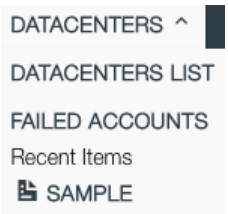
# Datacenter Section Added to the UI

This feature introduces a new approach to adding VIM accounts driven by the discovery of VIM resources in the UI. There is a new Datacenter dashboard menu option. A user can now view an account with a similar display to the Service and Package sections. A user can also add a new account and audit an account in this section.



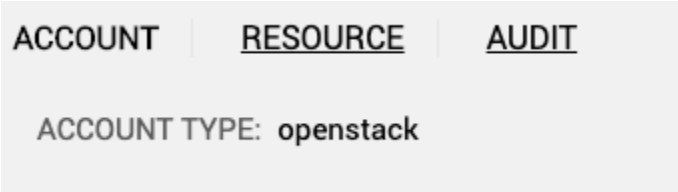
## DATACENTERS Dashboard Menu Option

The new **DATACENTER** menu option includes a dropdown list with **DATACENTER LIST** and **FAILED ACCOUNTS** sections.




**DATACENTER**: This menu options lists all the datacenters associate with your Launchpad. The datacenter lists the name, type, account status and discovery status for each service.

Click to open a full-page view for each specific datacenter. You can also delete, refresh and discover resources for a specific datacenter by clicking . When you open a full-page view for a specific datacenter, there are three sub tabs: ACCOUNT, RESOURCE and AUDIT.




ACCOUNT: The account section lists the details that you add when you create a datacenter. See details below on how to add a datacenter.

RESOURCE: The resource section includes a list of all possible resource. Choose a resource from the drop-down menu to view relevant info.

| Resource Name | Attributes displayed in the UI | Supported RO | | |
|---|---|---|---|---|
| | | OpenStack | VMware VCD | VMware VIO |
| service-status | AODH | Yes | Not Applicable | Not Applicable |
| security-group | ID, NAME | Yes | No | Yes |
| network | ID, NAME, SUBNET (ID, NAME) | Yes | Yes* | Yes |
| public-ip | ID, NAME | Yes | No | No |
| image | ID, NAME | Yes | Yes | Yes |
| flavor | ID, NAME, VCPU, RAM, DISK | Yes | No | Yes |
| volume | ID, NAME | Yes | No | Yes |
| server | ID, NAME, INTERFACE, NETWORK | Yes | Yes | Yes |
| interface | ID, NAME, NETWORK-ID | Yes | No | Yes |
| role | ROLE-NAME | Yes | Yes | Yes |
| vnfr | ID, NAME, STATUS | Yes | Yes | Yes |
| vim-quotas | QUOTA, VALUE | Yes | No | Yes |
| availability-zone | ZONE-NAME, HOST | Yes (Admin Only) | Yes (Admin Only) | Yes (Admin Only) |
| host-aggregate | AGGREGATE-NAME, ZONE-NAME, HOST, CUSTOM-DATA | Yes (Admin Only) | Yes (Admin Only) | Yes (Admin Only) |

***Note:** VMware VCD specific APIs cannot get details for subnet ids or names from VMware at this time.

AUDIT: See the RIFT RO and VIM Re-synchronization section for more details on how to run an audit report.


**DATACENTER LIST**: This section includes all the successful datacenters associated with your Launchpad.

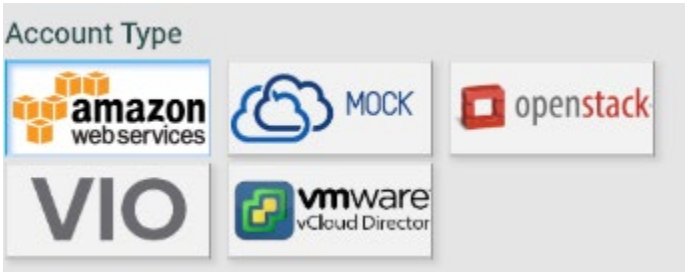
**FAILED ACCOUTS**: This section includes all the failed datacenters associated with your Launchpad.

## Add a Datacenter in the UI

You can add one of the below datacenters using this new menu option.

- Amazon Web Services Datacenter
- OpenStack Datacenter
- VMware VIO Datacenter
- VMware vCloud Director Datacenter

**Note:** Mock is an internal datacenter account for RIFT.inc. Please let me know if we want a different description for this account type.




1. On the **Launchpad** menu, click **DATACENTERS**.
2. Click the + button to add a new datacenter. An account wizard appears. Fill in the appropriate details for the specific datacenter.

**Amazon Web Services Datacenter**

a. Choose amazon web services in the **Account Type** section.
b. Add a unique account name in the ***Name** field.
c. **Vdu Instance Timeout information** (SECONDS): Maximum time allocated for resource instantiation in the VIM account. Default is 300.
d. Click **Next >>** and provide the following Amazon Web Services VIM account details:
   - **Key**: Access Key ID for the AWS account
   - **Secret**: Secret key for the Access Key ID for the AWS account
   - **Region**: Amazon EC2 region to use
   - **Availability Zone**: Availability zone where EC2 instance should be started
   - **SSH Key**: SSH public key to connect to EC2 instance
   - **VPC ID**: Virtual Private Cloud (VPC) ID to use to instantiate EC2 instances

- **Default Subnet ID**: Default subnet ID to create network interface for EC2 instances
- **Dynamic Flavor Support**: Support for creation of flavors in the VIM (True or False)

e. Carefully check your entries and click **SAVE**.

**OpenStack Datacenter**

a. Choose  in the **Account Type** section.

b. Add a unique account name in the ***Name** field.

c. **Vdu Instance Timeout information** (SECONDS): Maximum time allocated for resource instantiation in the VIM account. Default is 300.

d. Click **Next >>** and provide the following OpenStack VIM account details:

- ***Key**: Access Username for the account
- ***Secret**: Password for the account
- ***Auth URL**: Keystone URL to use for getting authentication tokens
- **Cert Validate**: Certificate validation policy in case of SSL/TLS connection (True or False)
- **User Domain**: Domain for the OpenStack user
- **Project Domain**: Domain of the OpenStack project
- ***Tenant**: Project to use for the account
- **Region**: Region of the VIM to use for the account
- **Admin**: Whether the user has admin privileges (True or False)
- **Mgmt Network**: Name of the provider network to use for creating management interfaces
- **Floating IP Pool**: Name of pool to use for allocating floating IP address
- **Security Groups**: Enter a comma separated list of values. Names of the security groups for the VM.
- **Dynamic Flavor Support**: Support for creation of flavors in the VIM (True or False)
- **City Code Distro Location**: City code used for naming the resources allocated, if enabled
- **State Code Distro Location**: State code used for naming the resources allocated, if enabled
- **Code Location**: Location code used for naming the resources allocated, if enabled

e. Carefully check your entries and click **SAVE**.

**VMware VIO Datacenter**

a. Choose  in the **Account Type** section.

b. Add a unique account name in the ***Name** field.

c. **Vdu Instance Timeout information** (SECONDS): Maximum time allocated for resource instantiation in the VIM account. Default is 300.

d. Click **Next >>** and provide the following VMware Integrated OpenStack (VIO) VIM account details:

- ***Key**: Username on the VMware VIO.
- ***Secret**: Password associated with your VMware VIO username/key for the application user
- ***Auth URL**: VIO URL
- **Cert Validate**: Certificate validation policy in case of SSL/TLS connection (True or False)
- **User Domain**: Domain for the OpenStack user
- **Project Domain**: Domain of the OpenStack project
- ***Tenant**: Project to use for the account
- **Region**: Region of the VIM to use for the account
- **Admin**: Whether the user has admin privileges (True or False)
- **Mgmt Network**: Name of the provider network to use for creating management interfaces
- **Floating IP Pool**: Name of pool to use for allocating floating IP address
- **Security Groups**: Enter a comma separated list of values. Names of the security groups for the VM.
- **Dynamic Flavor Support**: Support for creation of flavors in the VIM (True or False)

**VMware vCloud Director Datacenter**

a. Choose  in the **Account Type** section.

b. Add a unique account name in the ***Name** field.

c. **Vdu Instance Timeout information** (SECONDS): Maximum time allocated for resource instantiation in the VIM account. Default is 300.

d. Click **Next >>** and provide the following VMware vCloud Director (vCD) VIM account details:

- ***User** – Organization admin for vCD organization VDC
- ***Password** – Password for the account

- ***Auth URL** – VCD URL
- **Cert Validate**: Certificate validation policy in case of SSL/TLS connection (True or False)
- ***Tenant** – Organization VDC on VCD
- ***Organization** – Name of VCD organization
- ***Mgmt Network**– Name of the network where the management interfaces are located.
- **Dynamic Flavor Support**: Support for creation of flavors in the VIM (True or False)

e. Click **Next >>** and provide the following VMware vCD system admin credentials.
   - **User**: VMware vCD system administrator username
   - **Password**: VMware vCD system administrator password

f. Click **Next >>** and provide the following VMware vCD NSX details.
   - **Auth URL**: VMware NSX Authentication URL
   - **User**: VMware NSX username
   - **Password**: VMware NSX password

g. Click **Next >>** and provide the following vcenter-credentials details.
   - **Host**: VMware vCenter IP address or hostname
   - **Port**: VMware vCenter port
   - **User**: VMware vCenter admin username
   - **Password**: VMware vCenter admin password

h. Carefully check your entries and click **Save**.

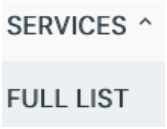
# Dynamic Editing of Running Network Service

This feature allows operators to modify a running service by adding or removing a VNF to a running service. While adding a VNF, it is possible to edit the VNF with specific customizations.

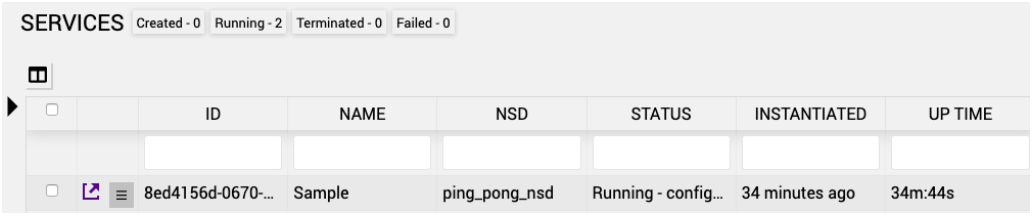


## UI Enhancements

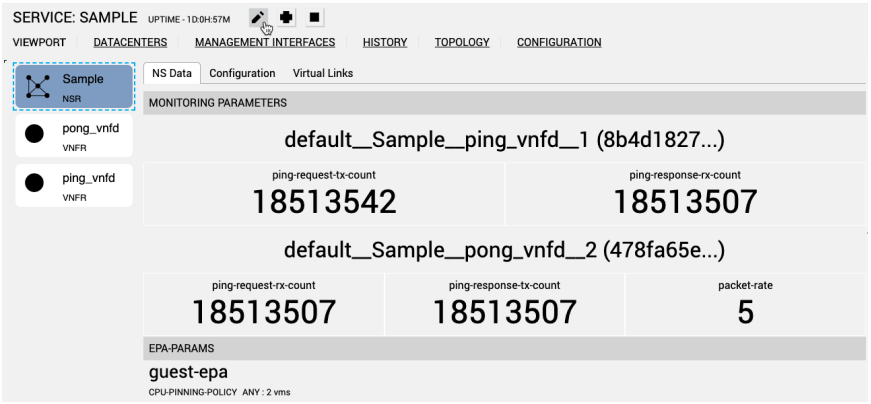
1. On the **Launchpad** menu, click **SERVICES** > **FULL LIST** to open the list of NS instances.




2. Click next to the NS that you want to see in more detail. The **VIEWPORT** screen opens.




3. Click to open the Service Update widget and provide the select-update details:

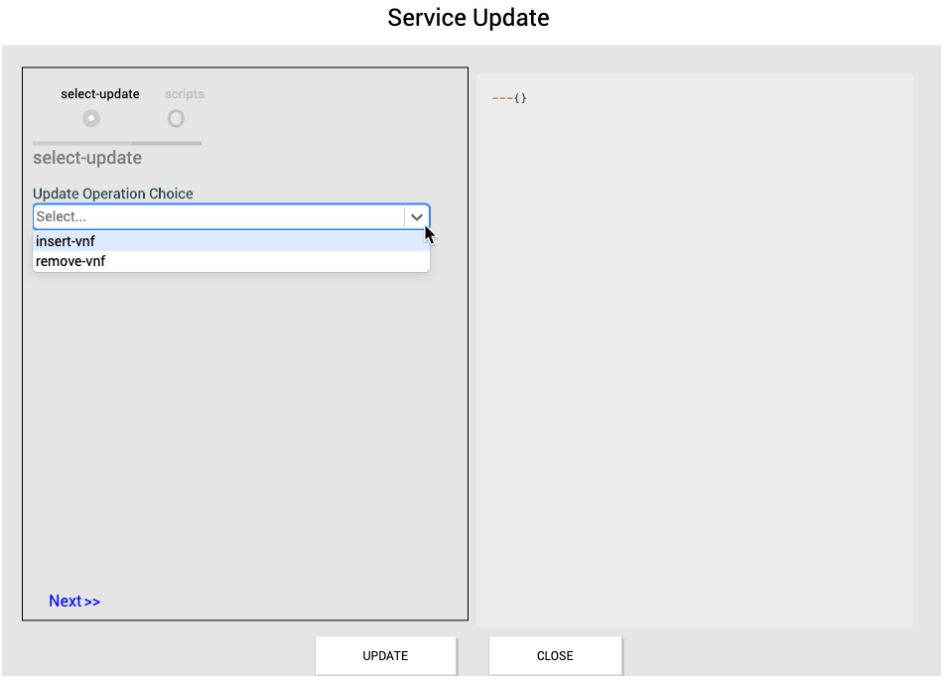



4. In the **Update Operation Choice** drop-down, either choose insert-vnf or remove-vnf.

### Insert a VNF into a Running Service

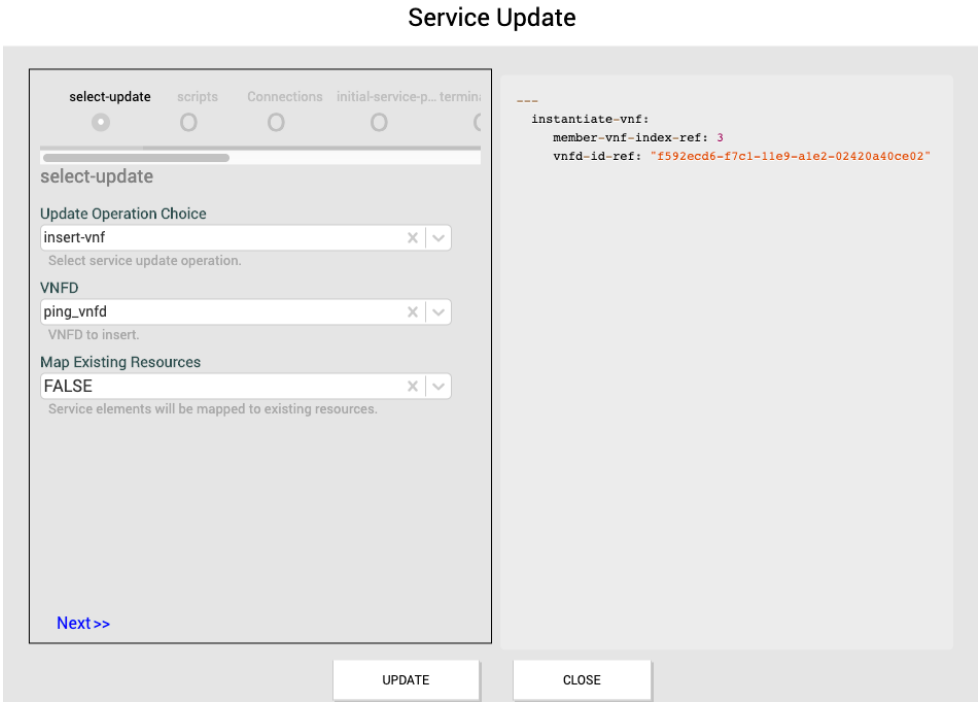
The steps and screenshots in this section are based on the ping pong sample application. This specific workflow is determined by the selected VNFD and NSD associated with the running service.

1. After choosing **insert-vnf**, additional fields appear. Provide the select-update details to insert a VNF into a Running Service:

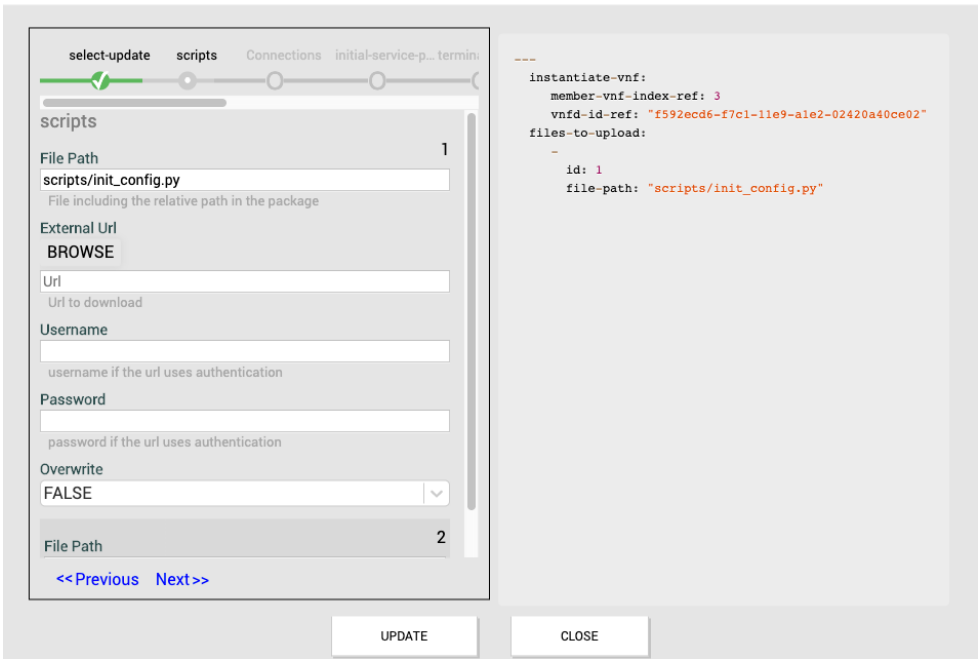



- **VNFD**: VNFD to insert. The options include the VNFDs from the project's catalog.
- **Map Existing Resources:** Service elements will be mapped to existing resources. (True or False). If you want to map existing resources, then see the Brownfield Instantiation and Discovery Support section for instructions on this process.

