Kea DHCP and NetBox

Integrating with JSON/REST API

Carsten Strotmann, Michael Schwartzkopff and the ISC Team
Welcome

Welcome to our Webinar on integrating external tools with Kea DHCP (using NetBox as the example)
In this Webinar

- What is NetBox
- A quick tour of NetBox
- The NetBox API (REST/JSON/Python)
- Managing Kea DHCP Settings via NetBox custom fields
- Kea DHCP Subnet definitions, DHCP pools, DHCP reservations and options from within NetBox
- Exporting NetBox data into Kea DHCP
- Importing Kea DHCP data into NetBox
Goal of this Webinar

- This webinar will show how an external application can be integrated with Kea DHCP via the provided API
- NetBox is used as an example, the same solutions discussed here can be applied to other DCIM or IPAM solutions (Open Source or closed source)
  - It should also be applicable to other types of applications (Ticket systems, Inventory databases, Configuration orchestration tools, cloud management solutions)
Code used in this webinar

- The code created for this webinar is available under a permissive open source license (Apache License)
- The code is meant as an example, it is not supported by ISC or sys4 and it might not be suitable for production environments
What is NetBox
What is NetBox

• NetBox is an Open Source Datacenter Inventory System and IP Address Management System
  ■ IP Subnets and IP-Address Ranges
  ■ VLANs
  ■ Devices and their IP-Addresses
  ■ Virtual Machines
  ■ Cabling and Power

• Website: https://netbox.dev
• Source: https://github.com/netbox-community/netbox
What is NetBox

- NetBox is open source under the Apache 2 license
- It is build using Python (3.8+) and the Django Web Framework
- Other components:
  - PostgreSQL database (11+)
  - Redis (Key-Value Store, Version 4.0+)
  - HTTP Server (reverse Proxy with NGINX, Apache etc)
  - LDAP Environment (optional)
A quick tour of NetBox

Demo Time
The NetBox API (REST/JSON/Python)
The NetBox API

- NetBox offers different ways to integrate with external applications
  - REST/JSON API (similar to Kea DHCP API - this is what we will use in this webinar)
  - GraphQL
  - Webhooks - triggers where NetBox will be able to inform external applications whenever an object in NetBox changes
  - NAPALM (https://github.com/napalm-automation/napalm): a Python library that implements a set of functions to interact with different router vendor devices using a unified API
  - Prometheus Monitoring Metrics
Python Wrapper for NetBox

- `pynetbox (https://github.com/netbox-community/pynetbox)` is a Python wrapper around the NetBox API
- This python module eases the development of integration of external tools into NetBox
- As Python offers rich support for working with JSON based REST API, we have used the Python as the implementation language for our examples in this webinar
  - The same functions can be implemented in C/C++, Rust, Go, Ruby or other languages with support for JSON REST API
Python Wrapper for NetBox Example

- Reading the subnet (aka "Prefix") definitions from NetBox

```python
from pprint import pprint
import pynetbox

nb = pynetbox.api(
    'http://netbox.dane.onl',
    token='<secure-token>'
)

subnets = nb.ipam.prefixes.all()
for subnet in subnets:
    pprint(dict(subnet))
```
Python Wrapper for Kea DHCP

- pykeadhcp - A python module used to interact the the Kea DHCP API daemons (dhcp4, dhcp6, ctrl-agent and ddns) https://github.com/BSpendlove/pykeadhcp
Using the Python wrapper

- Changing the Kea DHCP4 configuration from Python

```python
from pykeadhcp import Kea

# Connect to the Kea Agent endpoint
server = Kea(host="http://[::1]", port=9099)

# Fetch the DHCP4 Kea configuration
config = server.dhcp4.config_get()

# Print a configuration value
print("Kea DHCP setting 'authoritative':", config['arguments']["Dhcp4"]['authoritative'])

# Change a configuration value
config['arguments']["Dhcp4"]['authoritative'] = True

# Sending the new configuration to Kea DHCP4
response = server.dhcp4.config_set(config["arguments"])
assert response["result"] == 0

# Write the new configuration back to the config file
response = server.dhcp4.config_write("/etc/kea/kea-dhcp4.conf")
assert response["result"] == 0
```
Documentations

• Documentation of the NetBox REST API: https://docs.netbox.dev/en/stable/integrations/rest-api/
• Documentation of the Kea DHCP REST API: https://kea.readthedocs.io/en/latest/arm/ctrl-channel.html
Managing Kea DHCP Settings via NetBox custom fields
NetBox custom fields

- NetBox currently does not naively support the storage of data for DHCP
- The user of a NetBox system can extend the NetBox data model with "custom fields"
- The custom fields can be attached to almost any of the NetBox native object types (Subnet, IP-Ranges, IP-Addresses etc)
- The NetBox custom fields offer a wide range of pre-defined data types (IP-Addresses, Text-Fields, Booleans [Yes/No or On/Off], Numbers, Date etc)
NetBox custom fields

