ProVision-vRA-Plugin

Current Plugin Release Version: 3.6.21

Overview

The IPAM Service provider consists of:

- Endpoint type definition which must be registered with vRealize Automation
- One or more endpoints that connect to a backend ProVision system
- Network profiles that are associated with endpoints

During a typical VM lifecycle, vRealize Automation calls the ProVision IPAM SDK API endpoints that must provide required data as a return value.

Requirements

6connect ProVision - requires ProVision 5.3 or above

VMware Compatibility - vRealize 7.1 or above

Functionality

Get Address Space

This workflow returns a list of resource entries (Section: Resource Holder, Category: Address Space) in the ProVision. Each tenant in the vRealize Automation can have one or more address space. The parent of all address spaces is a tenant root resource.

> Category *Address Space* must be added on the ProVision server.

Get IP Ranges

Returns all the IP ranges that that are assigned to the selected address space. These IP Ranges are then assigned to the network profile. Further differentiation within all the ranges is done by specifying the search tags, passed via the custom properties as follows:

Property name	Scope	Example
Custom.ProVision.SearchTags.Global	All tenants	GlobalTag:1234
Custom.ProVision.SearchTags.Tenant	Tenant	TenantTag:Tenant1
Custom.ProVision.SearchTags.BusinessGroup	Business Group	BusinessTag:BG-A
Custom.ProVision.SearchTags.BluePrint	BluePrint	NetworkType:VM-1
Custom.ProVision.SearchTags.Mode	BluePrint	OVERRIDE

Table 1 IP Range filtering tags

All of the properties above are optional and can be combined in two ways (specified by the Mode property):

- MERGE: non-null properties from all levels are merged together when performing AIP queries
- OVERRIDE: properties defined at the lower level (Global - Tenant - Business Group - Blueprint) override any value above them

The values of the properties can be static dynamic. For example there could be a drop down that would list meaningful names to the customers (Floor 1, Floor 2, etc) and this can be translated to ProVision tags via vRealize Action item (which can in turn get this information externally or from vRO configuration element).

Allocate IP address

Allocates one or more IP addresses based on the input properties and returns then to vRealize automation. IP range selection can be further narrowed down by using custom properties (previous Section).

Custom property *Scc.Vm.Orch.NetworknicIndex.EnableDHCP* determines whether the allocation is DHCP or Static. In case of *Static* allocation, the IP address is allocated from the IP ranges satisfying the conditions in custom properties. The allocation process therefore follows this process:

1. User requests virtual machine in the vRealize Automation
2. During resource allocation process, vRealize Automation determines which network profile should be used when searching for IP address
3. By using network profile, vRealize Automation, determines which IP ranges are available for allocation and passes the info (along with any custom properties) to the *Allocate* workflow.
4. *Allocate* workflow accepts the parameters from the vRealize Automation and calls the ProVision REST API.
   Allocated IP addresses are tagged with a NIC tag, indicating the index of the Network Interface Card that IP address is allocated to. For example by default, NIC index is prefixed with **NIC:** giving the tags **NIC:0,NIC:1, etc.** Tags are created automatically on the fly. The NIC prefix is defined in *ProVision Settings* configuration element field *NICTagPrefix.*

If allocation on ProVision system is successful, workflow returns the allocated IP addresses to vRealize Automation (along with the additional information, see Table 2.

1. If *Scc.Vm.Orch.NetworknicIndex.EnableDHCP* is true, the address, static binding is also created.
2. If the requested NIC index is that of the default interface (default:0) and the *DNSZoneId* is configured in tenant settings, the DNS A record is also pushed to this zone. For the value of DNS A record, the vmName is used.
3. vRealize Automation configures returned IP addresses and makes configuration changes on the VM

Property	**Description**
SubnetPrefixLength	Obtained from IP range
Gateway	Obtained from IP range, /32 address tagged as gateway (Tag name configurable)
PrimaryDNS	Provided by custom properties
SecondaryDNS	

