Infoblox Client Documentation

Release 0.6.0

John Belamaric

Nov 29, 2022

Contents

1 Infoblox Client **3**
- 1.1 Installation . . . 3
- 1.2 Usage . . . 3
- 1.3 Objects Interface . . . 5
- 1.4 Supported NIOS objects . . . 6
- 1.5 Search by regular expression . . . 18
- 1.6 More examples . . . 18
- 1.7 Features . . . 20

2 Installation **21**

3 Usage **23**

4 Examples **25**
- 4.1 Print WAPI response page by page . . . 25

5 Contributing **27**
- 5.1 Types of Contributions . . . 27
- 5.2 Get Started! . . . 28
- 5.3 Pull Request Guidelines . . . 29
- 5.4 Tips . . . 29

6 Credits **31**
- 6.1 Development Lead . . . 31
- 6.2 Contributors . . . 31

7 History **33**
- 7.1 0.6.0 (2022-11-18) . . . 33
- 7.2 0.5.2 (2022-10-12) . . . 33
- 7.3 0.5.1 (2022-03-14) . . . 33
- 7.4 **0.5.0 (2020-05-14)** . . . 34
- 7.5 0.4.25 (2020-03-12) . . . 34
- 7.6 0.4.24 (2020-02-25) . . . 34
- 7.7 0.4.23 (2019-09-10) . . . 34
- 7.8 0.4.22 (2019-02-21) . . . 34
- 7.9 0.4.21 (2019-01-18) . . . 34
- 7.10 0.4.20 (2018-03-27) . . . 34

7.11 0.4.19 (2018-02-06) . . . 35
7.12 0.4.18 (2017-11-20) . . . 35
7.13 0.4.17 (2017-11-09) . . . 35
7.14 0.4.15 (2017-07-18) . . . 35
7.15 0.4.14 (2017-05-18) . . . 35
7.16 0.4.13 (2017-03-01) . . . 35
7.17 0.4.12 (2016-12-08) . . . 35
7.18 0.4.11 (2016-10-31) . . . 35
7.19 0.4.10 (2016-10-24) . . . 35
7.20 0.4.9 (2016-10-24) . . . 36
7.21 0.4.8 (2016-10-10) . . . 36
7.22 0.4.7 (2016-07-14) . . . 36
7.23 0.4.6 (2016-07-01) . . . 36
7.24 0.4.5 (2016-06-13) . . . 36
7.25 0.4.4 (2016-05-11) . . . 36
7.26 0.4.3 (2016-03-28) . . . 36
7.27 0.4.2 (2016-03-04) . . . 37
7.28 0.4.1 (2016-02-26) . . . 37
7.29 **0.4.0 (2016-02-19)** . . . 37
7.30 0.3.9 (2016-02-18) . . . 37
7.31 0.3.8 (2016-02-17) . . . 37
7.32 0.3.7 (2016-02-12) . . . 37
7.33 0.3.6 (2016-01-28) . . . 37
7.34 0.3.5 (2016-01-22) . . . 37
7.35 0.3.4 (2016-01-21) . . . 38
7.36 0.3.3 (2016-01-20) . . . 38
7.37 0.3.2 (2016-01-19) . . . 38
7.38 0.3.1 (2016-01-14) . . . 38
7.39 **0.3.0 (2016-01-14)** . . . 38
7.40 0.2.3 (2016-01-06) . . . 38
7.41 0.2.2 (2015-12-23) . . . 39
7.42 0.2.1 (2015-12-18) . . . 39
7.43 **0.2.0 (2015-12-17)** . . . 39
7.44 0.1.4 (2015-12-08) . . . 39
7.45 0.1.3 (2015-12-04) . . . 39
7.46 0.1.2 (2015-12-02) . . . 40
7.47 0.1.1 (2015-12-01) . . . 40
7.48 0.1.0 (2015-12-01) . . . 40
7.49 0.0.11 (2015-11-25) . . . 40
7.50 0.0.10 (2015-11-19) . . . 40
7.51 0.0.9 (2015-11-13) . . . 40
7.52 0.0.8 (2015-11-12) . . . 41
7.53 0.0.7 (2015-10-27) . . . 41
7.54 0.0.6 (2015-10-26) . . . 41
7.55 0.0.5 (2015-10-12) . . . 41
7.56 0.0.4 (2015-09-23) . . . 41
7.57 0.0.3 (2015-09-15) . . . 41
7.58 0.0.2 (2015-09-11) . . . 41
7.59 **0.0.1 (2015-09-11)** . . . 42