- We will create and use the custom field feature to store DHCP specific data (e.g. DHCP options, Lease-Times, Kea DHCP configuration) in NetBox
- The NetBox custom fields are available through the REST API and will be used to convert the data stored into a valid Kea DHCP configuration
Creating custom fields for DHCP options in NetBox
Creating custom fields for DHCP dynamic pools in NetBox

```
Custom Field

Content types: [IPAM > Prefix]

Name: dhcp_pool
Label: DHCP Pool
Group name: kea-dhcp
Type: Text
Object type: 
Description: Definition of an DHCP Pool within a Subnet
```
Creating custom fields for IP-Address lease time data

Custom Field

Content types: IPAM > IP Address

Name: dhcp_lease
Internal field name

Label: DHCP Lease time
Name of the field as displayed to users (if not provided, the field's name will be used)

Group name: kea-dhcp
Custom fields within the same group will be displayed together

Type: Date
The type of data stored in this field. For object/multi-object fields, select the related object type below.

Object type: 
Type of the related object (for object/multi-object fields only)

Required
If true, this field is required when creating new objects or editing an existing object.

Description: DHCP Lease time (read only - set from Kea DHCP Server)
Kea DHCP Subnet definitions, DHCP pools, DHCP reservations and options from within NetBox
Matching NetBox data objects with Kea DHCP configuration blocks

- In the following slides we offer some ideas how NetBox objects can be mapped into Kea DHCP and back
- There is no universal truth in these mappings, and some mappings are a matter of taste
- There are other ways to do this, and depending on the network and the use of NetBox, you will need to adapt these mappings to your needs
Kea DHCP Subnets

- We will use the NetBox build in "Prefix" object to create Kea DHCP subnet definitions
Kea DHCP Subnets

# remove existing kea subnet configuration
del (kea_config['arguments']['Dhcp4']['subnet4'])

# Array of Kea Subnets
kea_subnets = []

# Fetch all subnets from NetBox
subnets = nb.ipam.prefixes.all()

# Iterate over all subnet from NetBox
for subnet in subnets:
    subnet_name = subnet['prefix']
    dhcp_options = subnet['custom_fields']['dhcp_option']
    dhcp_pool = subnet['custom_fields']['dhcp_pool']

    # Create new Array of Subnets for Kea
    kea_subnet = {}
    print("Adding Subnet:", subnet_name)
    kea_subnet['subnet'] = subnet_name
    [ ... ]
Kea DHCP Pools

- We use a NetBox custom fields "dhcp_pool" to define dynamic DHCP Pools within the Kea DHCP Subnet definitions

<table>
<thead>
<tr>
<th>Custom Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>kea-dhcp</td>
</tr>
</tbody>
</table>

**DHCP Pool**: 192.0.2.100-192.0.2.200

Definition of an DHCP Pool within a Subnet
Kea DHCP Pools

```python
# Iterate over all subnet from NetBox
for subnet in subnets:
    [...]  
    kea_subnet["subnet"] = subnet_name

# If the custom fields "dhcp_pool" is filled
if dhcp_pool != None:
    print(" with DHCP Pool:", dhcp_pool)
    subnet_pools = []

    # Append a new Kea DHCP Pool structure
    pool = {}
    pool["pool"] = dhcp_pool

    # Append the new Pool to the Pools-Array
    subnet_pools.append(pool)

    # Add the Pools Array to the Kea Subnet config
    kea_subnet["pools"] = subnet_pools
    [...]"
```
### Kea Host Reservations

- We use the NetBox build in "IP Address" object together with custom fields to store DHCP options for Kea DHCP reservations

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address</td>
<td>192.0.2.10/32</td>
</tr>
<tr>
<td>Status</td>
<td>Active</td>
</tr>
<tr>
<td>Role</td>
<td></td>
</tr>
<tr>
<td>VRF</td>
<td></td>
</tr>
<tr>
<td>DNS Name</td>
<td>test.example.com</td>
</tr>
<tr>
<td>Description</td>
<td>One example host</td>
</tr>
<tr>
<td>Tags</td>
<td>Demo, Webinar</td>
</tr>
</tbody>
</table>
Kea Host Reservations

[
# remove existing kea (global) reservations from the configuration
if 'reservations' in kea_config['arguments']['Dhcp4']:
    del (kea_config['arguments']['Dhcp4']['reservations'])

# Array of Kea DHCP reservations
kea_reservations = []

# Iterating over all IP-Addresses from NetBox
for ip_address in ip_addresses:
    address = ip_address['address']
    # remove prefix notation from ip-address
    if '/' in address:
        address = address.split('/')[0]
    hw_address = ip_address['custom_fields']['hw_address']
    dhcp_options = ip_address['custom_fields']['dhcp_option']
    hostname = ip_address['dns_name']

    reservation = { "hw-address": hw_address, "hostname": hostname, "ip-address": address }

    [ ... process DHCP options ... ]

    # add new reservation to configuration
    print("Adding reservation: ", address)
    kea_reservations.append(reservation)

# Add the reservations to the Kea configuration
kea_config['arguments']['Dhcp4']['reservations'] = kea_reservations

[...]