Table 2 Additional allocation information

Property name	**Scope**	**Example**
VirtualMachine.Network.Network#.DNSSuffix	BluePrint	demo.local
VirtualMachine.Network.Network#.PrimaryDNS	BluePrint	10.200.1.11
VirtualMachine.Network.Network#.SecondaryDNS	BluePrint	10.200.1.12
VirtualMachine.Network.Network#.PrimaryWINS	BluePrint	10.1.3.1
VirtualMachine.Network.Network#.SecondaryWINS	BluePrint	10.1.3.2
VirtualMachine.Network.Network#.DNSSearchSuffixes	BluePrint	demo.local, demo2.local
VirtualMachine.Network.Network#. dhcpStaticRoutes121	BluePrint	[{“key”:”10.1.0.0”,”value”: “10.1.0.1”},…]
VirtualMachine.Network.Network#. interfaceMTU26	BluePrint	1500
VirtualMachine.Network.Network#. startDate67	BluePrint	1/1/1900
VirtualMachine.Network.Network#. tftpServerName66	BluePrint	Tftp1
VirtualMachine.Network.Network#. tftpServerAddresses150	BluePrint	[“10.1.1.1”,,”10.1.1.2”]
VirtualMachine.Network.Network#.nextServer	BluePrint	1.1.2.3
VirtualMachine.Network.Network#.autconfigureDNS	BluePrint	true or false
VirtualMachine.Network.Network#.leaseTime	BluePrint	86400

Custom.ProVision.SkipHolding	BluePrint	true

Release IP address

Workflow *Release* is called when vRealize Automation deprovisions the virtual machine. Workflow accepts the IP address that needs to be released and returns it to the free IP Range. Property *Custom.ProVision.SkipHolding* specifies whether IP address should be released immediately (value 'true') or put in a holding tank (undefined/empty or 'false').

If the requested NIC index is that of the default interface (default:0) and the *DNSZoneId* is configured in tenant settings, the corresponding DNS A record is also removed from this zone.

Installation

Please contact 6connect and you will then be able to download the plugin. Once downloaded, import the plugin into vRealize Orchestrator.

1.1. Create REST Host for ProVision in vRealize Orchestrator

First, the API endpoint for the ProVision server needs to be created in vRealize Orchestrator. This is performed by executing the *Add a REST Host* workflows in the *Library/HTTP-REST/Configuration* folder.

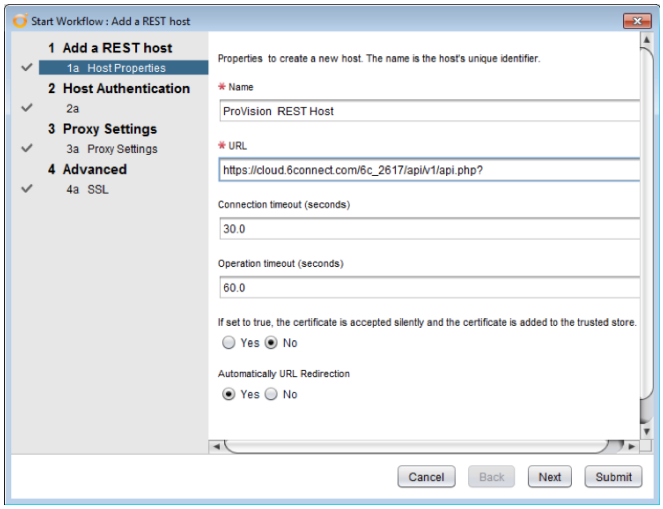

Figure 1 Add a REST Host

In the first step fill in the name and the URL of the API endpoint. URL can be one of the following:

- Cloud instance: https://cloud.6connect.com/tenantId/api/v1/api.php?
- Local instance: https://provision.example.com/api/v1/api.php?

1.2. Register ProVision IPAM Endpoint type

In order to be able to create ProVision endpoints, the IPAM Endpoint type needs to be registered in vRrealize Automation. This is performed by running the *Add/Update 6Connect IPAM Endpoint* workflow located in the *6Connect/IPAM Service Provider/Configuration* folder.

This operation essentially connects the vRA actions described in paragraph Provision IPAM Service Provider with the corresponding workflows in the vRO.

In the vRA step of the workflow, the vRA server and credentials need to be entered as shown in Figure 2. Credentials entered must have sufficient permissions on the vRA server. This should normally be *administrator@vsphere.local* if default values were used at installation (SSO Default tenant password).

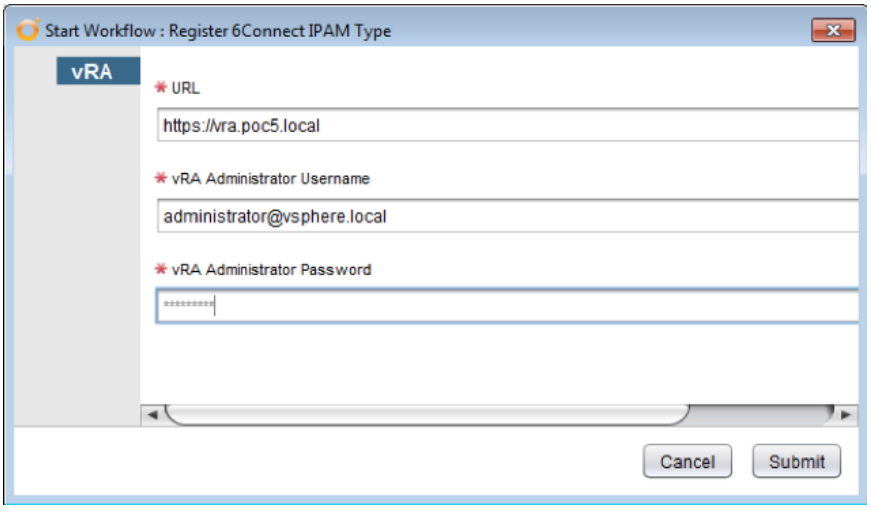

Figure 2 Register 6Connect IPAM Type

After this is done, new category will appear in the vRA when creating endpoints.