> **Note:** The YAML configuration appears on the right side of the screen for each section.




2. Click **Next >>** to provide the following details in the Scripts step to identify which scripts that you want to add to the service:



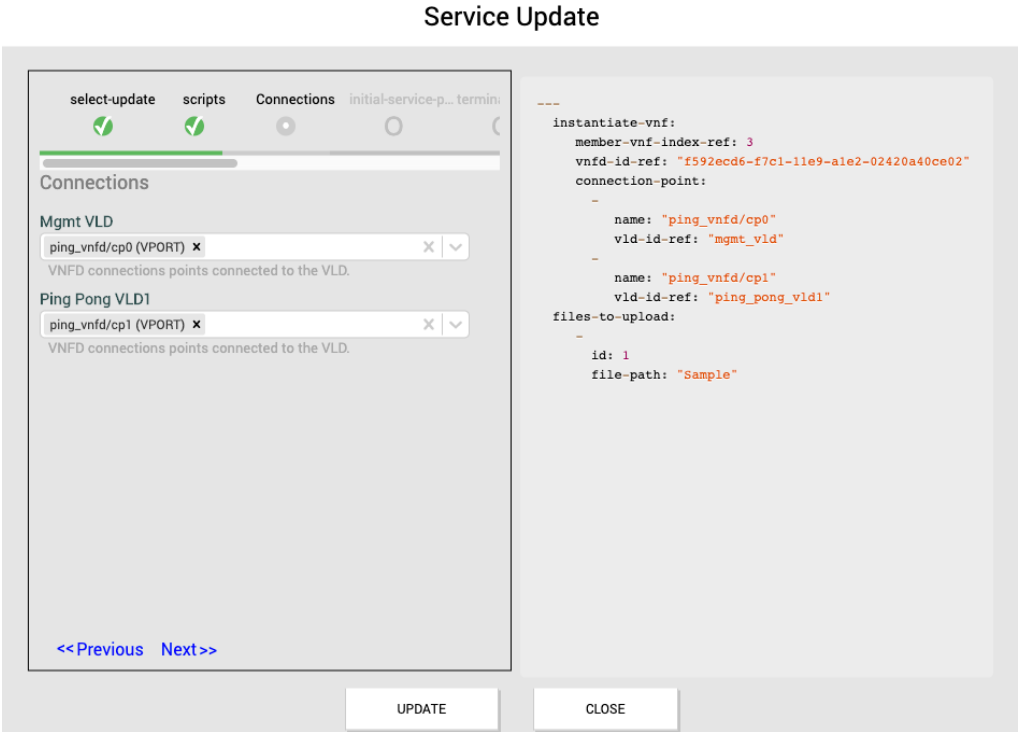

- **File Path 1**: File including the relative path in the package. After you type the file name, additional fields appear.

> **Note:** You must specify the script folder in this field.

- **External URL**: **BROWSE** or type the URL: URL to download. You can either browse or type your URL and add the script to the subfolder that you defined in the File path field.
- **Username**: Username if the url uses authentication.
- **Password**: Password if the url uses authentication.
- **Overwrite**: Overwrite the username and password in this field by choosing False (True or False)

> **Note:** If you have additional scripts, then type the path in File Path 2 and the same fields will appear for the subsequent scripts.

3. Click **Next >>** to provide details in the Connections step:

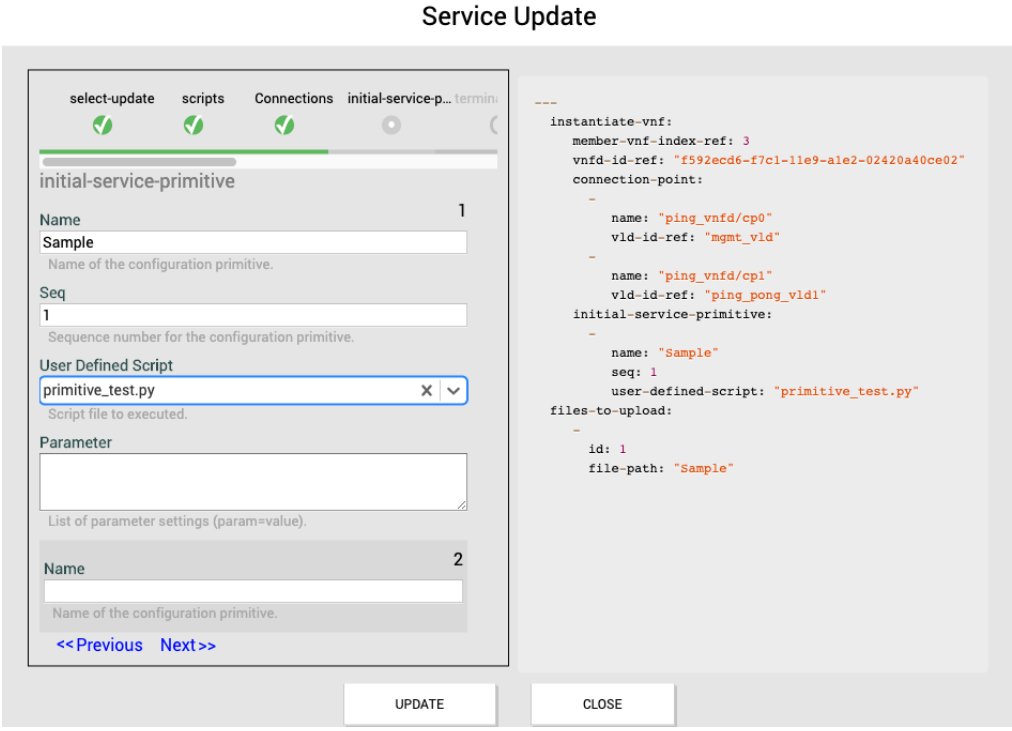



The fields listed in the Connections section are the VLDs in the NSD. If you add connections, then the connections points are linked to a VL.

> **Note:** When you choose a connection point in the first drop-down, it will not appear in the second drop-down section.

4. Click **Next >>** to provide the following details in the initial-service-primitive step:

> **Note:** This step is optional.

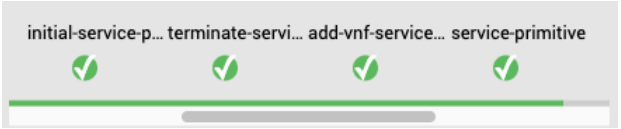



- **Name 1**: Name of the configuration primitive. After you type the configuration name, additional fields appear.
- **Seq**: Sequence number for the configuration primitive.
- **User Defined Script**: Select the script file to be executed from the drop-down menu. The options include:
  - script files that are already included in the NSD.
  - any script files added using this update process (file path scripts)
- **Parameter**: List of parameter settings (param=value).

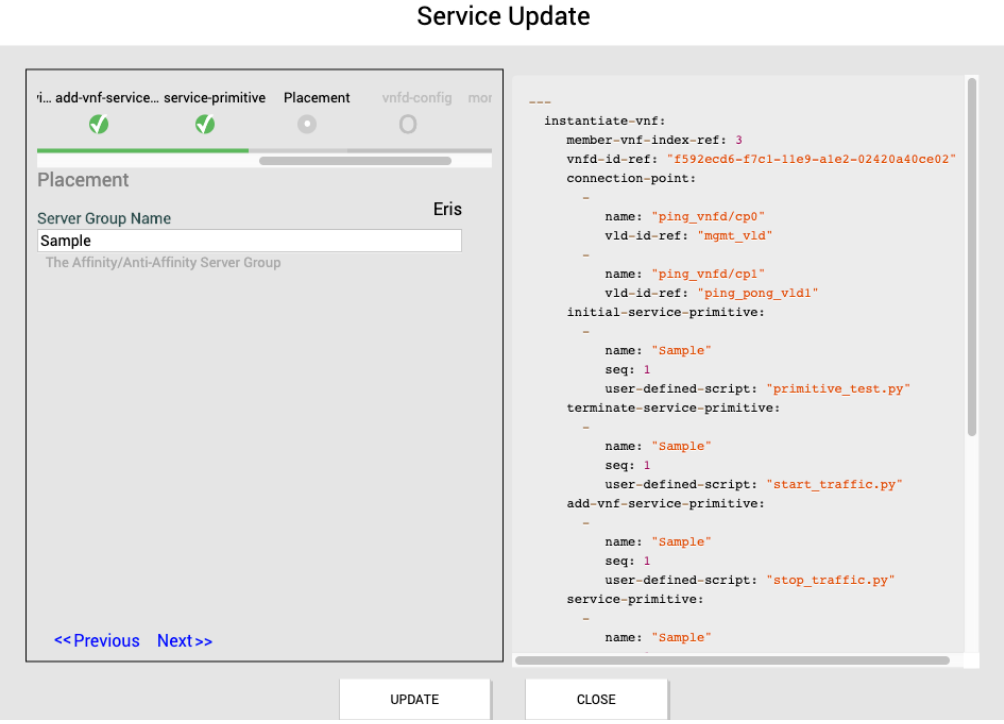
> **Note:** If you have additional initial-service-primitive, then type the name in Name 2 and the same fields will appear for the subsequent initial-service-primitives.

5. Click **Next >>** to provide details in the terminate-service-primitive, add-vnf-service-primitive and service-primitive steps. These steps include all same details as the initial-service-primitive step.

> **Note:** These steps are optional.



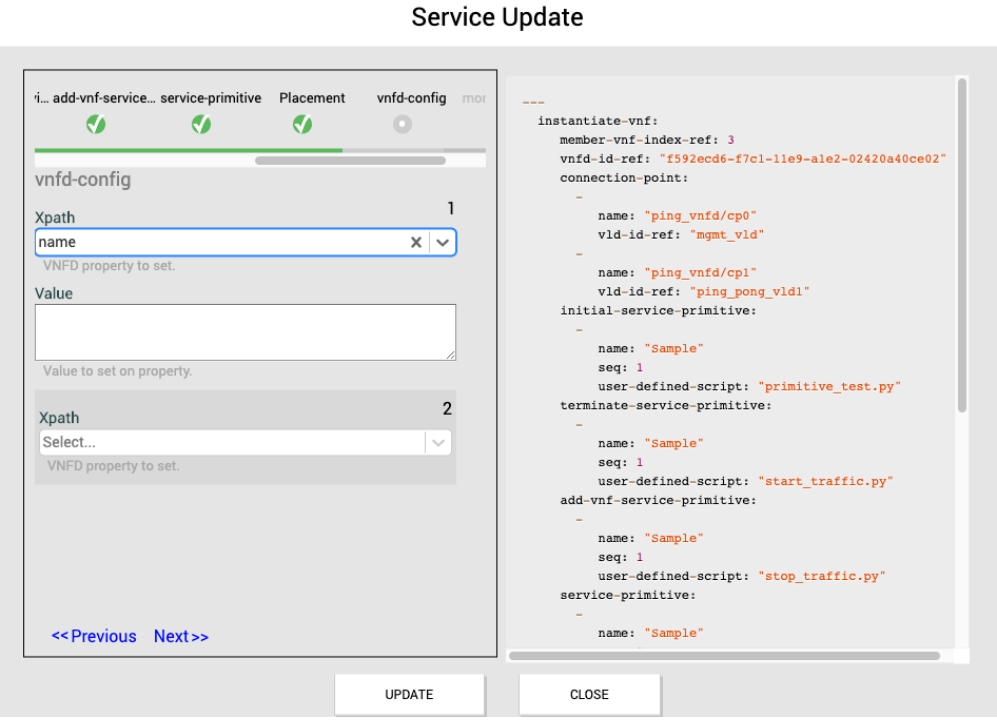

6. Click **Next >>** to provide the following details in the Placement step:




- **Server Group Name**: The Affinity/Anti-Affinity Server Group
- **Datacenter**: A datacenter select box appears if there is more than one available datacenter to pick in this option. The options in this section are dependent on the configuration of the selected datacenter. You may be able to specify a region, an availability zone and host aggregate meta-data.

> **Note:** This step only appears if the VNFD had defined a placement group. The options in this step can also change slightly depending on your selected datacenter.

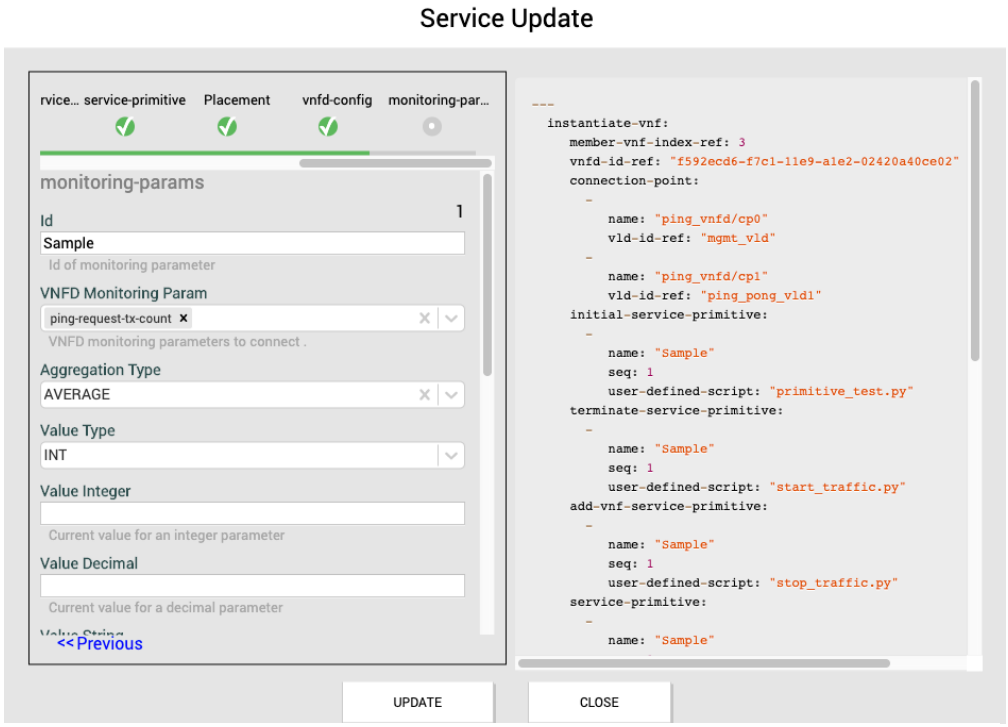
7. Click **Next >>** to provide the following details in the vnfd-config step:

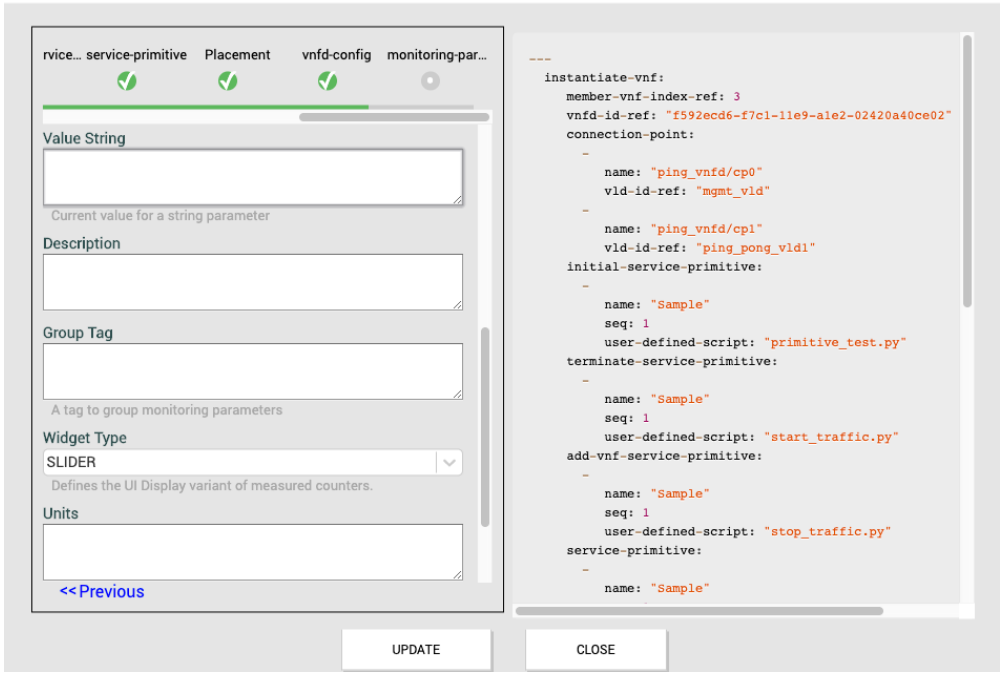



- **Xpath 1**: Select the VNFD property to set from the drop-down menu. After you choose the xpath, additional fields appear. The xpath options are dependent on the definition of the VNF that you are adding to your NS. These options are the same as the options available if you configure an input-parameter-xpath in the NSD designer.
- **Value**: Value to set on property.

> **Note:** If you have additional xpaths, then choose the xpath in the Xpath 2 drop-down menu and the same fields will appear for the subsequent xpaths.