Kea DHCP Options

- DHCP Options are stored in custom fields along with IP-Addresses (Reservations), IP-Ranges (Pools) or Prefixes (Subnets)
- We choose to store DHCP options as a text field with a simple structure of `![option-name|option-code: option-value-csv]. The optional ! in front of a option marks a *must send* option
- Example 1: `domain-name-servers: 9.9.9.9, 1.1.1.1`
- Example 2: `!15: example.com`
- Example 3: `!routers: 192.0.2.1`
Kea DHCP Options

Custom Fields

kea-dhcp

DHCP Lease time

Field is set to read-only.

domain-name-servers: 9.9.9.9,1.1.1.1;
route: 192.0.2.1;
custom-option: "Hello World"
Kea DHCP Options

# Iterate over all subnet from NetBox
for subnet in subnets:
    # Add the DHCP options
    if dhcp_options != None:
        # Remove line-breaks
        dhcp_options = ' '.join(dhcp_options.splitlines())
        # Split options into array
        options = dhcp_options.split(';
        kea_options = []
        for option in options:
            kea_option = {}
            option = option.strip()
            if len(option) > 1:
                always_send = (option[0] == '!'')
                if always_send:
                    option = option[1:]
                    print(" with DHCP Option:", option)
                    option = option.split(':
                    kea_option['name'] = option[0]
                    kea_option['data'] = option[1]
                    kea_option['always-send'] = always_send

                    # Add new Kea-Option to the options array
                    kea_options.append(kea_option)

                # Add the Options Array to the Kea Subnet config
                kea_subnet['option-data'] = kea_options
            [...]

...
Exporting NetBox data into Kea DHCP
Exporting NetBox data into Kea DHCP

- Our example Python integration periodically reads the DHCP configuration data from NetBox, reads the current Kea DHCP configuration and writes back the changed Kea DHCP server configuration into the Kea DHCP server
- It would be possible to use the NetBox webhooks to trigger dynamic reconfiguration of Kea DHCP whenever the data in NetBox changes
Importing Kea DHCP data into NetBox
Importing Kea DHCP lease times into NetBox

- After the new Kea DHCP server configuration has been applied, the example Python script reads the current leases from Kea DHCP using the `lease commands` hook API and writes the data back into IP-Address objects in NetBox
  - The `lease_cmds` hook in Kea DHCP is used to export the lease information
  - The custom field "lease time" is used to store the current lease time
Python example code to import leases from Kea DHCP to netbox

```python
# Read current leases from Kea DHCP
print("Updating lease information from Kea DHCP into NetBox")
kea_leases = server.dhcp4.lease4_get_all([1])
kea_leases = kea_leases["arguments"]["leases"]
for lease in kea_leases:
    leasetime = lease["cltt"]
    lease_cltt = datetime.utcfromtimestamp(leasetime).strftime('%Y-%m-%d %H:%M:%S')
    print("  Lease for " + lease["ip-address"] +": valid until " + lease_cltt + " UTC")
    try:
        nb_ip_address = nb.ipam.ip_addresses.get(address=lease["ip-address"])  
        if nb_ip_address == None:
            nb_ip_address.address = nb.ipam.ip_addresses.create(
                address=lease["ip-address"],
                description="Added from Kea DHCP"
            )
        nb_ip_address.custom_fields["dhcp_lease"] = lease_cltt
        nb_ip_address.save()
    except pynetbox.lib.query.RequestParamError as e:
        print(e.error)
```
The example code in action

- Demo time
Resources

- Blog Post: netbox integration with KEA DHCP Server
  https://www.sys4.de/blog/netbox-kea-dhcp.en/
- Source repository with example code from this webinar
  https://github.com/cstrotm/kea-dhcp-netbox-webinar
- Kea Lease commands hook
  https://kea.readthedocs.io/en/kea-2.2.0/arm/hooks.html?highlight=lease_cmds#lease_cmds-lease-commands-for-easier-lease-management
Resources

- The NetBox community mailing list: [https://netboxlabs-23319422.hs-sites.com/join-the-netbox-community](https://netboxlabs-23319422.hs-sites.com/join-the-netbox-community)
- JSON-View commandline JSON viewer [https://github.com/doorOfChoice/json-view](https://github.com/doorOfChoice/json-view)
Upcoming ISC Webinar

- 16 May - Migrating to Kea from ISC DHCP
- 07 Jun - Using the new dynamic templates in Kea
Questions / Answers