1.3. Create endpoint

ProVision endpoint can also be created from vRealize Automation web interface, but since custom property is used to specify REST host, it is simpler to use the workflow *Add/Update 6Connect IPAM Endpoint*.

In the *Common parameters* step, *tenantId* parameter needs to be entered. This must be the same value as tenant URL value of the corresponding tenant in vRA.

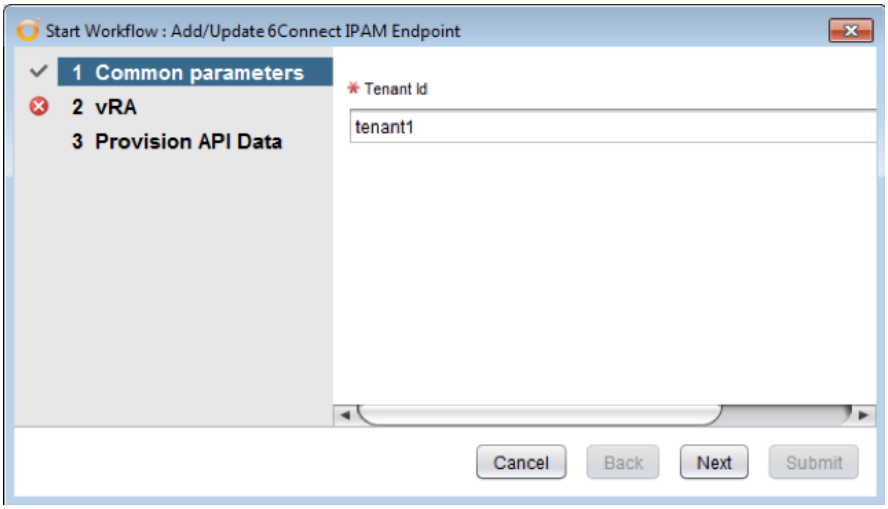

Figure 3 Tenant for IPAM endpoint

In the *vRA* step of *Add/Update 6Connect IPAM Endpoint* workflow, same information needs to be entered as in Figure 2.

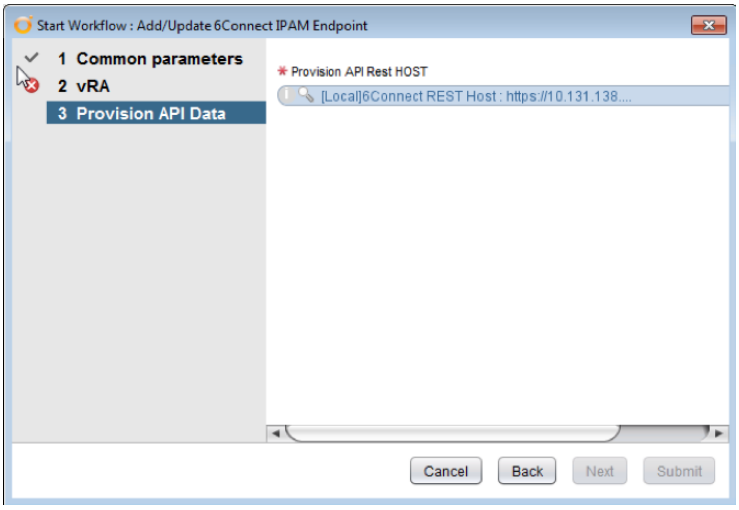

Figure 4 Adding IPAM Endpoint: Provision API data

In the third step (Figure 3), ProVision API data must be provided:

- ProVision API Rest HOST: select the REST Host created in 4.1.

Endpoint name is automatically generated in the followin form: *Provision:tenantId*

After successfully execution, the ProVision endpoint is created in vRealize Automation and with the following custom property:

- Custom.ProVision.RestHOSTId: value contains the GUID of the ProVision REST Host that was selected. This tells the workflows on which ProVision server to execute API calls
- Custom.ProVision.TenantId: contains the tenant to which the endpoint belongs.