8. Click **Next >>** to provide the following details in the monitoring-params step:



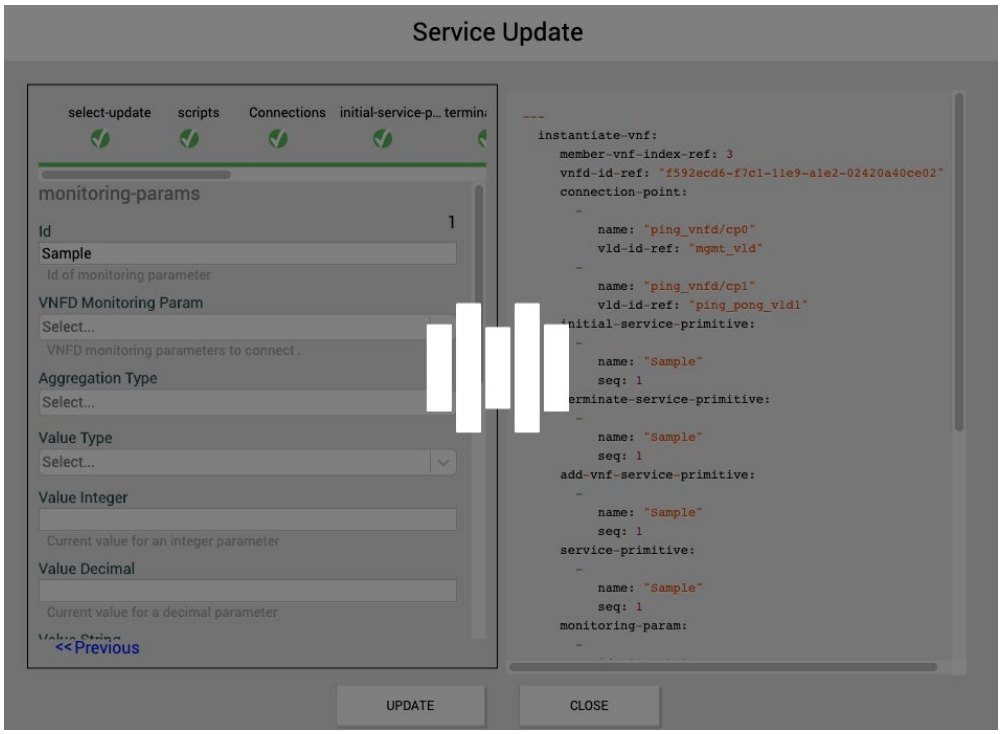




- **Id 1:** Id of monitoring parameter. After you type the ID, additional fields appear.
- **VNFD Monitoring Param**: Select the VNFD monitoring parameters to connect from the drop-down menu.
- **Aggregation Type**: Select the aggregation type from the drop-down menu. The options include:

  - AVERAGE
  - MINIMUM
  - MAXIMM
  - COUNT
  - SUM
- **Value Type**: Select the value type from the drop-down menu. The options include:
  - INT
  - DECIMAL
  - STRING
- **Value Integer**: Current value for an integer parameter.
- **Value Decimal**: Current value for a decimal parameter.
- **Value String**: Current value for a string parameter.
- **Description**: Type the description for the monitoring parameter.
- **Group Tag**: A tag to group monitoring parameters.
- **Widget Type**: Select the widget type to define the UI display variant of measured counters. The options include:
  - COUNTER
  - GAUGE
  - TEXTBOX
  - SLIDER
  - HISTOGRAM
  - BAR
- **Units**: Measured Counter Units (e.g., Packets, Kbps, Mbps, etc.)

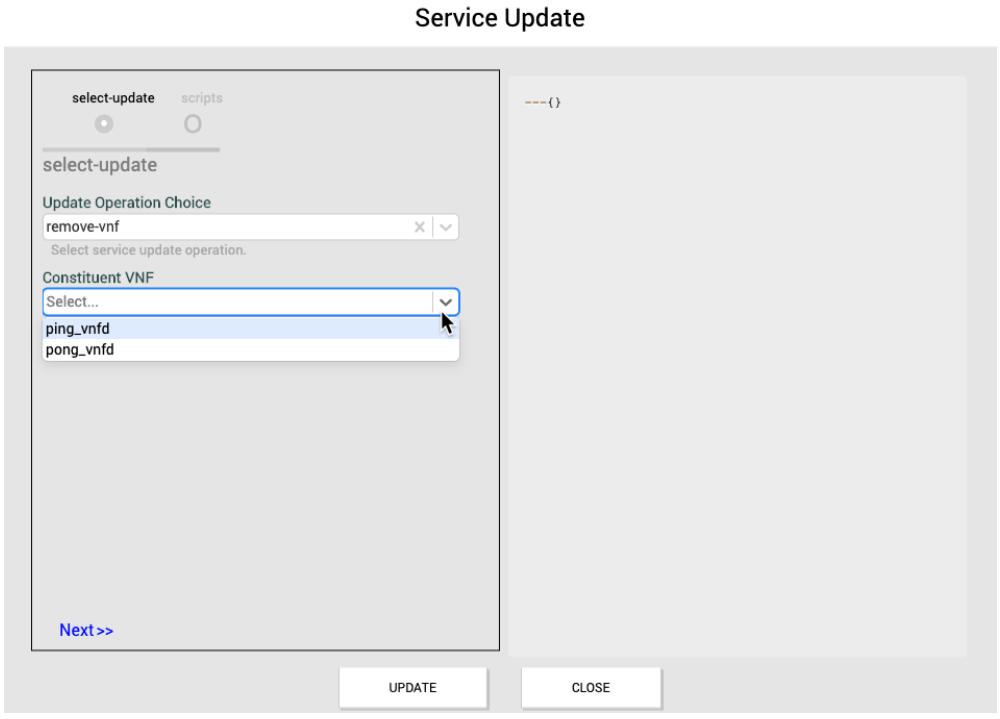
> **Note:** If you have additional monitoring parameter IDs, then type the ID in the ID 2 field and the same fields will appear for the subsequent monitoring parameters.

9. Click **UPDATE** and the insert progress bar appears.



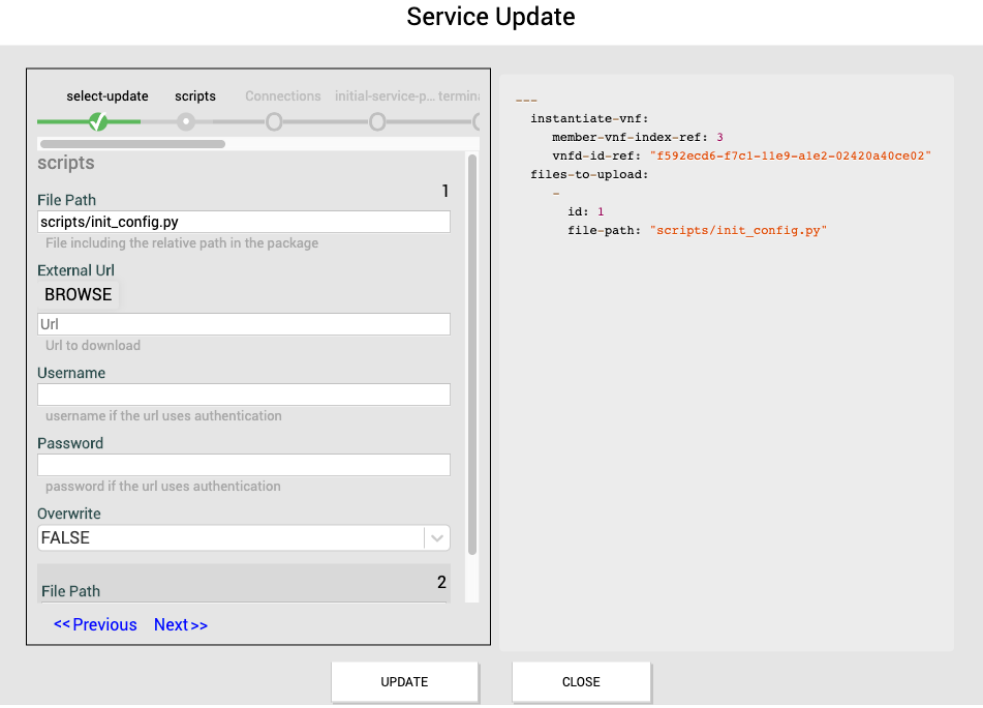

**Remove a VNF from a Running Service**

1. After choosing **remove-vnf**, additional fields appear. Provide the select-update details to remove a VNF from a Running Service:




- **Constituent VNF**: VNF to Remove. The options include the VNFs that are currently part of the running Service.

2. Click **Next >>** to provide the following details in the Scripts step to identify which scripts that you want to remove from the service:




- **File Path 1**: File including the relative path in the package. After you type the file name, additional fields appear.

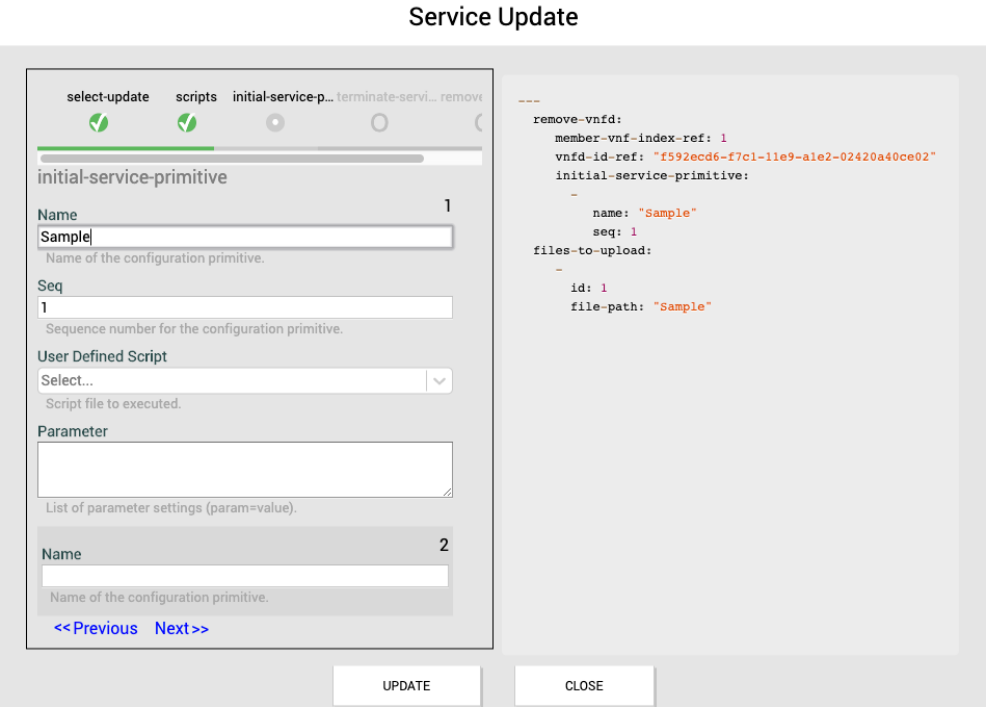
> **Note:** You must specify the script folder in this field.

- **External URL**: **BROWSE** or type the URL: URL to download. You can either browse or type your URL and add the script to the subfolder that you defined in the File path field.
- **Username**: Username if the url uses authentication.
- **Password**: Password if the url uses authentication.
- **Overwrite**: Overwrite the username and password in this field by choosing False (True or False)

> **Note:** If you have additional scripts, then type the path in File Path 2 and the same fields will appear for the subsequent scripts.

3. Click **Next >>** to provide the following details in the initial-service-primitive step:

> **Note:** This step is optional.




- **Name 1**: Name of the configuration primitive. After you type the configuration name, additional fields appear.
- **Seq**: Sequence number for the configuration primitive.
- **User Defined Script**: Select the script file to be executed.
- **Parameter**: List of parameter settings (param=value).

> **Note:** If you have additional initial-service-primitive, then type the name in Name 2 and the same fields will appear for the subsequent initial-service-primitives.

4. Click **Next >>** to provide the details in the terminate-service-primitive and remove-vnf-service-primitive steps. These steps include all same details as the initial-service-primitive step.

> **Note:** These steps are optional.

5. Click **UPDATE**.

## Model Enhancements

The following changes have been made to the nsr:update-network-service-v2.yang model to add the new topic:

**Input Fields**

| Name | Type | Cardinality | Description |
|---|---|---|---|
| project-name | String | 1 | Name of the associated project. |
| nsr-id-ref | String | 1 | NSR ID |

| | | | |
|---|---|---|---|
| ns-update-type | String | 1 | NS update type (Only instantiate-vnf and remove-vnf supported in Release 7.2.) |
| files-to-upload | List | 1 | Set of new assets needed for the NSR. |
| instantiate-vnf | Container | 1 | Details about the instantiate-vnf update NS operation. |
| remove-vnfd | Container | 1 | Details about the remove-vnf update NS operation. |
| deployment-profile | Container | 1 | Details about the deployment profile details for the update NS operation. |

**files-to-upload fields**

| Name | Type | Cardinality | Description |
|---|---|---|---|
| id | Int | 1 | Unique ID within this list of files to be uploaded. |
| file-path | String | 1 | File including the relative path in the package. |
| changelog-revision | String | 1 | Revision ID of the committing package change. |
| changelog-comment | String | 1 | User comment for the committing package change. |
| external-url | rwt:uri | 1 | URL to download. |
| username | String | 1 | Username if the URL uses authentication. |
| password | String | 1 | Password if the URL uses authentication. |
| overwrite | String | 1 | Overwrite the username and password |

**instantiate-vnf fields**

| Name | Type | Cardinality | Description |
|---|---|---|---|
| member-vnf-index-ref | String | 1 | New member-vnf within constituent-vnfds. |
| vnfd-id-ref | String | 1 | A reference to a VNFD in VNFD catalog. |
| connection-point | List | 1 | List of VNF-VL mapping. |
| Initial-service-primitive | List | 1 | Set of updated initial service primitives for this NS. |
| terminate-service-primitive | list | 1 | Set of updated terminate service primitives for this NS |
| add-vnf-service-primitive | list | 1 | Set of add VNF service primitives for this constituent VNF |
| remove-vnf-service-primitive | list | 1 | Set of remove VNF service primitives for this constituent VNF |
| monitoring-param | list | 1 | Set of NS monitoring params referring to newly added VNF monitoring params |
| service-primitive | list | 1 | List of NS manual service primitives |

**connection-point fields**

| Name | Type | Cardinality | Description |
|---|---|---|---|
| name | string | 1 | Name of the VNF connection point |
| vld-id-ref | string | 1 | Id of the VLD to connect to |

**initial-service-primitive, terminate-service-primitive, add-vnf-service-primitive and remove-vnf-service-primitive**

| Name | Type | Cardinality | Description |
|---|---|---|---|
| seq | int | 1 | Sequence number for the primitive |
| name | string | 1 | Name of the primitive |
| user-defined-script | string | 1 | A user defined script |
| parameter | list | 1 | Parameter |

**parameter fields**

| Name | Type | Cardinality | Description |
|---|---|---|---|
| name | string | 1 | Name of the parameter |
| value | string | 1 | Value of the parameter |

**monitoring-param fields**

| Name | Type | Cardinality | Description |
|---|---|---|---|
| id | string | 1 | Name of the parameter |
| name | string | 1 | Value of the parameter |
| value-type | | 1 | |
| numeric-constraints | container | 1 | |
| text-constraints | container | 1 | |
| value-integer | int | 1 | Current value for an integer parameter |
| value-decimal | | 1 | Current value for a decimal parameter |
| value-string | string | 1 | Current value for a string parameter |
| group-tag | string | 1 | A tag to group monitoring parameters |
| widget-type | | 1 | Defines the UI Display variant of measured counters |
| units | string | 1 | Measured Counter Units |
| aggregation-type | enum | 1 | aggregation-type |
| vnfd-monitoring-param | list | 1 | A list of VNFD monitoring params |

**numeric-constraints fields**

| Name | Type | Cardinality | Description |
|---|---|---|---|
| min-value | unit64 | 1 | Minimum value for the parameter |
| max-value | unit64 | 1 | Maximum value for the parameter |

**text-constraints fields**

| Name | Type | Cardinality | Description |
|---|---|---|---|
| min-length | unit8 | 1 | Minimum string length for the parameter |
| max-length | Unit8 | 1 | Maximum string length for the parameter |

**vnfd-monitoring-param fields**

| Name | Type | Cardinality | Description |
|---|---|---|---|
| member-vnf-index-ref | string | 1 | Reference to member-vnf within constituent-vnfds<br>This is a leafref to path: ../../../../nsd:constituent-vnfd + [nsd:id = current()/../nsd:id-ref] + /nsd:vnfd-id-ref<br>**Note:** An issue with confd is preventing the use of xpath. Seems to be an issue with leafref to leafref, whose target is in a different module. Once that is resolved this will switched to use leafref. |
| vnfd-id-ref | string | 1 | A reference to a VNFD |
| vnfd-monitoring-param-ref | string | 1 | A reference to the VNFD monitoring param |

**service-primitive fields**

| Name | Type | Cardinality | Description |
|---|---|---|---|
| name | STRING | 1 | Name of the service primitive. |
| parameter | LIST | 1 | List of parameters for the service primitive. |
| parameter-group | LIST | 1 | Grouping of parameters which are logically grouped in UI |
| vnf-primitive-group | LIST | 1 | List of service primitives grouped by VNF. |
| user-defined-script | STRING | 1 | A user defined script. |

**parameter fields**

| Name | Type | Cardinality | Description |
|---|---|---|---|
| name | string | 1 | Name of the parameter. |
| data-type | ENUM | 1 | Data type associated with the name. |
| description | STRING | 1 | Description of the primitive parameter |
| mandatory | BOOLEAN | 1 | Is this field mandatory |
| default-value | STRING | 1 | The default value for this field |
| parameter-pool | STRING | 1 | NSD parameter pool name to use for this parameter |
| read-only | BOOLEAN | 1 | The value should be dimmed by the UI. Only applies to parameters with default values. |
| hidden | BOOLEAN | 1 | The value should be hidden by the UI. Only applies to parameters with default values. |

**parameter group fields**

| Name | Type | Cardinality | Description |
|---|---|---|---|
| name | string | 1 | Name of the parameter. |
| parameter | List | 1 | List of parameters for the service primitive. |
| mandatory | BOOLEAN | 1 | Is this field mandatory |

**vnf-primitive-group fields**

| Name | Type | Cardinality | Description |
|---|---|---|---|
| member-vnf-index-ref | string | 1 | Reference to member-vnf within constituent vnfds |
| vnfd-id-ref | String | 1 | A reference to a vnfd. This is a leafref. |
| vnfd-name | String | 1 | Name of the VNFD. |
| primitive | List | 1 | |

**primitive fields**

| Name | Type | Cardinality | Description |
|---|---|---|---|
| Index | Unit32 | 1 | Index of this primitive. |
| Name | String | 1 | Name of the primitive in the VNF primitive. |

**remove-vnfd fields**

| Name | Type | Cardinality | Description |
|---|---|---|---|
| member-vnf-index-ref | string | 1 | Reference to member-vnf within constituent-vnfds |
| vnfd-id-ref | string | 1 | A reference to a VNFD |
| initial-service-primitive | list | 1 | Set of updated initial service primitives for this NS |

| terminate-service-primitive | list | 1 | Set of updated terminate service primitives for this NS |
|---|---|---|---|
| remove-vnf-service-primitive | list | 1 | Set of remove VNF service primitives for this constituent VNF |

**RPC samples**

**instantiate-vnf**

```
{

    "input": {
        "project-name": "default",
        "nsr-id-ref": "ad7b78ae-def9-4e55-8d38-d914361eab6a",
        "ns-update-type": "instantiate-vnf",
        "instantiate-vnf": {
            "member-vnf-index-ref": "3",
            "vnfd-id-ref": "8ff3262f-a1ba-4b82-acbd-4b06f1b18465",
            "connection-point": [{"name": "ping_vnfd/cp1", "vld-id-
ref": "ping_pong_vld1"}, {"name": "ping_vnfd/cp0", "vld-id-ref":
"mgmt_vld"}],
            "monitoring-param": [
                   {"id": "6", "name": "MYCOUNT1", "value-type": "INT",
"widget-type": "COUNTER", "aggregation-type": "AVERAGE", "vnfd-
monitoring-param": [{"member-vnf-index-ref": 3, "vnfd-id-ref":
"8ff3262f-a1ba-                                    4b82-acbd-
4b06f1b18465", "vnfd-monitoring-param-ref": "1"}]},
                   {"id": "7", "name": "MYCOUNT2", "value-type": "INT",
"widget-type": "COUNTER", "aggregation-type": "AVERAGE", "vnfd-
monitoring-param": [{"member-vnf-index-ref": 3, "vnfd-id-ref":
"8ff3262f-a1ba-                                    4b82-acbd-
4b06f1b18465", "vnfd-monitoring-param-ref": "2"}]}
                   ]
          }
     }
}
```

**remove-vnfd**

```
{
    "input": {
        "project-name": "default",
        "nsr-id-ref": "71c4068a-0788-4e41-9573-fe1ffc72deb4",
        "ns-update-type": "remove-vnfd",
        "remove-vnfd": {
            "member-vnf-index-ref": "3",
            "vnfd-id-ref": "f1e3f920-6ff6-11e9-90e6-02420a40db24",
            "terminate-service-primitive": [{"seq": 2, "name":
"NEWscript2", "user-defined-script": "NEWstop2.py"}, {"seq": 4, "name":
"script4", "user-defined-script": "stop4.py"}]
        }
    }
}
```

# Improved Amazon Web Services Support

This feature adds improved support for Amazon Web Services (AWS) Accounts.

