Netrino UI Documentation

Release 0.0.0

Dave Kruger

Nov 28, 2017

Contents

1 Contents **3**
1.1 Netrino UI User Guide 3
1.2 Netrino UI Development Guide 8
1.3 Frequently asked Questions 8

Release v0.0

Netrino is an open source Network Orchastration tool. It acts as the middleware between your network and BSS/OSS.

It has a RESTfull API through which you can program your network, as well as a WEB UI to start making use of it immediately, without any additional development.

Netrino is a module of the Tachyonic Project and its neutrino Framework, after which it is named.

These documents describe the UI portion. For the API portion refer to netrino-api.readthedocs.io.

CHAPTER 1

Contents

1.1 Netrino UI User Guide

1.1.1 Users

Creating users follows the same procedure as with Tachyon UI.

With Netrino, the username has to match the login that the user makes use of to log into the network device. For example, if the network devices makes use of TACACS+, and a user uses TACACS+ username `bob`, then the Netrino username must also be `bob`.

At this time only SSH is supported, and only key authentication. All the user private SSH keys need to exist on the server. The location of this directory must be listed in the netrino-celery.cfg file under the section `[locations]`:

```
[locations]
ssh_keys = /some/secure/location
```

Each key has to be in the format:

username.key

where *username* is the user's login name.

Note that this directory and these key files have to be readable by the user that runs the neutrino WSGI process. Be sure to have strict file permissions.

This means that whatever user creates the Netrino service request(s), will also be the user that logs into the network device. This way you do not require any additional RBAC methods in order to use Netrino - existing RBAC will be used.

1.1.2 Devices

Adding Devices

In order to add a new device to the system, navigate to

`Infrastructure -> Network -> Devices`

and click on Create.

Complete the two fields: device IP address and SNMP community string.

Once you hit the `Create` button, the system will generate a new *service request* for this discovery.

SNMP v2 is used, and only to detect the device hostname and operating system.

Thereafter the system will log into the device to obtain interface and IP address information. In order to do that be sure to have the *user SSH keys* configured.

For this action, a *Service Request* is created. It will have no customer or service attached, but you may observe the status and result at the *Service Requests* page.

Note: One may supply a subnet in order to provide a device range. In that case, each and every IP address in the subnet will be queried, and a service request will be created for each IP address.

Devices currently supported:

- Cisco IOS
- Junos

Need more? Let us know by loggin an issue on github.

Viewing Devices

Simply Navigate to `Infrastructure -> Network -> Devices`. Here you can see a list of all devices. The table is searchable (all fields) and sortable. To view the specific port, service and customer information for a specific device, click on the magnify glass to the right of the device. This takes you to the view device page, which will also show you when last the device port and ip address information was updated. Because Netrino knows about the customers and services (that it provisioned) attached to an interface, this information will be up to date automatically. Once again, this table is searchable and sortable.

Updating Devices

From time to time it might be required to update the port information of a device. To do this, navigate to `Infrastructure -> Network -> Devices`, click on the `view device` icon, and hit the `edit` button. Here you are presented with the same form as when the device was discovered initially. Simply hit the `save` button to initiate a new device discovery service request.

Deleting Devices

In order to delete a device, navigate to `Infrastructure -> Network -> Devices`, click on the `view device` icon, and hit the `edit` button. Then click on the `delete` button, you will be prompted with a confirmation dialog. If there are any service requests ACTIVE on the device, it will list the number of active services. To view them, navigate to the *Service Requests* page, and enter the IP address for the device (in decimal form, the dialog will provide it).

1.1.3 Interface Groups

Interface Groups is a mechanism to prevent certain users from configuring certain device ports.

It thus allow users to attach a *Service Request* only to a device port for which he or she is authorized to do so.

For example, one can have a `backbone` interface group, which only users with the `Core Engineer` role is allowed to provision on.

When a device is newly discovered, the device ports belong to no interface group. This means by default no port is configurable. This is done intentionally to safeguard against unauthorized network configuration.

Creating Interface Groups

Navigate to `Infrastructure -> Network -> Interface Groups` and hit the `Create` button.

Simply enter the name of the Interface Group, and hit the `Save` button.

Viewing Interface Groups

Navigate to `Infrastructure -> Network -> Interface Groups`. Here a list of all configured Interface Groups are displayed.

Edit Interface Group Names

Navigate to `Infrastructure -> Network -> Interface Groups` and click on the view icon next to the Interface Group to be renamed. Hit the `Edit` button, provide the new name, and hit `Save`.

Deleting Interface Groups

Navigate to `Infrastructure -> Network -> Interface Groups` and click on the view icon next to the Interface Group to be deleted. Hit the `Edit` button, and then the `Delete` button.

Assign Device ports to an Interface Group

After a device has been *discovered*, and at least one Interface group has been *created* , navigate to `Infrastructure -> Network -> Devices` and click on the view icon next to the Device. Hit the `Add Interface Groups` button below the list of ports.

Now a checkbox appears to the right of each port, and a drop-down list below the table. Select one or more ports by checking the checkboxes, as well as the Interface Group from the drop-down list, and hit the `Save` button.

1.1.4 Services

The Services section provides the device configuration templates that define services. Each service has to be associated to an *Interface Group*, as well as a User Role.

This way one can ensure that services are only applied to interfaces for which they are allowed, and only by users that are allowed to provision them.

Templating Engine

The Jinja2 templating engine is used to expand the text entered in the configuration sections. Any variable referenced with `{{ .. }}`, will result in an additional text field in the *Create Service Request* form.