1.4. vRO Configuration (ProVision)

vRO configuration is stored in the cofiguration element object which can be accessed by selecting *Configurations* tab in the vRealize Orchestrator client. Configuration element is located in *Settings/ProVision Settings.*

Attribute name	Type	Default value	Description
ApiKey	String		Value ApiKey for the ProVision API
ApiSecret	String		Value of ApiSecret for the ProVision API
restHost	RESTHost		REST host for accessing the ProVision API (default)
skipHolding	Boolean	True	Global skip holding setting when releasing the IP address
Credentials	Array		Array of [RestHost, ApiKey, ApiSecret] for multi ProVision environments
AddressSpacesCategorySlug	String	customer	Resource entries category for Address Spaces in the vRealize Automation
AddressSpaceSectionSlug	String	resource-holder	Resource entries section for Address Spaces in the vRealize Automation
VirtualMachineSectionSlug	String	virtual-machine	Section in which the virtual machine resource is created

GatewayTag	String	Gateway	Tag that indicates a gateway IP address
NICTagPrefix	String	NIC:	Tag Prefix for NIC that block is assigned to (NIC Index is appended to this)
tenantSettingsPath	String	Settings /TenantSettings	Location where to store tenant settings
AggregateCategorySlug	String	aggregate	Aggregate category slug
AddressSpaceReservationTag	String	VRA:AS-Reserved	Tag to mark the address space IP blocks
AddressSpaceCategoryName	String	Address Space	The name of the address space category
AggregateCategoryName	String	Aggregate	The name of the aggregate category
TenantRootCategorySlug	String	customer	Tenant root resource category
TenantRootSectionSlug	String	resource-holder	Section of tenant root
TenantRootResourceType	String	entry	Resource type for tenant root
AggregateSectionSlug	String	resource-holder	Aggregate section
AggregateResourceType	String	entry	Aggregate resource type
nsxConfigurationElementPath	String	Settings	Path to the nsx configuration element
nsxConfigurationElementName	String	nsx-configuration-default	Name of the nsx configuration element
dhcpEnablerPropertyName	String	Scc.Vm.Orch.Network	VM property prefix for DHCP enabled property
MaxRetries	Number	3	Maximum number of retries of failed API calls
retrySleepTime	Number	3	Sleep time between API call retries

Section 1.0 ProVision Setup

The process of setting up the tenant, tenant aggregates, address spaces and ip blocks assignment is fully automated through the workflows described below.

1.1. Initial ProVision Setup

Before new tenants can be provisioned in ProVision, basic configuration workflows need to be executed once per ProVision server and vRO environment.

1.1.1.Tagging the ProVision settings element

In order to simplify searching for the ProVision settings configuration element, the following global tags are added to it:

- CONFIGURATION-SYSTEM:PROVISION
- CONFIGURATION-TYPE:GLOBAL

Execute workflow ***OrchTagConfiguration*** once per vRO server when performing upgrade existing plugin installation. The workflow will do nothing if tags are already present.

1.1.2.Creating required ProVision objects

In order for multitenancy to function properly, several object need to exist before tenants can be onboarded. This includes:

- Aggregate resource category
- Address space resource category
- Gateway tag

Execute workflow ***OrchProVisionBaseConfiguration*** once per ProVision server. If required object are already present, the workflow will do nothing.

1.2. Create New Tenant

Workflow ***OrchCreateNewTenant*** creates the following objects on ProVision:

- TenantRoot resource named tenantId
- TenantAggregates resource named tenantId-aggregates

Workflow checks if objects above already exist before creating new ones. Workflow creates a configuration element under *Settings /TenantSettings: TenantSettings-tenantId* and adds the following attributes to it:

- TenantId
- RestHostId: RestHostId of the ProVision server for tenant
- TenantRootResourceId: Resource id of the TenantRoot resource on ProVision server
- TenantAggregateResourceId: Resource id of the TenantAggregates resource on the ProVision server
- EndpointId: endpoint name that is created for this tenant
- DNSZoneId: DNS zone id (internal Id of ProVision DNS zone – visible in URL as zoneId query parameter)