- UI Changes
  - AWS Account Validation
  - SRIO-V Support
- Support for Alarms using Amazon SNS and CloudWatch

After implementing this feature, users will see a Network Service failure notification early if it fails during the creation of connection points or while associating a public IP address to connection points.

When creating an AWS account, you can create a snapshot to use instead of an image like OpenStack. If you require modifications to the snapshot, then you make changes by editing the running volumes and saving it again. When setting up the account you can:

- Create a fresh volume and attach it to an instance.
- Create a volume based on a snapshot and attach it to an instance.
- Attach an already existing volume to an instance.

**Note:** In release 7.2.0, scaling is not supported in AWS accounts.

## UI Changes

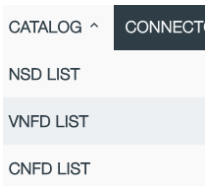
### AWS Account Validation

Support for AWS is implemented in 7.2 for VIM account validation. See the Datacenter Section Added to the UI section for steps on how to set up an AWS VIM account.

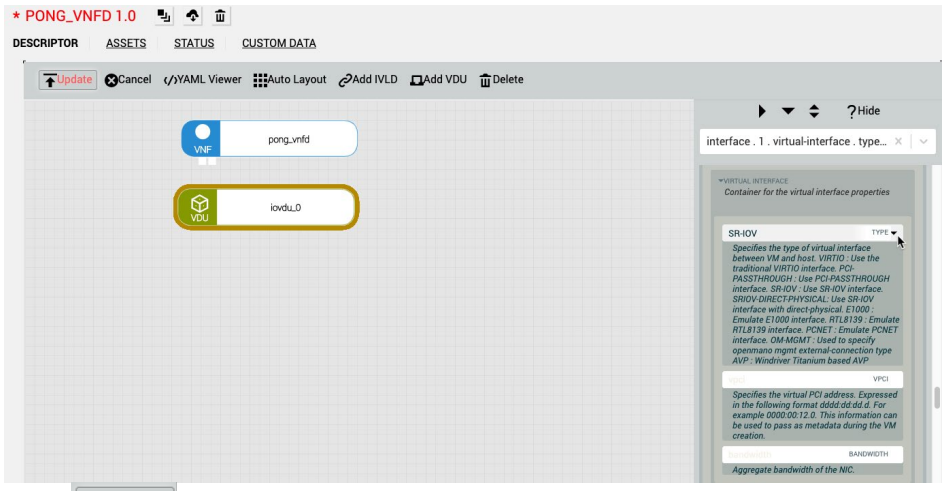
### SRIO-V Support

This feature enhances networking using Single Root I/O Virtualization (SRIO-V) to provide high-performance networking capabilities on support instance types. A user can enable SRIO-V drivers on an EC2 instances by setting the SRIO enabled flag in VNFD -> interface.

1. On the **Launchpad** menu, click **CATALOG** > **VNFD LIST** to open the list of catalog items.




2. Click next to the ping_vnfd or pong_vnfd that you want to see in more detail. The **DESCRIPTOR** tab opens.
3. Search for **VIRTUAL INTERFACE** field to enable the **SR-IOV** flag.




4. Click Update.

**Note:** If you enable the SRIO flag and the image does not support SRIO, then the Network Service fails.

## Support for Alarms using Amazon SNS and CloudWatch

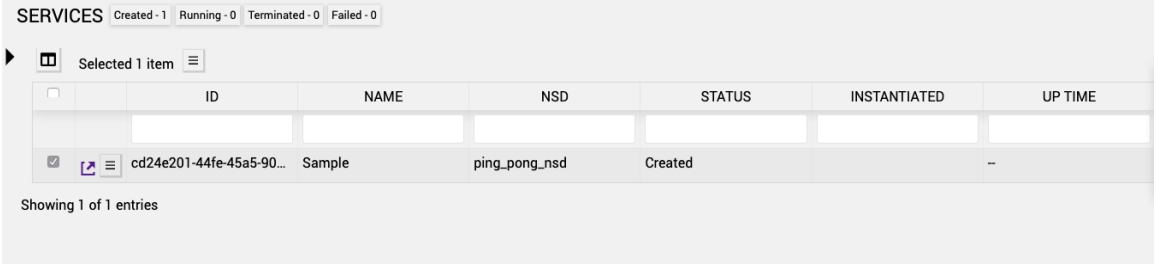
Amazon SNS is used to set notifications from AWS to the external world. RIFT.ware Launchpad registers one endpoint for notifications on SNS. Amazon CloudWatch is a monitoring tool that provides data for situations like rebooting, shutting down and powering off of Virtual Machines. As part of instantiation, Launchpad registers a particular instance on CloudWatch to listen for erroneous situations. CloudWatch then sends this notification to Launchpad using SNS. Once Launchpad receives a notification that a particular VM has failed, then RIFT.ware marks the Network Services as failed.

# Mapping Existing Service Elements to Existing Resources during Instantiation

This feature introduces a new capability available in the instantiation UI that allows user to add existing resources into a running NS. When you are instantiating a service, it is not possible to map existing service elements to existing resources.

The updated workflow is specific to the NSD from which the service is created. The NSD in the below instantiation workflow is ping_pong_nsd.

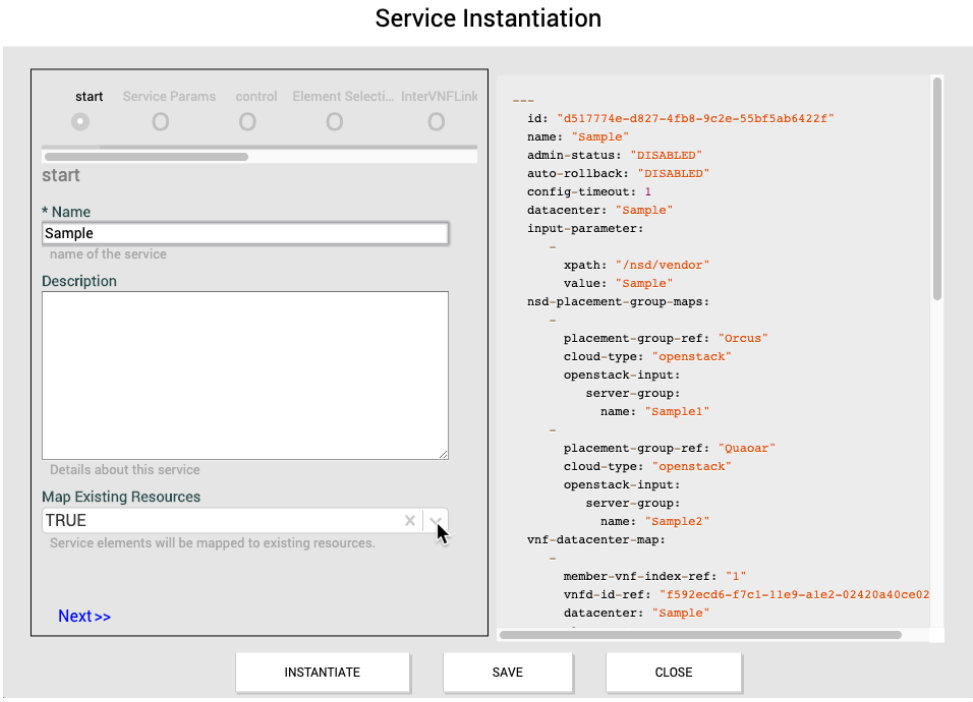
1. On the **Launchpad** menu, click **SERVICES**.
2. Click the ≡ button to edit a service.




An account wizard appears. Fill in the appropriate details for the specific service. The right side of the wizard screen displays what can be changed using this edit option.

**Note:** You cannot navigate forward without filling in required fields (*). The **Next >>** button will not function if the previous required field isn't populated. The **Save** button allows you to save your NSR config content and edit it later if you are not ready to Instantiate.

3. Provide the following details in the Service Instantiation step:
   - ***Name**: Name of the service
   - **Description**: Details about this service
   - **Map Existing Resources**: Service elements will be mapped to existing resources – Choose **True** from the drop-down menu in this field to map existing service elements to existing resources.

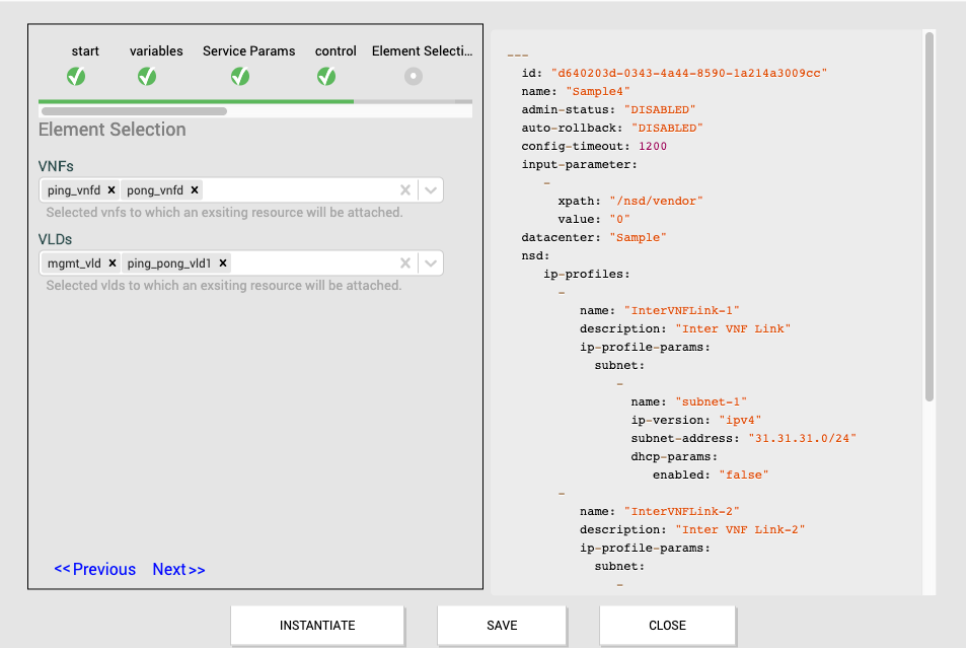



The steps to instantiate a service that will use existing resources mapped to elements defined in the service are customized for your NSD. The Network Service Instantiation section includes more information on designing and editing specific variables in an NS prior to instantiation. Two of the steps specific to this process are Element Selection and Preexisting VLDs.

**Note:** There are steps to configure each VNF elected in the Element selection step.

4. Click **Next >>** to provide the following details in the Element Selection step:
   - **VNFs**: Selected vnfs to which an existing resource will be attached. You can select an option for each VNFD that is part of the NSD.
   - **VLDs**: Select vlds to which an existing resource will be attached. You can selection an option for each VLD that is part of the NSD.

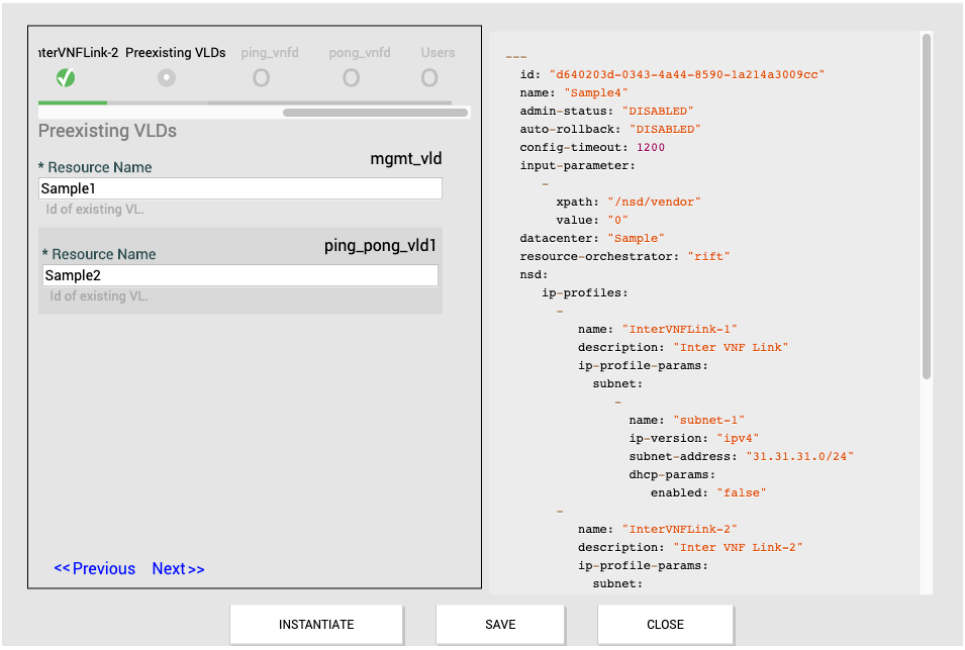
Service Instantiation




**Note:** By default, all options are included so click the **X** next to an element if you want to remove it.

5. Click **Next >>** to provide the following details in the Preexisting VLDs step:
   - ***Resource Name**: ID of existing VL

Service Instantiation




*

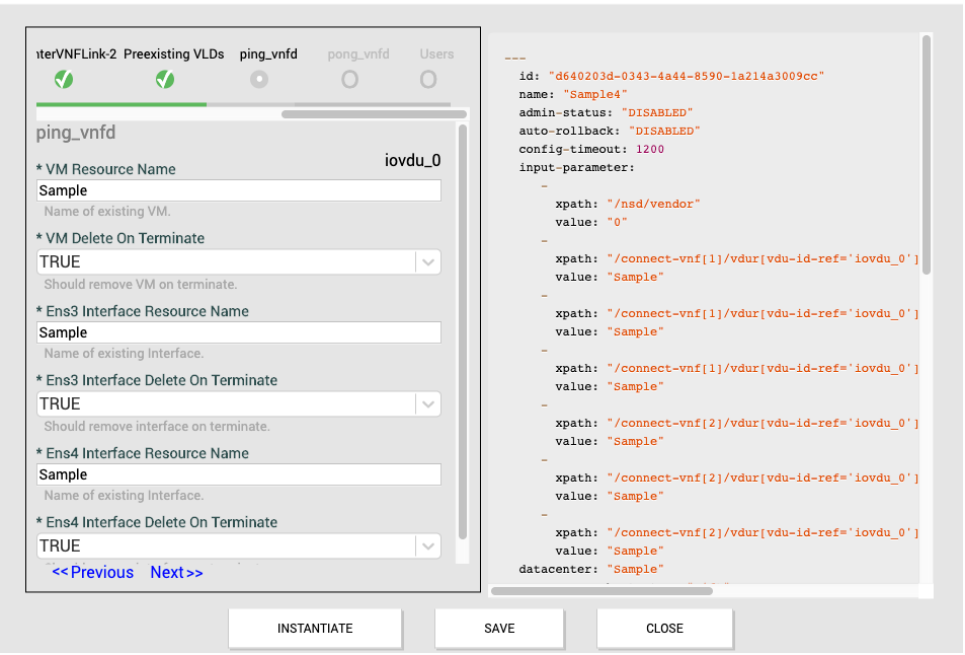
6. Click **Next >>** to provide following details in the VNFD step:

If there are multiple versions of the same vnfd in your service, then ping_vnfd will include the member index at the top of this page to differentiate the versions.

- ***VM Resource Name**: Name of existing VM. There is a section in this step for each VDU defined in the VNFD.
- ***VM Delete On Terminate**: Should remove VM on terminate. (True or False)
- ***Interface Resource Name**: Name of existing Interface.
- ***Interface Delete On Terminate**: Should remove interface on terminate. (True or False)

**Note:** If the vdu in your service has volumes defined, then they will appear in this ping_vnfd details page.

Service Instantiation



# Network Service Instantiation Updates

This feature simplifies the process of designing and editing specific variables in an NS prior to instantiation. The workflow changes slightly depending on how you are instantiating your service.