For example, this template:

```
interface FastEthernet0/0
  description "{{ description }}"
  ip address {{ ipv4 }} {{ mask }}
```

Will result in three additional required fields to be completed when creating a service request with this Service: `description`, `ipv4` and `mask`

Load merge technique will be used to deploy the configuration.

Special Variables

Ports/Interfaces:

The `{{ interface }}` variable will not result in a text field, but a drop-down list instead. This list will be populated with the most recent ports discovered on the selected device.

Creating a Service

After at least one *Interface Group* has been created, navigate to `Infrastructure -> Network -> Services`, and hit the `Create` button.

Complete the fields for the `Service Name`, `Interface Group` and `User Role`.

Next, there are three paragraph fields. Only the first one, `Config`, is compulsary.

1. Config

This is the configuration that will be deployed when the Service Request is *created*.

2. Activation config

This is the configuration that will be deployed when the Service Request is *activated*. This may be used for example to enable a port once it has been confirmed that payment for the service was received.

If left blank, Service Requests can be activated without touching the device.

3. Deactivation config

Use this part of the Service template to provide the decomissioning configuration. It will be deployed when the Service Request is *deactivated*.

If left blank, Service Requests can be deactivated without touching the device.

1.1.5 Service Requests

The purpose of Netrino is to create Service Requests.

A Service Request is the association of these 3 items:

- A Tenant/Customer,
- to a Service,
- to a Network Device and/or port.

Creating a Service Request

1. Select the Tenant/Customer.

After at least one Tenant has been provisioned to the system, enter the name of the Tenant at the `Tenant Name` Search bar at the top right. Select the name from the auto drop-down, or hit search. From the pane below, hit the `Open` button for the applicable Tenant.

2. Select the Service

Aftter at least one Service have been created, navigate to `Infrastructure -> Network -> Service Requests`, and hit the `Create` button. From the first Drop Down list, select the applicable Service. Once selected, more fields may appear in the form if there were additional fields associated with the Service.

3. Select the Device

After at least one Device has been *discovered*, from the second drop-down list, select the applicable Device to which th service request is to be deployed. It is possible to select more than one device, but at this time no interface/port configuration is supported in such a case.

4. Select the port (optional)

If the Service contained the special variable `{{interface}}`, a drop-down list will appear listing all the ports on the device that:

- belong to the same Interface Group as the Service, and
- do not have any ACTIVE service requests currently attached to them.

Note: Only the ports discovered in the last device discovery attempt will be listed; it might be neccesary to *rediscover* the device to show an up-to-date list

---

Select the applicable port, and hit the create button.

The system will initiate a deployment task in the background, and redirect to the *Service Requests* Page. The status will probably be PENDING at this stage, as it takes a short while to log in and configure the device.

Viewing Service Requests

Simply navigate to `Infrastructure -> Network -> Service Requests` to view a list of all service requests. Note that if a Tenant is selected, only service requests for that Tenant will be displayed. Otherwise service requests for all Tenants are displayed, as well as device discovery service requests. To view a specific Service Request, hit the view icon next to it. This will display the log and status of the service request.

Activating Services

If the current status of the Service Request is SUCCESS, on the *View Service Request* page, there will be an `Activate` Button. Hit this button to activate the service.

If the Activation config of the service is empty, the Status will change to ACTIVE immediately. Otherwise, the status will change to PENDING as a new task is loaded in the background to configure the device.

Deactivating Services

If the current status of the Service Request is ACTIVE, on the *View Service Request* page, there will be a `Deactivate` button. Hit this button to deactivate the service.

If the Deactivation config of the service is empty, the Status will change to INACTIVE immediately. Otherwise, the status will change to PENDING as a new task is loaded in the background to configure the device.

1.2 Netrino UI Development Guide

This Guide describes some of the concepts used in the development of the UI for Netrino.

1.2.1 Introduction

The Netrino UI follows the same development concepts used for Tachyonic UI.

It obtains all the data via API calls to the Netrino API, and as such does not have to reside on the same server.

Each section is referenced in its own view, which makes use of functions that reside inside the `controllers.py` file.

All the tables are rendered with the datatables jquery plugin. The select2 jquery plugin is used for dropdown lists.

Some of the views have special options available. These are described here.

Devices

The `/infrastructure/network/devices` route takes an optional `X-Format` HTTP Header:

- select2: returns a JSON object with the devices and their id's listed in select2 format. That is, a list of dictionaries with keys `id` and `text`.

The `/infrastructure/network/device/{id}/ports` route returns a JSON object containing all the port data for the device.

Options:

- An optional POST object `igroup`. If this value is supplied, the result will only include ports belonging to the specified igroup ID.
- an optional `X-Format` HTTP Header:select2: returns a JSON object with the devices and their id's listed in select2 format. That is, a list of dictionaries with keys `id` and `text`.

Services

The `/infrastructure/network/services` route takes an optional `X-Format` HTTP Header:

- fields: returns only the variables mentioned in the Service templates, as a JSON array
- fields+igroup: returns a JSON object with two keys: the interface group text name, and the fields array.

1.3 Frequently asked Questions

Q. I have no coding skills. Can I still use Netrino?

A. This is the reason why the Netrino UI exists - no coding is required. The only "coding" that might required is to make use of variables in the templates. Jinja offers more flexibility though, so if you have a rudimentary understanding of programming then you can get more creative with your templates.

Of course, if you have python knowledge, you may adapt the system to suit your needs. If you require something and don't have python abilities, you are welcome to get in touch *with the team to request Professional Services.*

Q. I want to deploy a Service that does not belong to a Customer, such as an Infrastructure service. Is this possible?

A. Of course. Although every service request requires a Tenant to be selected, one may simply create an Internal (not billed) Tenant, and use this for infrastructure deployments.