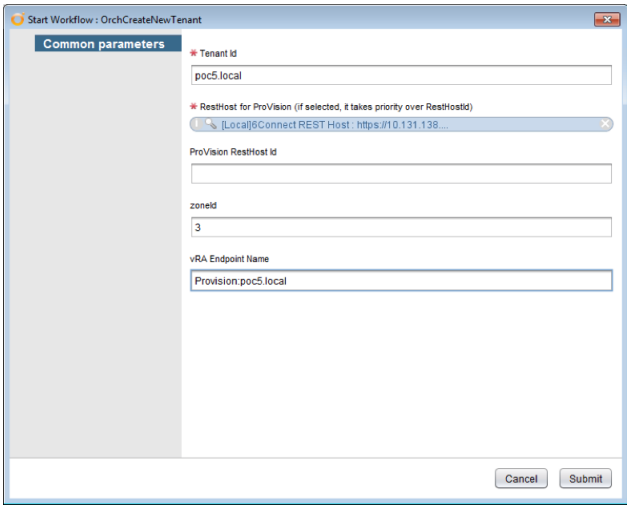

Figure 5 Create New Tenant input form

The ***OrchCreateNewTenant*** can be repeatedly executed with new input values. If the tenant with the same id already exists, it will update the values for *RestHostId*, *DNSZoneId.* Any ProVision object that already exists will be left intact.

1.3. Add Aggregate To Tenant

Workflow *OrchAddAggregateToTenant* adds new aggregate to a tenant.

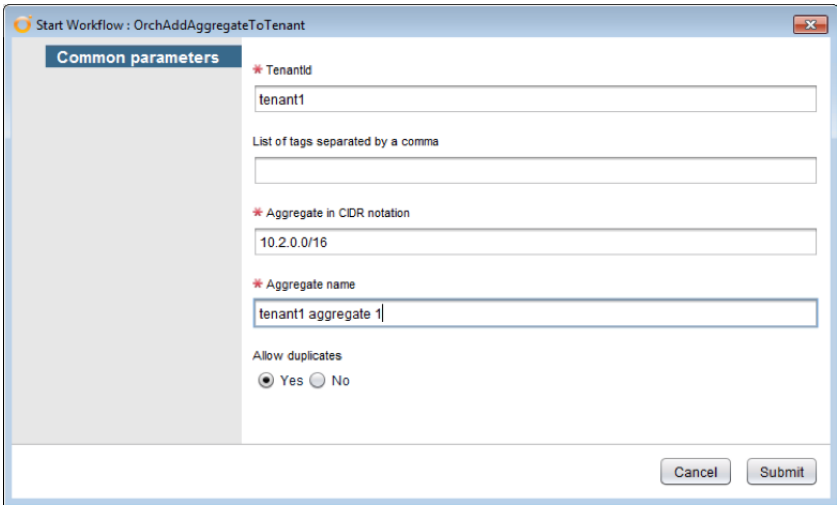

Figure 6 Add Aggregate To Tenant

Worflows performs the following:

- Creates specified aggregate block.
- Assigns it to *TenantAggregate* resource.
- If allow duplicates is set to Yes, it creates aggregate even if the same block already exists (same or other tenant)
- If allow duplicates is set to No, it will skip creation if duplicate is detected in same tenant, otherwise it will report error.

1.4. Create Address Space And Assign Block

Workflow ***OrchCreateAddressSpaceAndAssignBlock*** creates new address space resource and then assigns specified block to it. If the address space with the same name already exists, new address block is added to it. This workflow is a composite of the workflows ***OrchCreateAddressSpace*** and ***OrchAssignBlockToAddressSpace*** which are called in a sequence. They can also be called separately, depending on requirements.

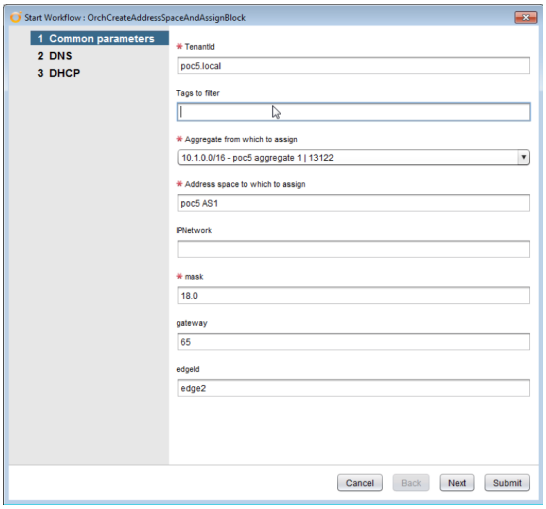

Figure 7 OrchCreateAddressSpaceAndAssignBlock step 1

In step 1 of the ***OrchCreateAddressSpaceAndAssignBlock*****,** network parameters are specified:

- tenantId
- Tags to filter: a list of tags by which to filter aggregates dropdown
- Aggregates from which to assign
- IPNetwork: IP network to assign
  IP network can be empty, in which case the new network is allocated from a pool of available blocks of a given size (mask)
- Mask: size of the network
- Gateway: Default getway for a given block.
  Gateway can be specified as
  - IP address literal (e.g. 10.2.1.1) – only if IPNetwork parameter is specified
  - IP address index: Ordinal number of the ip address in a give block size (e.g. IPNetwork = 10.1.1.0/24 and gateway = 65 gives gateway address of 10.1.1.65)
  - edgeId: Edge id for DHCP reservation (NSX DLR)