**Note:** There is no change to the process of creating a service.



The updated workflow is specific to the NSD from which the service is created. The NSD in the three instantiation workflows is ping_pong_nsd.

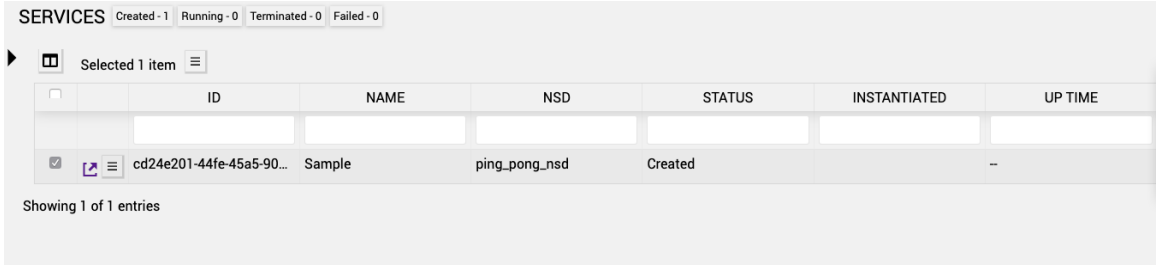
## Instantiating an NS Process Updates

The new process saves changes to the NSR config so the user will not have to instantiate in this form. It is possible to exit the form and come back later. If you skip a step that does not include required fields, then Launchpad uses the default data.

If you try to navigate away from the instantiation page, then a prompt appears asking if you want to save data prior to leaving the page.

**Note:** The above prompt appears if pop-ups for the site are enabled.

1. On the **Launchpad** menu, click **SERVICES**.
2. Click next to the NS in the Service page for an NS that has not been instantiated and the current configuration of the service appears.
   This page also includes the button to start the instantiation process for a service or a delete a service.




   A Service Instantiation screen appears. Fill in the appropriate details for the specific service. The right side of the screen displays what can be changed using this edit option.

**Note:** You cannot navigate forward without filling in required fields (*). The **Next >>** button will not function if the previous required field isn't populated.

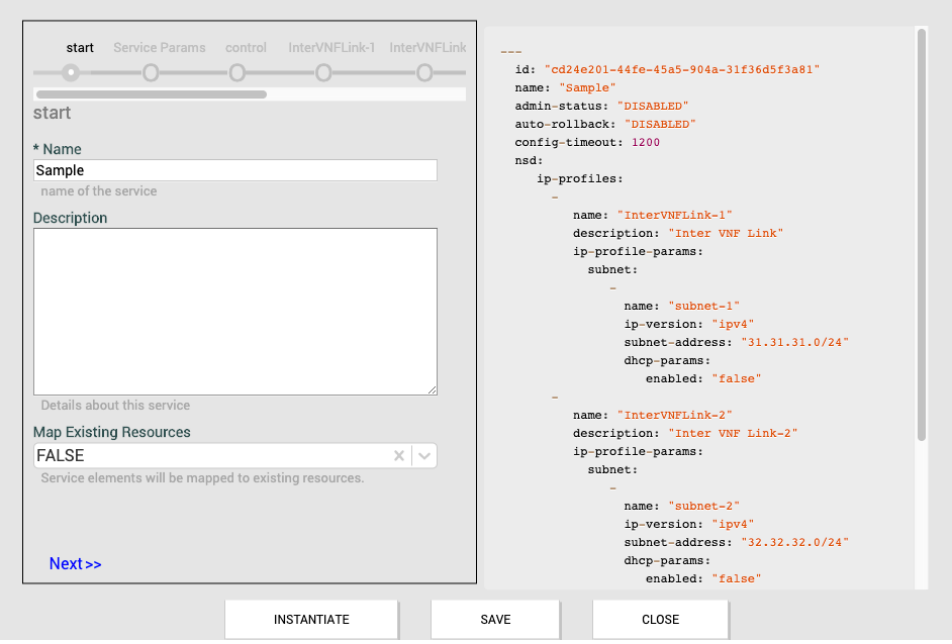
**Note:** The Save button allows you to save your NSR config content and edit it later if you are not ready to Instantiate.

3. Provide the following details in the Service Instantiation step:
   - ***Name**: Name of the service
   - **Description**: Details about this service
   - **Map Existing Resources**: Service elements will be mapped to existing resources (True or False)

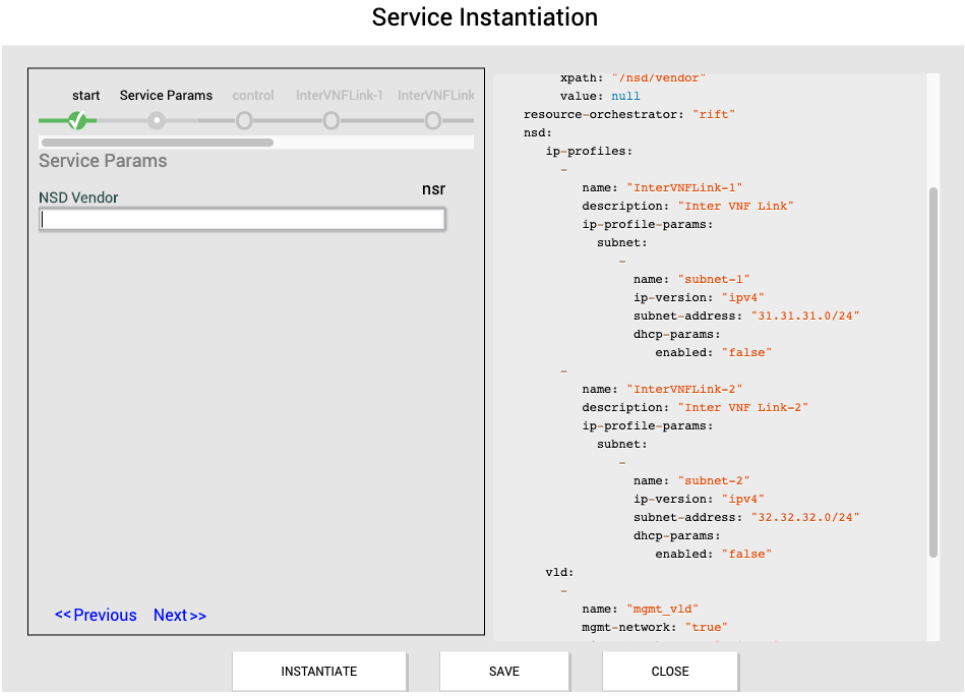
The new NS instantiation process is customized for the NSD from which the service was created. Therefore, if the NSD has xpath Variables or Instantiation Variables, then the process is slightly different. You can refer to the steps for these processes. The remaining steps in this section are specific to the ping_pong_nsd.

- Instantiating a Service with xpath Variables
- Instantiating a Service with instantiation Variables

Service Instantiation



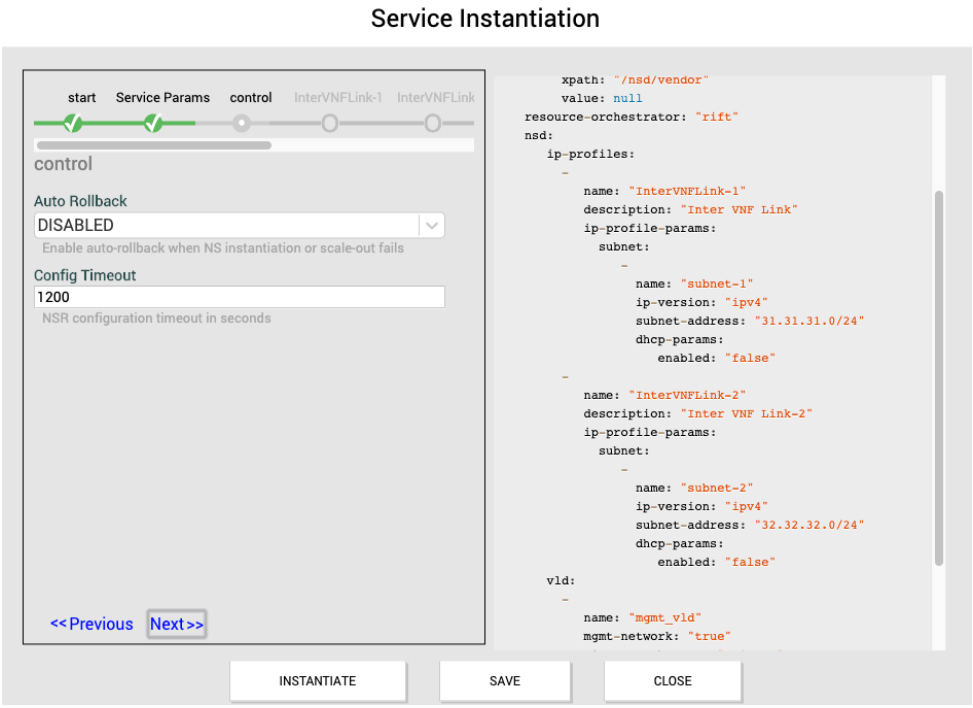

4. Click **Next >>** to provide the following details in the Service Params step:
   - **NSD Vendor**: nsr



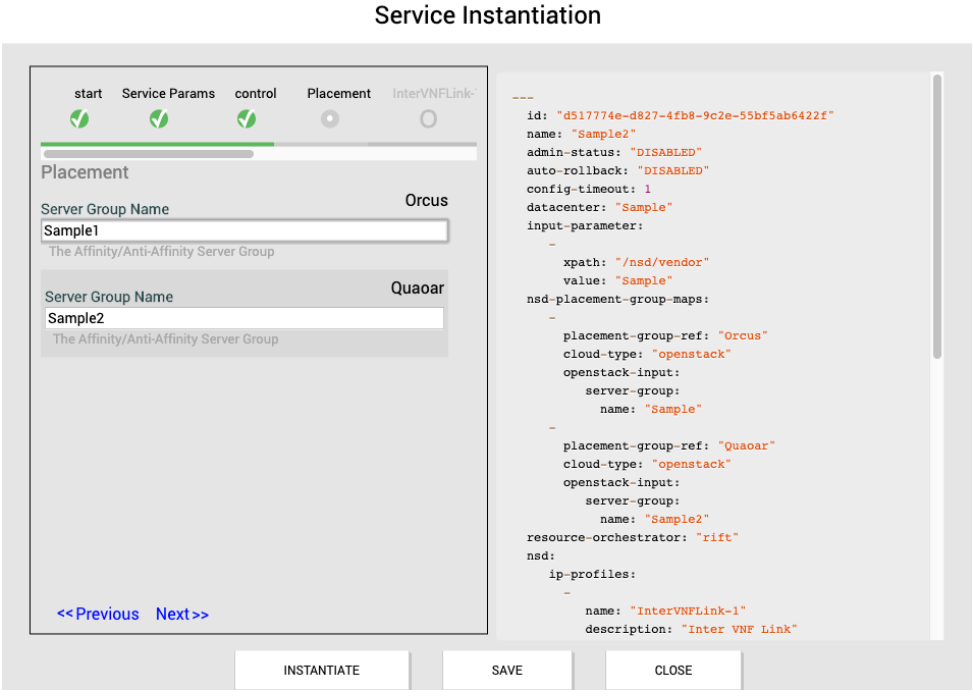

5. Click **Next >>** to provide the following details in the Control step:

> **Note:** This step can appear earlier in the process if there are no instantiation-variable or input-parameter-xpath entries.

- **Auto Rollback**: Enable auto-rollback when NS instantiation or scale-out fails
- **Config Timeout**: NSR configuration timeout in seconds
- ***Resource Orchestrator**: Orchestrator to use for allocation/managing service resources. This option only appears if you have more than one RO defined in your service.
- ***Datacenter**: Default datacenter form which to appropriate service resources. This option only appears if you have more than one datacenter defined in your service.




6. Click **Next >>** to provide the following details tin the Placement step.
   In this specific workflow, this option appears because the NSD has placement groups defined.



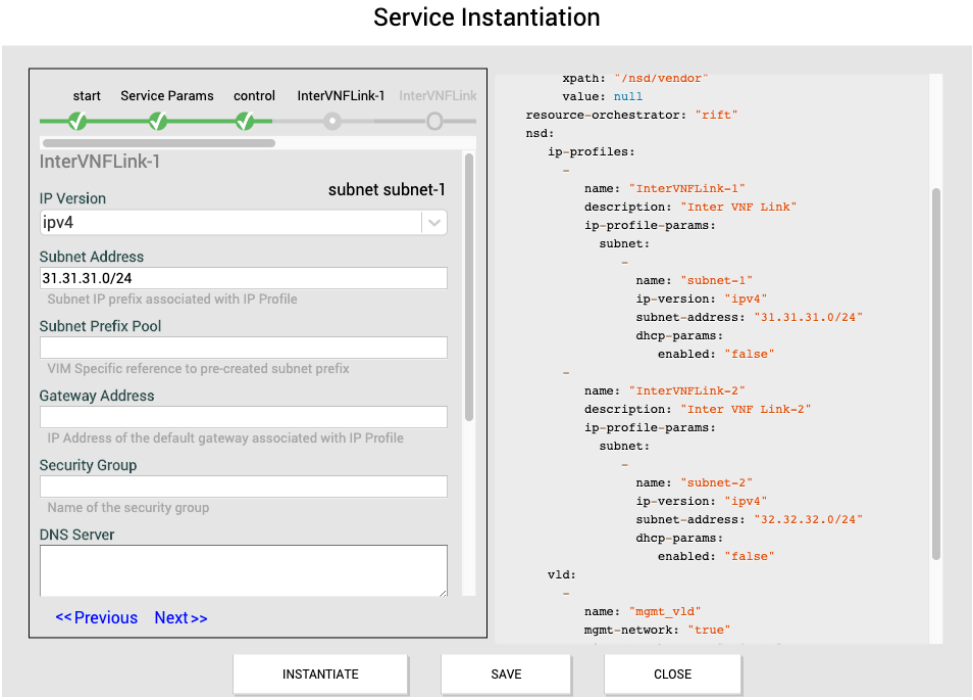

- **Server Group Name Orcus**: The Affinity/Anti-Affinity Server Group
- **Server Group Name Quaoar**: The Affinity/Anti-Affinity Server Group

7. Click **Next >>** to provide the following details in the InterVNFLink-1 step to map an ip profile from the NSD.

In this specific workflow, there are two profiles defined (InterVNFLink-1 and InterVNFLink-2). The name of the step in the UI matches that of the ip profile. An ip profile can define a list of subnet definitions which can be configured in this section. Each subnet definition has a name that appears on the right side of the UI. In this InterVNFLink profile, one subnet is defined, and it is named subnet-1. The input fields do not change.

**subnet subnet-1**

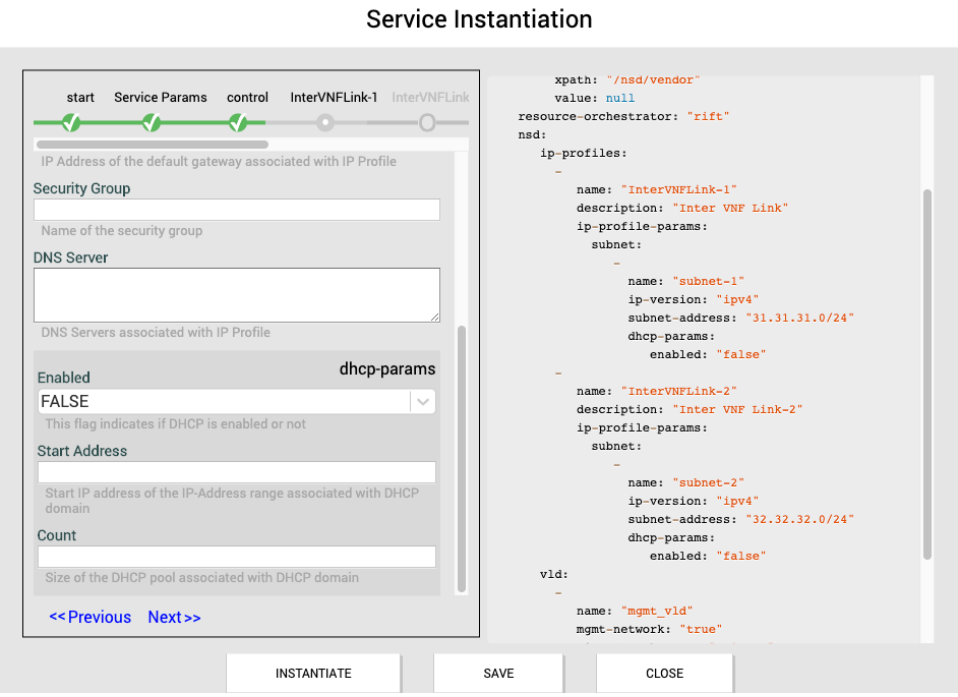
- **IP Version**: Select the IP version. The options include:
  - Unknown
  - ipv4
  - ipv6
- **Subnet Address**: Subnet IP prefix associated with IP Profile
- **Subnet Prefix Pool**: VIM specific reference to pre-created subnet prefix
- **Gateway Address**: IP Address of the default gateway associated with IP Profile
- **Security Group**: Name of the security group
- **DNS Server**: DNS Servers associated with IP Profile




**dhcp-params**

- **Enabled**: This flag indicates if DHCP is enabled or not (True or False)
- **Start Address**: Start IP address of the IP-Address rang associate with DHCP domain

- **Count**: Size of the DHCP pool associated with DHCP domain




8. Click **Next >>** to provide the following details in the InterVNFLink-2 step to map an ip profile from the NSD:

   **subnet subnet-1**

   - **IP Version**: Select the IP version. The options include:
     - Unknown
     - ipv4
     - ipv6
   - **Subnet Address**: Subnet IP prefix associated with IP Profile
   - **Subnet Prefix Pool**: VIM specific reference to pre-created subnet prefix
   - **Gateway Address**: IP Address of the default gateway associated with IP Profile
   - **Security Group**: Name of the security group
   - **DNS Server**: DNS Servers associated with IP Profile