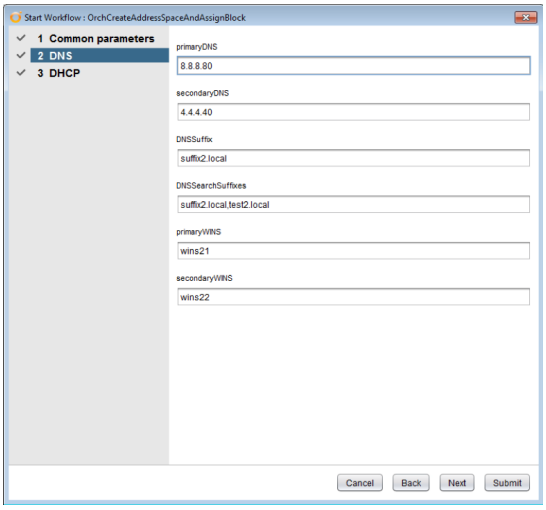

Figure 8 OrchCreateAddressSpaceAndAssignBlock step 2

In step 2, DNS parameters are specified. These values are stored in the allocated IP block and are returned to vRA when *Allocate* is requested. These values can be overridden for each vNIC separately by values of the following properties:

- VirtualMachine.Network.Network#.DNSSuffix
- VirtualMachine.Network.Network#.PrimaryDNS
- VirtualMachine.Network.Network#.SecondaryDNS
- VirtualMachine.Network.Network#.PrimaryWINS
- VirtualMachine.Network.Network#.SecondaryWINS
- VirtualMachine.Network.Network#.DNSSearchSuffixes

If any of these values are not present, the value from IP block is used as a default.

In step 3, additional DHCP options can be specified in form of key-value pairs.

1.5. Delete Address Space

Workflow ***OrchDeleteAddressSpace*** removes address space and releases all of its blocks back to aggregate.

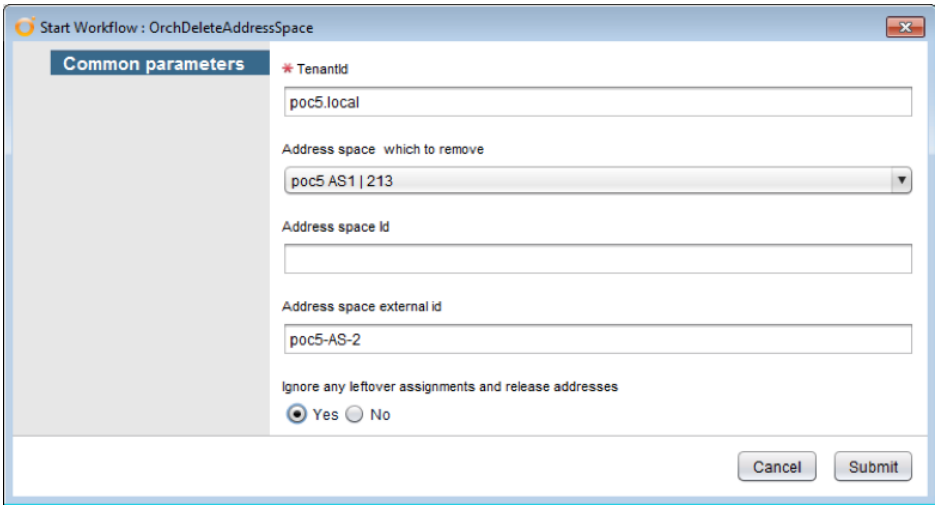

Figure 9 Delete Address Space

- tenantId
- Address Space which to remove: drop down list displays all the address spaces of the specified tenant
- Address Space Id: Address space resource id can also be specified if running in none interactive mode
- Address Space Slug: Address space alphanumeric Id. This is an Id of ***addressSpaceExternalId*** property of the address space on the vRA networkProfile.
- Ignore any leftover assignments and release addresses: If set to ***Yes***, any assigned IP addresses are unassigned before removing address space. If set to No and there are still IP addresses assigned (apart from network address, broadcast address and gateway), the workflow throws an error

When IP blocks are removed, the reaggregation is performed up to maximum blocksize possible.

1.6. Remove IP Block From Address Space

Workflow ***OrchRemoveIPBlockFromAddressSpace*** unassigns a single IP block from address space and releases it back to parent aggregate. Reaggregation is performed up to a maximum possible block size.