   **dhcp-params**

   - **Enabled**: This flag indicates if DHCP is enabled or not (True or False)
   - **Start Address**: Start IP address of the IP-Address rang associate with DHCP domain
   - **Count**: Size of the DHCP pool associated with DHCP domain

9. Click **Next >>** to provide the following details in the VLDs step for all the VLDs that are defined in the NSD.

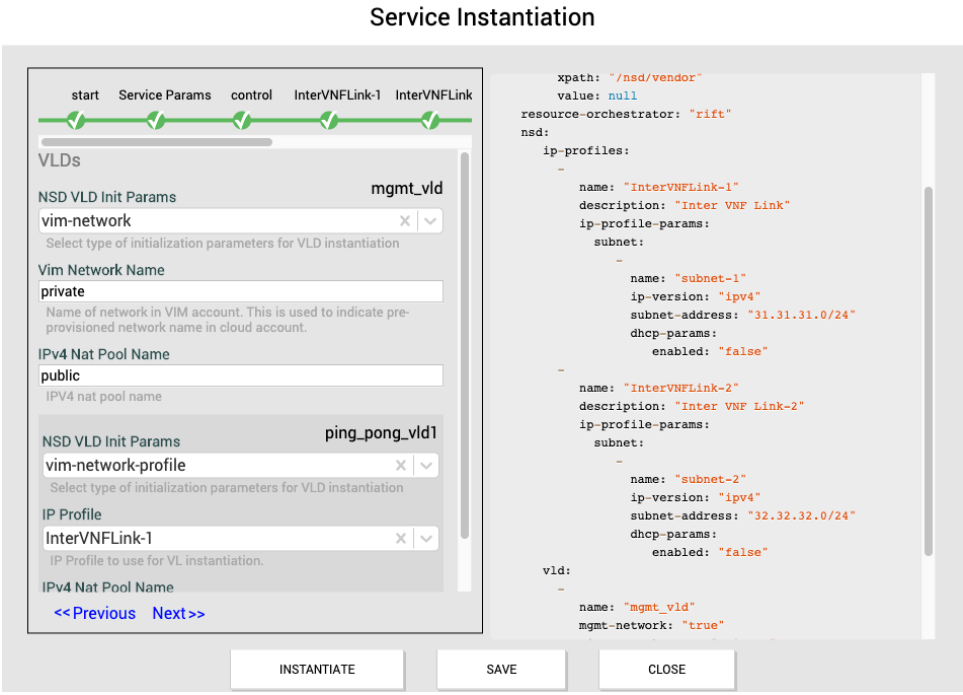
   In this specific workflow, the NSD defines two VLDs mgmt_vld and ping_pong_vld1. The specific vlds for this NSD fully describe NSD VLD Init Params, VIM Network Name, VIM-network-profile and Ipv4 Nat Pool Name. This NSD has some default information so the UI preselected the InitParam option vim-network for mgmt_vld and vim-network-profile for ping_pong_vld1.

   **mgmt_vld**

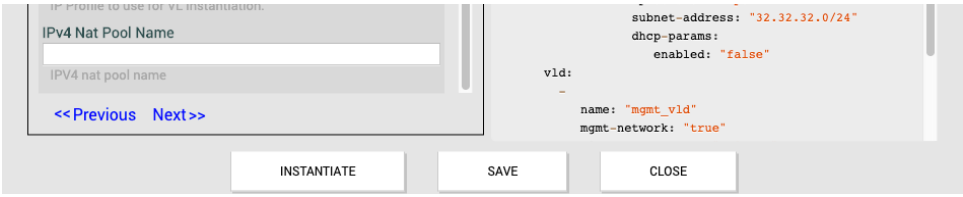
   - **NSD VLD Init Params**: Select type of initialization parameters for VLD instantiation. The options include:
     - Vim-network
     - Vim-network-profile
   - **Vim Network Name**: Name of network in VIM account. This is used to individuate pre-provisioned network name in cloud account.
   - **Ipv4 Nat Pool Name**: IPV4 nat pool name

   **ping_pong_vld1**

   - **NSD VLD Init Params**: Select type of initialization parameters for VLD insanitation. The options include:
     - Vim-network
     - Vim-network-profile
   - **IP Profile**: IP Profile to use for VL instantiation. The options include:
     - InterVNFLink-1
     - InterVNFlink-2




- **IPv4 Nat Pool Name**: IPv4 pool name




10. Click **Next >>** to provide the details in the Datacenter step to select the datacenter on which to place the VNF and to configure the placement groups.

    This step is repeated for each VNFD in the NSD and the content can change depending on how you have configured your datacenter. In this specific workflow, the VNFD has a placement group named Eris defined and there are a few fields that can be configured for the placement group.

    Note:

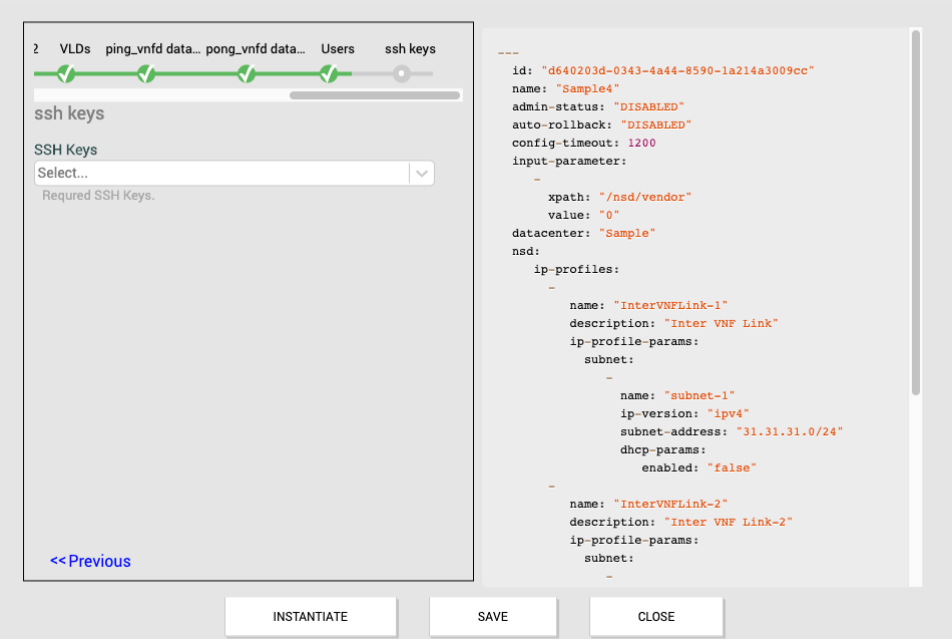
    > **Note:** If your VNFD has no placement groups defined and no datacenter to pick, then this step does not appear.

    - **Server Group Name Eris**: The Affinity/Anti-Affinity Server Group
    - **Datacenter:** Select the Datacenter to use when instantiating the VNF.

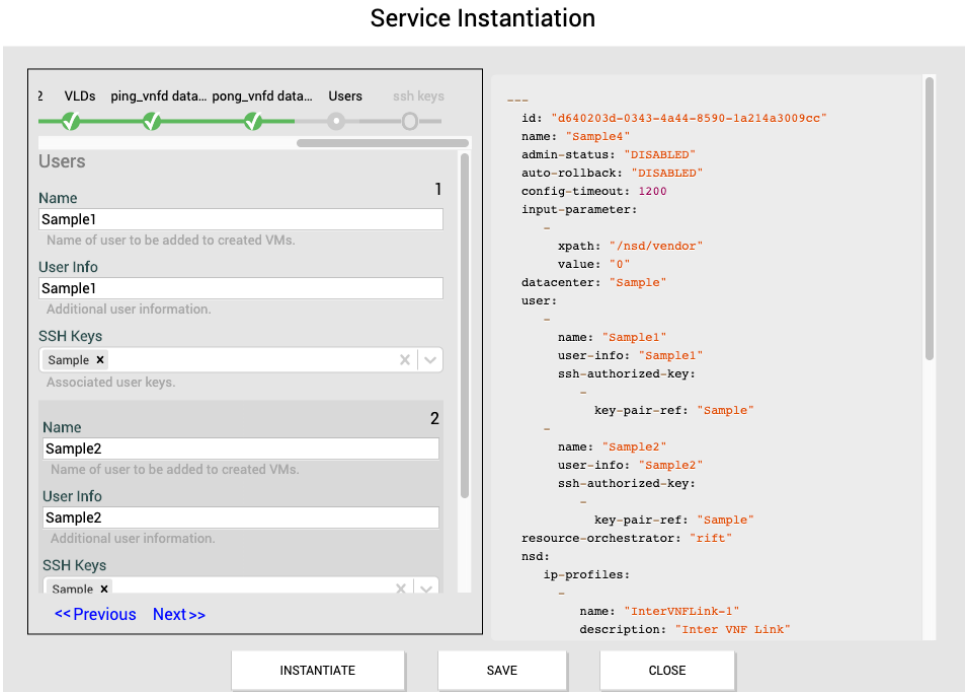
11. Click **Next >>** to provide the details in the ssh keys step:
    - **Ssh Keys**: Use the drop-down menu to select all the keys you want copied to the create VMs of this service.

**Note:** If you do not have ssh keys to find, then this step is not present in the UI.

Service Instantiation




12. Click **Next >>** to provide the following details in the Users step. This information defines user accounts that will be created on all the VMS created for a specific service.
    - **Name**: Name of the user account. After you type a name into this field, additional fields become available.
    - **User Info**: Additional user information.
    - **SSK Keys**: Associated user keys.




13. Carefully check your entries. Click **INSTANTIATE** to start the instantiation of the service or click **SAVE** if you are not ready to instantiate. Return to the Service Instantiation process at later date to continue with the configuration.

**Instantiating a Service with xpath Variables**

If you set up NSD xpath variables in the input-parameter-xpath list property when you create a service, then you'll see different options while instantiating the service. In this workflow, the following sections appear in the service parameters step:

- start
- control
- service parameters
- ssh keys

If you are configuring a service with xpaths, then there is a separate Service Parameters step. In this step, you can provide values for all the input-parameter-xpath entries defined in the NSD. The values for this step are dependent on what is defined in your NSD.

**Instantiating a Service with instantiation Variables**

If there are instantiation variables defined in the NSD from which this service was created, then you'll see different options while instantiating the service. In this workflow, the Variables step appears. Please see the Variables step in the Instantiating an NS Process Updates Section.

**Note:** Instantiation variables come before the control variables because they can impact what information you are able to provide in the control section. Also, if your NSD has instantiation variables defined, then the second step is variables which hare used in the input-parameter-xpath. If all the input-parameter-xpaths are configured with a variable, then there will not be a service parameters step. However, if not all of them are configured with a variable, then the step will appear to provide values for those without variables.

# Or-Vnfm Account (ETSI GS NVF-SOL 003) Enhancements during Instantiation

RIFT.ware now supports heal and heal-retry options for Or-Vnfm accounts. This feature also introduces a retry option at the NSR level for instantiate and terminate Network Service (NS) operations for SOL 003. It is also possible to rollback at the NSR level after instantiation.
These operations are only possible if you have configured a sVNFM account in Launchpad and select the appropriate VNFM Account in the **NS/VNF ACCOUNT PLACEMENT** section while instantiating. See Add a VNFM Account.

**Note:** There is no scale support in SOL 003. RIFT does not support VNF scaling at this time. This is required for adding scale support to Or-Vnfm interface.



## Heal Operation

### Heal and Heal-Retry

An operator can attempt to heal a VM. See the Support for NS and VNF Heal section for details on how to restart a VM which is in the **RUNNING** state.

### Auto-Heal

Auto-heal is a process that starts from the sVNFM account. Notifications are sent to RIFT.ware to start the heal process and update operational data. This process is done automatically and there are no actions required from the user.

## Retry Operation

If the NSR reaches a failed state during instantiate, terminate or heal, then a user can retry the operation again. After the failure occurs, click **TAKE ACTION** and a dialog box appears to retry the operation. After the user clicks Retry, the NSR operation may be successful or it might fail again. Check the Events tab for debugging details to determine the cause of the failure.

### Instantiate

The retry option allows users to attempt to instantiate again if the operation fails. When a user attempts to retry instantiation, the NSR attempts to retry the VL instantiation and the VNF instantiation. If one of the VNFRs changes to the **RUNNING** state, then the NSR

state might also change to **RUNNING** depending on how many other VNFRs are pending. However, if some VNFR operations fail, then the NSR state will return to **FAILED**.
If the second attempt is successful, then the NS status changes to **RUNNING** and **CONFIGURED**.

**Note:** The retry attempt does not change any existing resources.

**Terminate**
The retry option allows users to attempt to terminate if the operation fails. When a user attempts to retry termination, the NSR attempts to retry the VNF termination and VL termination. If one of the VNFRs is termination, then the NSR state might also change to Terminated depending on how many other VNFRs are pending. However, if some VNFR operations fail, then the operation needs to try the NS terminate Retry again.
If the second attempt is successful, then the NS status changes to **TERMINATED**.

## Rollback Operation

**Instantiate**
If instantiation fails, then a user can also choose to rollback the request. This option releases all the elements that were allocated during the failed instantiation. This is similar to terminating the NS.

# Os-Ma-Nfvo Interface (ETSI GS NVF-SOL 005) Enhancements during Instantiation

RIFT.ware now supports heal and heal-retry options for Os-Ma-Nfvo interface. This feature also introduces a retry option at the NSR level for instantiate, terminate, scale-out, scale-in and heal Network Service (NS) operations for SOL 005. It is also possible to rollback and scale-out at the NSR level after instantiation.

The heal operation is only possible if you have instantiated a network service and it is in the running state. An operator can use the following SOL 005 APIs to check on the service:

- *GET /ns_instances* SOL 005 API to see if the service is in INSTANTIATED state
- *GET /ns_lcm_op_occs* SOL 005 API to check the status of the Sol005 Instantiate request.

Set the operationState field to COMPLETED in the response.

The retry and rollback options are only possible if the Life Cycle Management operation is in the Failed state. RIFT.ware indicates this as FAILED or FAILED_TEMP (FAILED_TEMP is for scaled operations). An operator can use the GET /ns_lcm_op_occs SOL 005 API to check on the state. Set the operationState field to FAILED_TEMP regardless of the LCM.



## Heal Operation

### Heal and Heal-Retry

An operator can attempt to heal a VM. See the Support for NS and VNF Heal section for details on how to restart a VM which is in the **RUNNING** state.

## Retry Operation

If the NSR reaches a failed state during instantiate, terminate or heal, then a user can retry the operation again. After the failure occurs, click **TAKE ACTION** and a dialog box appears to retry the operation. After the user clicks Retry, the NSR operation may be

successful or it might fail again. Check the Events tab for debugging details to determine the cause of the failure.

**Instantiate**
The retry option allows users to attempt to instantiate again if the operation fails. When a user attempts to retry instantiation, the NSR attempts to retry the VL instantiation and the VNF instantiation. If one of the VNFRs changes to the **RUNNING** state, then the NSR state might also change to **RUNNING** depending on how many other VNFRs are pending. However, if some VNFR operations fail, then the NSR state will return to **FAILED**.
If the second attempt is successful, then the NS status changes to **RUNNING** and **CONFIGURED**.

**Note:** The retry attempt does not change any existing resources.

**Terminate**
The retry option allows users to attempt to terminate again if the operation fails. When a user attempts to retry termination, the NSR attempts to retry the VNF termination and VL termination. If one of the VNFRs is termination, then the NSR state might also change to Terminated depending on how many other VNFRs are pending. However, if some VNFR operations fail, then the operation needs to try the NS terminate Retry again.
If the second attempt is successful, then the NS status changes to **TERMINATED**.

**Scale-in**
The retry option allows users to attempt to scale-in again if the operation fails. If the second attempt to scale-in again for the failed scale group is successful, then the NS status changes to **RUNNING else FAILED_TEMP**.

**Scale-out**
The retry option allows users to scale-out again if the operation fails.

## Rollback Operation

**Instantiate**
If instantiation fails, then a user can also choose to rollback the request. This option releases all the elements that were allocated during the failed instantiation. This is similar to terminating the NS.

**Scale-Out**
If scale out fails, then a user can also choose to rollback the scale-out request. This option releases all the elements that were allocated during the failure. This is similar to Scale-in.

# RIFT RO and VIM Re-synchronization

This feature introduces a new audit capability in the **DATACENTERS** menu option. A user can click AUDIT to run a report that displays inconsistences between the expected Launchpad state and the actual peer entity state.

**Note:** This audit process is per all services running in a project. This feature is supported on OpenStack versions pike and above and VMware vCD CD 8.2 and above.



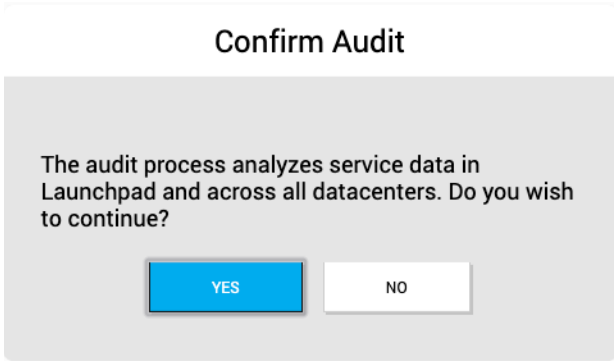
## UI Enhancements

In order to run an audit report, a user needs to set up a federation id and then create a cloud datacenter account. See **Datacenter Section Added to the UI** for more details on how to run an audit report. If a user is running the report that did not create the resources, then that user must have admin privileges.

In order to run an audit report, you must instantiate an NS and it needs to be in the running state. If the NS is in instantiation or termination phase, then the UI will display inaccurate resources.