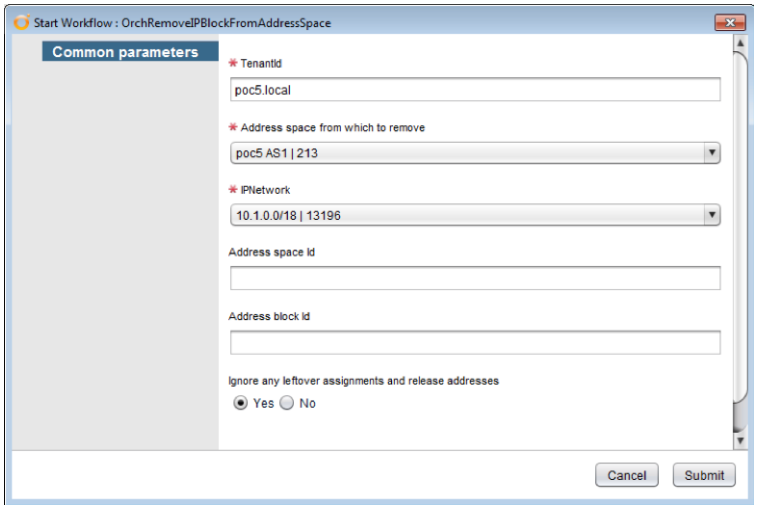

Figure 10 Remove IP Block From Address Space

- tenantId
- Address space from which to remove: A drop down list displays all of tenant address blocks
- IPNetwork: A drop down list displays all IP blocks that are assigned to selected address space
- AddressSpaceId: address space resource id (for non-interactive execution)
- Address BlockId: address block id (for non-interactive execution)
- Ignore any leftover assignments and release addresses: If set to ***Yes***, address block is aggregated regardless of assignments. If set to No and there are still IP addresses assigned (apart from network address, broadcast address and gateway), the workflow throws an error

1.7. Remove Aggregate From Tenant

Workflow ***OrchRemoveAggregateFromTenant*** removes an aggregate from a tenant.

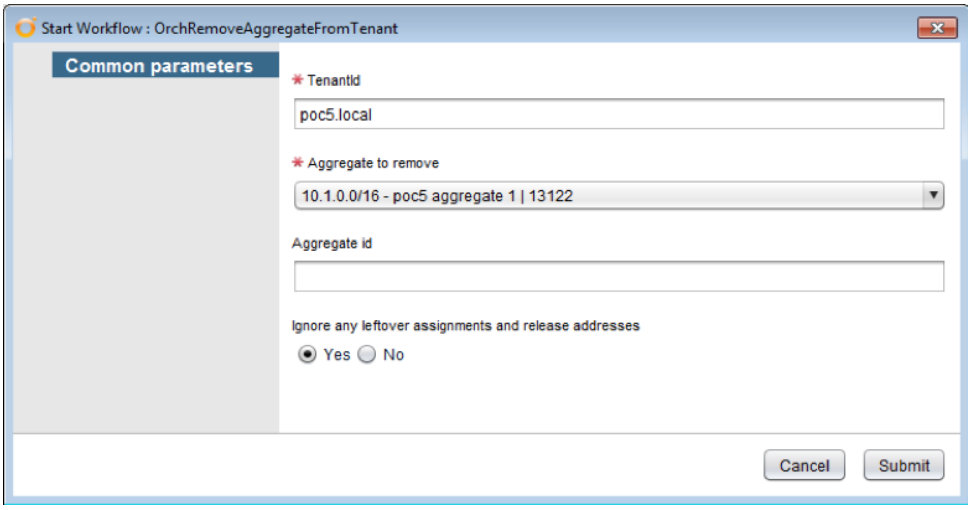

Figure 11 Remove Aggregate From Tenant

- tenantId
- Aggregate to remove: a drop down list display a list of tenant aggregates
- Aggregate Id: aggregate block id (for non-interactive execution)
- Ignore any leftover assignments and release addresses: If set to ***Yes***, aggregate is deleted regardless of assignments. If set to No and there are still IP blocks assigned the workflow throws an error

1.8. Delete Tenant

Workflow ***OrchDeleteTenant*** removes tenant resources from the ProVision server.