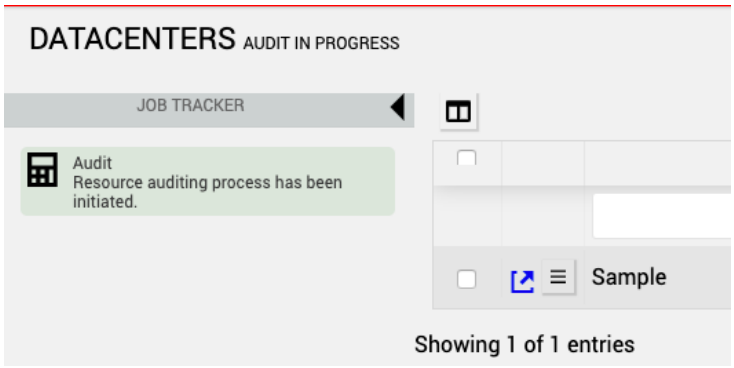
**Note:** Once you start the audit process, it is not possible to stop it until the audit is successful, fails or times out.

1. On the **Launchpad** menu, click **DATACENTERS**.
2. Click the check box to the left of a specific service and the click  button to audit a specific service. A **Confirm Audit** screen appears.

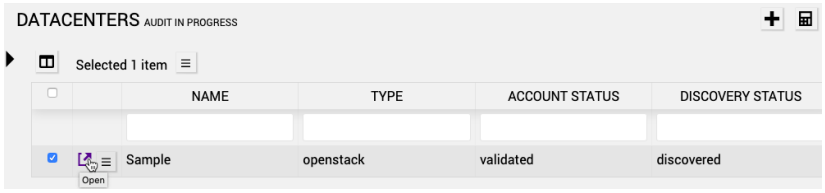



3. Click **YES** to start the audit process. When the audit process begins, the Job Tracker screen appears on the left of the screen indicating **Audit Resource auditing process has been initiated**.

   If the audit attempt is successful, then **AUDIT IN PROGRESS** appears next to **DATACENTERS** at the top of the screen.

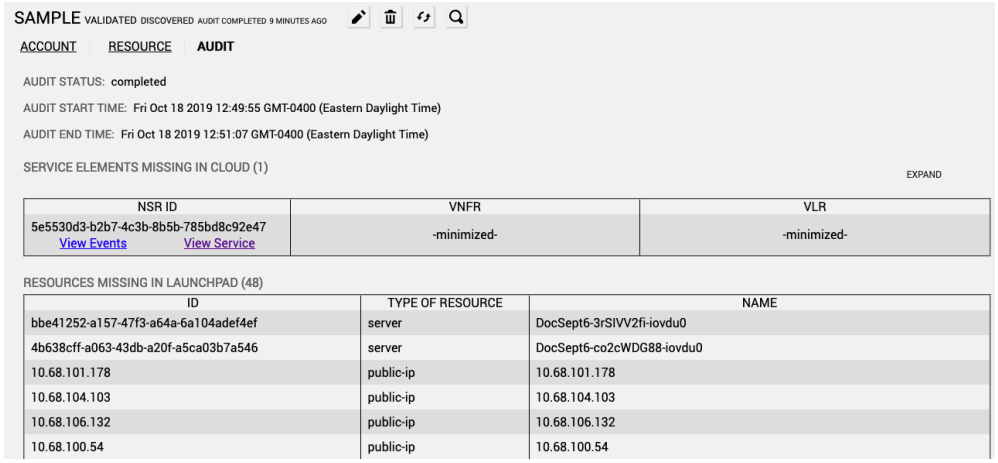



4. Next, click the arrow to open a specific service.



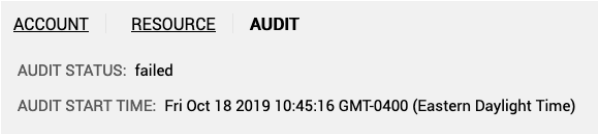

5. After you open the account, you'll see the account details. There is also a RESOURCE section and an AUDIT section for the account. Click AUDIT.
6. While the audit is in progress, a progress bar appears in the **AUDIT** section. The audit time varies depending on the specific cloud account and the amount of resources in that account. The timeout maximum is 15 minutes. If the process reaches the 15-minute mark, then the audit fails.


The audit details appear.

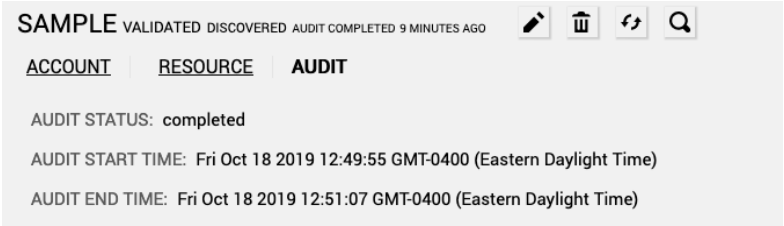


If the audit is successful, then a time stamp appears at the top of the screen: **AUDIT COMPLETED X XXXXX AGO**.

If the audit is not successful, then the message **AUDIT STATUS: failed** appears. You can open to the Events tab to see the root cause of the issue. After identifying and resolving the issues, the user can run the audit again. For more information on checking Events, see the Notifications guide.




If there are no missing service elemetns in the cloud or missing resources in the Launchapd, then the user will see that the audit is complete with no issues.




**Note:** Only one audit can be run at a time. If you make a second attempt to run an audit, you will receive an error message.

The audit details screen includes two sections: **SERVICE ELEMENTS MISSING IN CLOUD** and **RESCOURES MISSING IN LAUNCHPAD**.

### Service Missing in Cloud

The **SERVICE MISSING IN CLOUD** provides resources references that exist in the Launchpad but are unreachable in the cloud account. The details include the NSR ID, VNFR and VLR.

#### NSR ID

If there is a service missing in the cloud, then the NSR ID section includes a View Events link so you can see all the events listed for that particular service. For more information on checking Events, see the Notifications guide. There is also a View Service link, so that you can open the specific service for more information.

#### VNFR and VLR

You can click EXPAND on the right side of the screen to view more information about the missing VNFR and VLR in the cloud. You can also click MINIMIZE to collapse the screen.


### Resources Missing in Launchpad

The **RESCOURES MISSING IN LAUNCHPAD** section lists resources in the cloud that Launchpad fails to delete. The details include the ID, type of resources and the name of the resource.

## Support for Kubernetes as VIM

Support to Kubernetes as VIM is an experimental lab feature in release 7.2.0. It requires support from RIFT. If you have questions related to this feature, then contact RIFT.ware Services & Support (support@riftio.com).

# Support for NS and VNF Heal

NS and VNF Heal is a new Life Cycle Management operation. A user can restart a VM which is already in the **RUNNING** state using the heal feature. If heal fails, then a user can do a heal-retry. SOL005 and SOL003 standards do not support VDU level healing.

**Note:** Retry for instantiation, scale-out, scale-in and termination do not impact this feature.

For SOL003, it is not possible to identify if there is an SVNFM account as part of the service. If there is an expectation for 'additionalParams', then it is possible to pass those parameters. However, there is no standard and each SVNFM may behave differently.

For SOL005, RIFT will pass VDURs in 'additionalParams' to force VDU level healing.

**Note:** The Heal supports OpenStack and VCD for RIFT VNFM accounts.

See the Or-Vnfm Account (ETSI GS NVF-SOL 003) Enhancements during Instantiation and Os-Ma-Nfvo Interface (ETSI GS NVF-SOL 005) Enhancements during Instantiation for additional details on healing for SOL003 and SOL005.

- Heal
- Heal-Retry
- UI Enhancements
- Model Enhancements

A user can deploy Aodh services in OpenStack for RIFT.ware to receive alarms from the VIM. These alarms indicate that an error occurred with the VM. After RIFT.ware receives an alarm, then Launchpad decides if it is necessary to bring down the previously successfully instantiated NS or trigger a heal operation. For details on deploying Aodh service in OpenStack, refer to the OpenStack Aodh Documentation.

## Heal

An operator can restart a VM when a Network Service is in the Running state. When a user attempts to Heal the service, the status in the UI shows the VNF-Heal progress. RIFT.ware processes a Heal on each VDU. The heal options are restart vdu and reconfigure vdu.

- **Restart VDU**: This heal option attempts to restart a VM on the running NS. This is the default operation.
- **Reconfigure VDU**: The heal option means that only pre-heal primitives are run on the service. Before heal, pre_heal_service_primitive will run (not applicable for SOL003).

During the NSR Heal operation, the status can either change to Running (successful) or Failed. If one of the VNFRs in the service changes to the running state, then the NSR status can change to Running depending on how many other VNFRs are pending.

SOL003 supports both manual and auto-heal.

## Heal-Retry

If the first heal attempt fails, then an operator can try to restart the VM again. This action can only be taken when the NSR is in the Failed state. A retry can also be made when scale instance state is in the failed-temp state. When a user attempts to retry the heal, the status in the UI shows the VNF-Heal progress. RIFT.ware processes a Heal on each VDU. The heal options are restart and no operation.

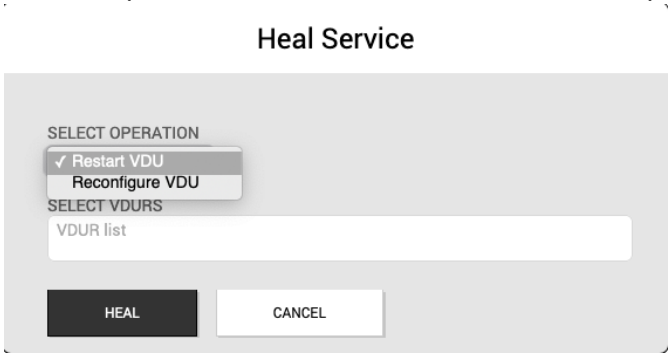
- **Restart VDU**: This heal option attempts to restart a VM on the running NS. This is the default operation.
- **Reconfigure VDU**: This heal option means that only pre-heal primitives are run on the service. Before heal, pre_heal_service_primitive will run (not applicable for SOL003). If you choose this option, you are reconfiguring the VDU. Reconfigure VDU does not restart the VM.

During the NSR Heal operation, the status can either change to Running (successful) or Failed. If one of the VNFRs in the service changes to the running state, then the NSR status can change to Running depending on how many other VNFRs are pending.

## UI Enhancements

There is now a **Heal** button in the **SERVICES** section in the UI. This option allows users to heal a VM.

1. On the **Launchpad** menu, click **SERVICES**.
2. Click the ≡ button next to a specific service to start the heal process. If you open a running NS, then click the ✚ button at the top of the screen.
3. Choose **Restart VDU** or **Reconfigure VDU** in the **SELECTION OPERATION** drop down menu to attempt heal.
   - **Restart VDU**: This heal option attempts to restart a VM on the running NS. This is the default operation.
   - **Reconfigure VDU**: This heal option means that only pre-heal primitives are run on the service. Before heal, pre_heal_service_primitive will run (not applicable for SOL003). If you choose this option, you are reconfiguring the VDU. This reconfigure option does not restart the VM.
4. Click on xpath in the **SELECT VDURS** field and the possible vdurs appear.

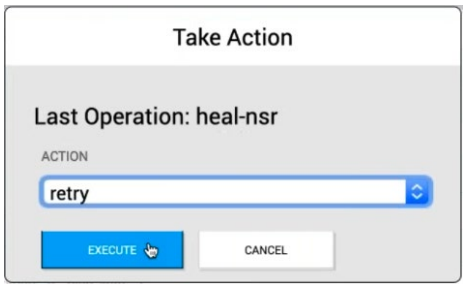



5. Choose the appropriate vdurs and click **HEAL**.

6. During the heal attempt process, the Services tab shows the progress state – **Heal in progress**.


7. If the heal attempt fails, then the state changes to **Vnf Heal failed**.


8. If the heal attempt fails, then an error message appears.
9. Click TAKE ACTION and a **Last Operation: heal-nsr** dialog box appears.




Choose **retry** and click **EXECUTE** to attempt the heal again.

**Note:** There is no rollback option for a heal operation.

The VNF-Heal state will no longer appear when the NS instantiation is successful.

You can check the Events logs to see information about the success or failure of the heal attempt.

## Model Enhancements

Added the following information to the async-transaction-command in the nsr.yang model to heal and auto-heal an operation on an asynchronous NS transaction.

- Trigger the retry option for a failed NS
- vfr-list
- vdur
- NSR Changes

**Trigger the Retry, Rollback or Continue Option for a Failed NS**

Input

**Fields**

| Name | Type | Cardinality | Description |
|---|---|---|---|
| project-name | string | 1 | Name of associated project |
| nsr-id-ref | string | 1 | NSR ID |
| vfr-list | list | 1 | List of vnfr-ids and corresponding vdur_ids which need be healed |

**vnfr-list**

Input

**Fields**

| Name | Type | Cardinality | Description |
|---|---|---|---|
| vdur | list | 1 | List of associated vdur |

**vdur**

Input

**Fields**

| Name | Type | Cardinality | Description |
|---|---|---|---|
| id-ref | string | 1 | vdur id |
| operation | string | 1 | Name of the operation that is executed on vdur |

Output

**Fields**

| Name | Type | Cardinality | Description |
|---|---|---|---|
| operational-status | string | 1 | Status of the operation |

**NSR Changes**

The NSR will now store the last Life Cycle Management record with all the sub tasks. It will also keep a record of the new heal-in-progress state while attempting to heal the VM.

Sample:

```
nsd-catalog in nsd-base.yang- added following fields to the model

    list pre-heal-service-primitive {
      description
        "Pre heal service primitives for NSD.";
      key "name";
      leaf name {
        type string;
      }
      leaf ns-service-primitive-name-ref {
        description "Reference to the NS service primitive";
        type leafref {
          path "../../service-primitive/name";
        }
      }
    }
    list post-heal-service-primitive {
      description
        "Post heal service primitives for NSD.";
      key "name";
      leaf name {
        type string;
      }
      leaf ns-service-primitive-name-ref {
        description "Reference to the NS service primitive";
        type leafref {
          path "../../service-primitive/name";
        }
      }
    }

ns-instance-opdata in nsr.yang

        list pre-heal-service-primitive {
          description
            "Pre heal service primitives for NSD.";
          key "seq";
          uses heal-primitive;
        }

        list post-heal-service-primitive {
          description
            "Pre heal service primitives for NSD.";
          key "seq";
          uses heal-primitive;
        }
.....................
  grouping heal-primitive {
```

```
      leaf seq {
        description "Sequence number for the service primitive.";
        type uint64;
      }
      leaf ns-service-primitive-name-ref {
        description "Reference to the NS service primitive";
        type string;
      }
  }

vnf-configuration in rwmano-types.yang

list pre-heal-config-primitive {
  description
    "Pre heal config primitives for VNF.";
  key "seq";

  leaf seq {
    description
      "Sequence number for the pre heal configuration primitive.";
    type uint64;
  }
  leaf config-primitive-ref {
    description
      "Reference to config-primitive by name";
    type string;
  }
}

list post-heal-config-primitive {
  description
    "Post heal config primitives for VNF.";
  key "seq";

  leaf seq {
    description
      "Sequence number for the post heal configuration primitive.";
    type uint64;
  }
  leaf config-primitive-ref {
    description
    "Reference to config-primitive by name";
    type string;
  }
}
```

# Fixed Issues in RIFT.ware 7.2.0

| Support Tickets | Title | Description |
|---|---|---|
| RIFT-22950 | Ability to upload certs from UI. | Added an upload button to upload a certificate from a file.<br>This issue is resolved. |
| RIFT-23531 | New catalog table should have copy, export, delete at top of table. | Added a button at the top of a table in the Catalog tab to copy, export or delete services. Now the opt<br>This issue is resolved. |
| RIFT-24273 | Config Mgr: Python helper library for config scripts. | Created a rw-config-util python3 module that can be used to create configuration scripts for NSD and VNFD. This new module provides a standard interface for all configuration scripts. The script no longer has to parse different input data based on the Life Cycle Management event that triggered the script.<br>This issue is resolved. |
| RIFT-24399 | Topology: Arrange layout dynamically. | In previous releases, if a service had many VMs then all of them were displayed horizontally on the Topology tab in the UI. Updated the Topology screen so that the elements appear in a grid according to the number of items present.<br>This issue is resolved. |
| RIFT-24589 | List the Discovered Virtual machines along with associated respective ports/networks/resources. | The discovered VMs are now listed along with the respective ports, networks and resources. This provides additional information to create VDUs in real-time use cases.<br>This issue is resolved. |
| RIFT-24811 | WebUI not properly handling different Projects opened in different browser Tabs. | A user had two browser tabs open with the same version of Launchpad.<br>'Project1' was selected on one tab and 'Project2' was selected on the other tab. In the above scenario, there were a few issues with the data-polling that was done from the UI. RIFT.ware does not support working on multiple projects at the same time. |

| Support Tickets | Title | Description |
|---|---|---|
| | | After this fix, a user can only work on one project at a time so that the polling issues will not occur.<br>This issue is resolved. |
| RIFT-24884 | NS failure state should be able to distinguish between external failures and internal failures. | Retry functionality is used to re-run a particular phase of Network Service Instantiation.<br>Upon receiving AODH alarm events, RIFT.ware marked the NS as failed. The failed state was not sufficient enough for RIFT.ware to identify which NS needed a retry attempt.<br>Added FAILED-EXT-FACT to identify this specific situation. In this scenario, a user must click the Retry button to attempt instantiation again.<br>This issue is resolved. |
| RIFT-24923<br>RIFT-24581 | Support heal for SOL005. | Added heal support for SOL 005.<br>This issue is resolved. |
| RIFT-24961 | Offer heal operation on VNFs within a running NS. | It is necessary to provide a heal option when an NS is in the running state.<br>There is now an option to heal an NS in the running state on the NS listing page and the NS details page. When a user clicks the heal button, a dialog box appears. The user can select the operation type and the VDURs to heal.<br>This issue is resolved. |
| RIFT-25080 | async-transaction-command rpc is not showing status correctly. | async-transaction-command RPC was used to get the status of all async RPCs. The 'record-status' field of the output of async-transaction-command status query should show the current status but it was not working correctly.<br>If an operator used this RPC to identify the status of any async RPC, the user was not notified when the operation finished.<br>Now the async-transaction-command status-query RPC works as expected.<br>This issue is resolved. |

| Support Tickets | Title | Description |
|---|---|---|
| RIFT-25083 | Blank page seen on clicking Topology while using sol3 descriptors. | When SOL3 services were instantiated, the UI topology section was not loading any visualizations. This was because RIFT.ware was missing some required data about internal VLs. Also, the internal connection points were not populated in operational data.<br>After this fix, RIFT.ware auto fills required information from the RIFT package and service details even though SOL3 plugin does not send all the required information.<br>This issue is resolved. |
| RIFT-25162 | AODH: Check the vm status in vnfm plugin before marking VNFR to failed. | If user tried to HEAL a Network Service by restarting a VM, then an alarm is generated which marks the NS as failed.<br>Added a solution to mark the NS as failed if the NS is not attempting a heal operation.<br>This issue is resolved. |
| RIFT-25169 | Take Action retry option does nothing for configuration failed case. | Added support for configuration failure retry option.<br>This issue is resolved. |
| RIFT-25212<br>RIFT-25307 | Add Tagging feature for Resource Auditing. | Added tags that can be used to insert into specific resources for auditing. The tags which can be used are Federation ID and Project name. They can be used to create resources such as VM, Networks, Subnets, etc.<br>Updated the cal layer to support the tags that are passed from the service layer so that they can be attached to resources. The current support clouds are OpenStack and VMware vCD.<br>This issue is resolved. |
| RIFT-25214 | Basic Audit: Compare and Publish LP and OpenStack resource leakage (without tag). | This issue is resolved. |
| RIFT-25215 | Audit Resource Model | Introduced a new model to show the audited result which can be seen in *show project default audit record*. |

| Support Tickets | Title | Description |
|---|---|---|
| | | This issue is resolved. |
| RIFT-25334 | VIM plugin: vim accounts list page - Quota status support. | Quotas can now be viewed as percentages on the datacenter list table.<br>This issue is resolved. |
| RIFT-25382 | NS termination success even when Launchpad is not able to reach es2: "termination failure" never happens. | If a running NS was terminated from RIFT.ware Launchpad and there was no connectivity between RIFT.ware and the OpenStack VIM, then RIFT.ware was reporting the NS termination successful. However, the OpenStack dashboard showed that the resources were not released yet because connectivity failed from Launchpad. Launchpad should validate the connectivity to the VIM account before starting the NS termination.<br>This issue is resolved. |
| RIFT-25426 | Error prompt for uploading a NSD before the dependent VNFDs needs to be improved. | When a user uploaded a NSD before a required VNFD, the following error message appeared:<br>**Onboarding failed: POST request error.**<br>This error message was not clear.<br>After this fix, a proper error message now appears in this situation.<br>If a required VNFD is absent the following error message appears:<br>**Required VNFD(s) not present.**<br>If a resource is missing in the VNFD, then the follow error message appears:<br>**VNF Resource missing in VNFD.**<br>This issue is resolved. |
| RIFT-25490 | Support heal feature from the service details page when the service is running. | Added support for deal support from the service details page when the service is running.<br>This issue is resolved. |
| RIFT-25491 | Create heal in progress view. | Added the ability to view the progress of the heal attempt during instantiation.<br>This issue is resolved. |
| RIFT-25492 | Implement heal retry. | Added an attempt to retry the heal operation in the Take Action dialog box if it fails during instantiation. |

| Support Tickets | Title | Description |
|---|---|---|
| | | This issue is resolved. |
| RIFT-25624 | Configuration manager rework. | Updated the configuration manager so all configuration execution occurs in the configuration manager.<br>This issue is resolved. |
| RIFT-25642 | Fix Pep8 and Review and Usability feedback | Added support for CNFs (Container Network Functions).<br>This issue is resolved. |
| RIFT-25643 | Fix Redis operational data recovery | Added support for CNFs (Container Network Functions).<br>This issue is resolved. |
| RIFT-25654 | Allow local helm charts to be installed. | Added support for CNFs (Container Network Functions).<br>This issue is resolved. |
| RIFT-25725 | Account Wizard and Datacenter View requires visual update. | Improved the datacenter view in the UI.<br>This issue is resolved. |
| RIFT-25772 | Include a "description" field for the Service Primitives parameters. | When listing service primitives in the NSD, it is possible to have multiple parameters in them. Previously, there was no description field in this section. This fix added a description field so that a user can add a short note on the significance of the parameter.<br>This issue is resolved. |
| RIFT-25783 | Datacenter: list page - Quota status dashboard | Added three columns in the Datacenter list page (cores, ram size and instances) with a graphical view in the column.<br>This issue is resolved. |
| RIFT-25975 | [7.1.1 release] [AWS] VIM account validation is not working. | Added account validation for AWS accounts to verify the credentials other information added by a user.<br>This issue is resolved. |
| RIFT-26097 | Upgrade 7.1.1 to 7.2.0: Services in failed state in new LP. | A user performed an HA upgrade. After upgrading Launchpad 2 (standby), the user shut off Launchpad 1 which was the active LP. After returning to LP2, the UI continuously asked the user to refresh. Also, the running configured instances failed.<br>This issue is resolved. |

| Support Tickets | Title | Description |
|---|---|---|
| RIFT-26161 | UI NS List API for Dashboard | Enhanced the performance of List Views in the UI.<br>This issue is resolved. |
| RIFT-26165 | UI NS List API for Services | Enhanced the performance of List Views in the UI.<br>This issue is resolved. |
| RIFT-26185 | Implement RFC8040 fields query params standard. | Enhanced a RIFT.ware API to allow queries with selected fields only. These fields can be selected across nested levels. This helps the API response time, especially when the user is querying for a list item.<br>This issue is resolved. |
| RIFT-26222 | Investigate API/UI Response Times for List Items. | This issue is resolved. |
| RIFT-26357 | NS with VNFFG stuck at init phase indefinitely. | A simple NS composed of 3 cirros based VNFs and a VNFFG was failing to instantiate. Upon instantiation, the NS was stuck at VNF-INIT indefinitely.<br>This issue is resolved. |

# Known Issues in RIFT.ware 7.2.0

| Support Tickets | Title | Description | Impact | Workaround |
|---|---|---|---|---|
| RIFT-13715 | Confd configuration transaction abort results in inconsistent state. | Confd can abort configuration change transactions due to its own internal reasons. RIFT.ware cannot undo the changes (on an ABORT form Confd) because it is already internally committed. | This could result in a data mismatch between that is there in the configuration database and what is known to the RIFT.ware backend. | This issue mostly occurs when a user makes successive config changes without any idle time in between the modifications.<br>Any test or automation script using RIFT REST APIs to complete configuration changes must have a delay between two successive config change operations. |
| RIFT-19249 | RBAC API to Update user's project/MANO roles has changed. | RBAC API to Update user's project/MANO roles has changed. This information needs to be updated in the User Documentation. | An older REST API cannot be used to change the User's Project/MANO roles and might break existing scripts. | Use the Launchpad UI to change a user's Project/MANO role, or use the New API. |
| RIFT-21978 | Password is displayed in plain text in event logs. | A password is displayed in plain text in the events log. | A password is displayed in plain text in the events log. | N/A |
| RIFT-23621 | Introduce a new field in VNFM account page for RIFT.ware<--->VNFM handshake URL. | This is a request to add a new field in the VNFM accounts page that the Operator so the operator can populate the URL for handshaking. If the URL field is empty, handshake is not required. | RIFT.ware uses a GET call to /vnf_instances to validate VNFM Account.<br>If a SVNFM account does not support GET on /vnf_instances, then the VNFM Account validation may Fail. The UI shows the | N/A |

| Support Tickets | Title | Description | Impact | Workaround |
|---|---|---|---|---|
| | | | account status as failed. If the SVNFM account supports notification, then RIFT.ware sends a subscription request next as part of the validation. This is successfully and the VNFM account status is successful. Therefore, a SVNFM account which doesn't support GET call on /vnf_instances, then the VNFM Account status will appear to fail. This will not block an further operations such as instantiation and termination. | |
| RIFT-23681 | VIM Account connection status shows failure which was a success before. | A VIM account is created successfully but then an error message appears showing 'Failure connection status'. Refreshing the account does not fix the issue. | Create a new account with the same details and the VIM account is created successfully. | Create a new VIM account with the same details. |
| RIFT-23760 | Rel-6.2: Terminating one service with the same name on any Launchpad interrupts existing | This issue occurs because all services on all Launchpads have the same UniqueID which is the InstanceID on ES2. When a user | If a user terminates any of the services on either of the network-services, this will make any other services orphan. | Use an unique NSR name if an organization is using multiple Launchpads. Ensure that an OpenStack tenant is |

| Support Tickets | Title | Description | Impact | Workaround |
|---|---|---|---|---|
| | services on all Launchpads. | terminates a service, then terminates Instance on ES2 which causes issues for all existing services because they have the same UniqueID. | | controlled by a single Launchpad.<br>**Note**: There are no plans to change this behavior. |
| RIFT-24343 | cpu stays high for rwmain tasklet due to msgbroker. | In some instances when many packages are uploaded or instantiations are done, the main rwmain process starts using 100% CPU. | In this scenario, the CPU usage is high because the lib dispatch library starts subscribing for more threads to the DTS router queue and each thread keeps polling out of the event loop. | N/A |
| RIFT-24706 | Second VL instantiation failing to take input params in latest R7.1.0.2. | A user instantiated and NS with a vim-network-name provided for both the mgmt-network and ping-pong-vld. During instantiation, NS failed at the VL-init stage. | N/A | If user is going to configure the vim-network-name in VLDs using input variable xpaths, then the ip-profile-ref value in the NSD should not be set. |
| RIFT-24872 | Updating ipv4 to ipv6 fails in the case of FIXED IP ADDRESS and vice versa. | A user modified the 'FIXED IP ADDRESS' field and then added an IPv4 address in the descriptor details pane for a VDU. Next the user updated the configuration to IPV6 and clicked the 'Update' button.<br>An error message appeared on the UI. | This issue only occurs when there are two IP addresses in interface configurations (IPv4 or IPv6) and the user tries to change one address to one with a different version. | Delete the existing address and add one new address with different a version. |
| RIFT-24990 | Launchpad is crashing with an | Launchpad had 3 projects with | N/A | N/A |

| Support Tickets | Title | Description | Impact | Workaround |
|---|---|---|---|---|
| | error- application communication failure. | brownfield discovered instances on all the projects. Launchpad stopped running when the user was checking the status of the network service. | | |
| RIFT-24992 | Running AWS NS fails after failover to a new LP. | A user created an NS on Launchpad 1. The service is in the running state for tiny descriptors with an AWS VM account. The user set up an HA pairing with another Launchpad of a different version. Failover occurred to Launchpad 2; the NS was that running in the Launchpad 1 failed in instantiate in Launchpad 2. | N/A | Terminate the NS and re-instantiate in the second LP. |
| RIFT-25384 | Kafka account in error state failover after an upgrade. | A user configured a new kafka account in Launchpad. After a failover occurred, the kafka account failed on the new Launchpad. | N/A | Reconfigure kakfa account post failover. |
| RIFT-25902 | Discovery gets triggered even on failed account due to non-reachable network status. | A user created a new VIM account to OpenStack. If the network is down and the cloud setup is not reachable, then when a user attempts discovery it fails but the process is still triggered. | N/A | N/A |

| Support Tickets | Title | Description | Impact | Workaround |
|---|---|---|---|---|
| RIFT-25610 | REL_7.1.1.0: AWS - NS stuck indefinitely in VNF-INIT phase when using a scaling-group and constituent VNF's start-by-default = False. | A user tried to instantiate a simple NS with one VNF, a single VDU (for an Ubuntu VM), and a single VLD. The NSD had a scaling group and the constituent VNFD's start-by-default value is False. When the NS is instantiated, it is stuck forever in the VNF-INIT phase. There are no errors seen in the Event Logs or in rift.log. | No VNF is started by default. Since the min-instance-count for scaling group is 1, the VNF is supposed to be scaled out once NS instantiates. | N/A |
| RIFT-26002 | Sol005 ns-lcm-op-occs details not recovered after LP reboot. | A user instantiated an NS using SOL005 APIs. The user verified that ns_lcm_op_occs API had a proper response. The user performed a restart/reboot of Launchpad and called the same API again. The reboot of Launchpad did not handle the op occurrence details and the response is empty. This scenario also occurs during HA failover. | The ns_lcm_op_occs API does not show any details. Since the status of an LCM triggered with sol005 is obtained using this API, the user will have to rely on other RIFT APIs/RPCs to get the status. For example, async-transaction-command status-query rpc. | The user can query ns-instance-opdata using rpc to obtain details of last LCM. If this is not a last LCM, then the user has to first find the transaction-id of the rpc from the logs and then use the async-transaction-command status-query rpc to get the status of that LCM. |
| RIFT-26478 | Audit button should be disabled when no datacenter is configured. | This is a request to disable the Audit button when no datacenter is configured in the UI. | The audit button is enabled even if no datacenter is configured. | N/A |

| Support Tickets | Title | Description | Impact | Workaround |
|---|---|---|---|---|
| RIFT-26481 | Launchpad blanked upon the presence of an wrong RO config. | A user configured the wrong resource orchestrator, and this caused the Launchpad to be unusable. | The HOME/Dashboard page of the Launchpad crashes if an RO with missing mandatory configuration is present. It is only possible to create an RO with the missing mandatory configuration from the CLI. It is not possible from the UI. | Delete the configuration using the CLI. |
| RIFT-26494 | Server Error 500 while deleting datacenter associated with SDN account. | A user created two SDN accounts with valid credentials and then navigated to the datacenters page. The operator tried to create one by selecting OpenStack → corresponding SDN account. After selecting the datacenter page, the user tried to delete the failed datacenter account and an error message appeared. | Sometimes, the UI shows an error message when deleting an account. This error message appears in the UI when certain criteria is met. | N/A |
| RIFT-26496 | Injection of SSH public key via Rift.ware isn't working. | RIFT.ware has the capability of injecting public SSH keys into the VDUs. There are two different mechanisms to add SSH keys. An operator can upload a SSH key to a Project | An operator will not have the capability to inject SSH keys throughout RIFT.ware. | SSH-keys can be injected via VDU cloud-init. |

| Support Tickets | Title | Description | Impact | Workaround |
|---|---|---|---|---|
| | | (Administration -> SSH Keys). On the NS Instantiation page, you can refer to this SSH key. This SSH key should be injected into the VDU. A user can also define s SSH key in a NSD and this key-pair should be injected into the VDU accordingly.<br>The above mechanisms are not working correctly. | | |
| RIFT-26498 | Creation of custom "User" in the VDUs via RIFT.ware isn't working. | In the NS instantiation page, Launchpad provides the option to create new users in the VDUs. This functionality isn’t working correctly. | N/A | N/A |
| RIFT-26499 | Disable take action button for terminate failed NS restart scenario. | A user created a VIM account, onboarded scaling descriptors and instantiated NS. Next, the operator edited /etc/resolv.conf with wrong data. In this scenario, the user tried to terminate the NS but it failed. There is an option to retry | N/A | N/A |
| RIFT-26503 | External cp creation disabled option makes NS fail. | A user created a VNFM account where ext cp creation is set no and instantiation failed. | It is not possible to successfully instantiate for a VNFM account with external cp creation disabled (set to no). | Make external cp enabled (set yes) in the VNFM account. |

| Support Tickets | Title | Description | Impact | Workaround |
|---|---|---|---|---|
| RIFT-26506 | Sol003: For failed-init NS terminate has no effect; calls rollback; delete is not clean. | If a user deletes a failed SOL003 NS, it is successful. When the user deletes the corresponding datacenter account, account deletion fails. This issue occurs only for services which use a VNFM account with external cp disabled (set to no). | Deleting a VIM account is blocked. User will have to keep it in the UI. | Make external cp enabled (set yes) in the VNFM account. |
| RIFT-26517 | VNFR should show cloud-init with substituted variables. | A VDU's cloud-init can have Custom Meta Data variables. These variables can be substituted with actual values when a NS is instantiated and finally the cloud-int (with substituted values) is pushed into the VM. This is the actual cloud-init inside a VM. The issue occurs after NS instantiation. In the VNFR, a user should see the cloud-init with the substituted variables. However, in the VNFR the cloud-init includes the previous variables and not the substituted values. | In Launchpad, an operator has no way of knowing whether the cloud-init (with substituted values) were actually injected into the VDUs. | Once the NS is instantiated, an operator can log into the VDUs and cross-check whether cloud-init was actually applied. |
| RIFT-26519 | Placement groups implementation | Currently, the support of placement groups is at the VNFD level. It | N/A | N/A |

| Support Tickets | Title | Description | Impact | Workaround |
|---|---|---|---|---|
| | do not include NS level. | is possible to configure it but there is no way to add it at the NSD level during instantiation. | | |
| RIFT-26532 | Network Services are running failed after the Stand alone upgrade of the LP (7.1 to 7.2). | An NS failed after upgrading the Launchpad. | N/A | N/A |
| RIFT-26537 | Datacenter edit option always populates field values from SDN account. | A user created a SDN OpenStack account and a datacenter account associated with an SDN account. The fields are auto-populated from the SDN account. The user edited the datacenter account details with different credentials. The user clicked on the edit option. The details populated in the text fields are not the same as the ones that the user added. The text fields are being populated from the SDN account. | The details populated in the text fields are not the same as the ones that the user added. | Clear the values populated from the SDN and your values. |
| RIFT-26559 | Inside datacenter discovery host-aggregate and availability-zone don’t list any info. | After discovery is done on datacenters, the UI shows the host-aggregates and availability-zones connected to these datacenters in the RESOURCE tab in the UI along with other resources like vim- | The data for host-aggregates and availability-zones is currently not populated in the UI. | Check for the data in the CLI. |

| Support Tickets | Title | Description | Impact | Workaround |
|---|---|---|---|---|
| | | quotas, interfaces, etc. The host-aggregates and availability-zones are only populated for datacenters with admin access. | | |
| RIFT-26571 | Placement groups: UI needs to provide a way override discovered details. | The placement group configuration of the Instantiation form depends on information the discovery process uncovers. This discovery process is checking the selected datacenters. In addition, the credentials provided in the datacenter account configuration must have admin privileges in order for the information about these placement options to be gathered. | If this information is not discovered, then the vnf deployment placement feature cannot be used. | Provide the appropriate credentials in the datacenter account configuration and run the discovery operation on that datacenter. |
| RIFT-26643 | Failure in Audit operation causes Failure in Account validation. | A failure in an audit operation causes a failure in account validation. Also, the audit feature is only workable in OpenStack Pike and above. | N/A | Click on Account Refresh button in case of failure. |