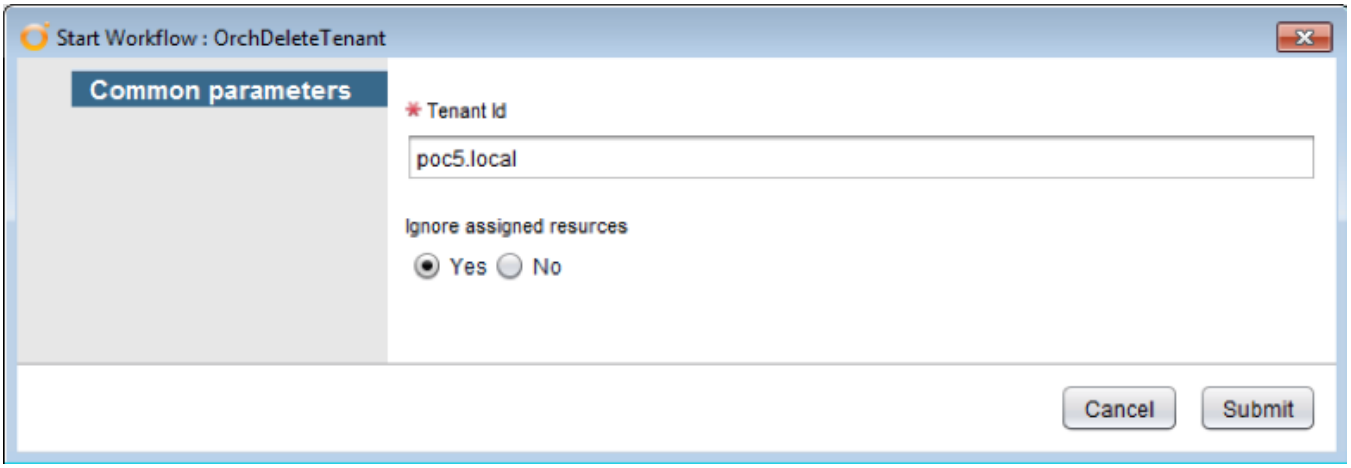

Figure 12 Delete Tenant

- tenantId
- Ignore assigned resources: If set to ***Yes***, tenant is deleted regardless of assignments. If set to No and there are still IP blocks assigned the workflow throws an error.

Workflow tags a configuration element under *Settings/TenantSettings: TenantSettings-tenantId* with new tag value: CONFIGURATION-STATUS: DELETED

Section 2.0 vRealize Automation Setup

This section summarizes which custom properties and property groups need to be created in vRealize Automation. In order to simplify creation of properties, they are provided in a form of *.zip* files that can be imported with the help of cloud client application (https://code.vmware.com/tool /cloudclient/4.1.0)

Cloud client is a java CLI application that is started with the following command :

```
cloudclient.bat (or cloudclient.sh if running on Linux)
```

In order to login to the vRA server execute the following command from cloud client CLI:

```
vra login userpass --tenant <tenant>
```

To import the .zip file, execute the following:

```
vra content import --path <path to zip file> --resolution OVERWRITE --precheck ON
```

2.1. Custom properties

Properties are optional component that needs to be created if more granular selection of IP ranges is needed within one address space (base on the Tenant, BusinessGroup). In the simple scenario, where one address space contains one IP range they are normally not needed. In order for properties to take place, they need to be attached to the blueprint.

The following properties need to be created in *Administration - Property dictionary - Property definitions:*

- Custom.ProVision.SearchTags.Global
- Custom.ProVision.SearchTags.Tenant
- Custom.ProVision.SearchTags.BluePrint
- Custom.ProVision.SearchTags.Mode
- Custom.ProVision.SearchTags.Tenant
- Custom.ProVision.SkipHolding

2.2. Property groups

The following properties need to be created in *Administration - Property dictionary - Property groups:*

- CustomProVisionTenantPropertyGroup

2.3. Business group properties

The following property must be created on any business group within tenant: *Administration - Users&Groups - Business Groups:*

- Custom.ProVision.SearchTags.BusinessGroup

2.4. DHCP Subscriptions

DCHP subscription workflows are used to push the DHCP information on NSX during the VM provisioning phase. There are two subscription workflows for this purpose:

- OrchDHCPSubscriptionAllocate
  - Triggered in phase ‘EVENT’ and Life Cycle Event: ‘CloneWorkflow.CloneMachine.EVENT.OnCloneMachineComplete’ when MAC address becomes known
  - Pushes the IP address and DHCP options for each NIC that is DHCP enabled
  - DHCP enabled NIC must have property ‘Scc.Vm.Orch.Network#.EnableDHCP’ set to ‘true’
  - OrchDHCPSubscriptionRelease
    - Triggered in phase ‘PRE’, state ‘VMPSMasterWorkflow32.Disposing’
    - Removes the DHCP binding for all DHCP enabled NICs

DHCP Subscriptions need to be imported for each tenant separately. Run the *6Connect/Operations/OrchDHCPSubscriptionImporter* workflow and provide a *tenantId* as a parameter.

The workflow will automatically read the subscriptions stored in *6Connect-subscriptions/dhcp-subscriptions* resource element and import them into vRA.