8 Indices and tables **43**

Contents:

CHAPTER 1

Infoblox Client

Client for interacting with Infoblox NIOS over WAPI.

- Free software: Apache license
- Documentation: https://infoblox-client.readthedocs.org.

1.1 Installation

Install infoblox-client using pip:

```
pip install infoblox-client
```

1.2 Usage

Configure logger prior to loading infoblox_client to get all debug messages in console:

```
import logging
logging.basicConfig(level=logging.DEBUG)
```

1.2.1 Low level API, using connector module

Retrieve list of network views from NIOS:

```
from infoblox_client import connector

opts = {'host': '192.168.1.10', 'username': 'admin', 'password': 'admin'}
conn = connector.Connector(opts)
# get all network_views
network_views = conn.get_object('networkview')
# search network by cidr in specific network view
network = conn.get_object('network', {'network': '100.0.0.0/8', 'network_view':
↪'default'})
```

For these request data is returned as list of dicts:

```
network_views:
[{u'_ref': u'networkview/ZG5zLm5ldHdvcmtfdmlldyQw:default/true',
  u'is_default': True,
  u'name': u'default'}]

network:
[{u'_ref': u'network/ZG5zLm5ldHdvcmskMTAwLjAuMC4wLzgvMA:100.0.0.0/8/default',
  u'network': u'100.0.0.0/8',
  u'network_view': u'default'}]
```

1.2.2 High level API, using objects

Example of creating Network View, Network, DNS View, DNSZone and HostRecord using NIOS objects:

```
from infoblox_client import connector
from infoblox_client import objects

opts = {'host': '192.168.1.10', 'username': 'admin', 'password': 'admin'}
conn = connector.Connector(opts)
```

Create a network view, and network:

```
nview = objects.NetworkView.create(conn, name='my_view')
network = objects.Network.create(conn, network_view='my_view', cidr='192.168.1.0/24')
```

Create a DNS view and zone:

```
view = objects.DNSView.create(conn, network_view='my_view', name='my_dns_view')
zone = objects.DNSZone.create(conn, view='my_dns_view', fqdn='my_zone.com')
```

Create a host record:

```
my_ip = objects.IP.create(ip='192.168.1.25', mac='aa:bb:cc:11:22:33')
hr = objects.HostRecord.create(conn, view='my_dns_view',
                               name='my_host_record.my_zone.com', ip=my_ip)
```

Create host record with Extensible Attributes (EA):

```
ea = objects.EA({'Tenant ID': tenantid, 'CMP Type': cmptype,
                 'Cloud API Owned': True})
host = objects.HostRecord.create(conn, name='new_host', ip=my_ip, extattrs=ea)
```

Create a host record with inherited Extensible Attributes (EA):

```
my_ip = objects.IP.create(ip='192.168.1.25', mac='aa:bb:cc:11:22:33', use_for_ea_
↪inheritance=True)
hr = objects.HostRecord.create(conn, view='my_dns_view',
                               name='my_host_record.my_zone.com', ip=my_ip)
```

Set the TTL to 30 minutes:

```
hr = objects.HostRecord.create(conn, view='my_dns_view',
                               name='my_host_record.my_zone.com', ip=my_ip,
                               ttl = 1800)
```

Create a new host record, from the next available IP in a CIDR, with a MAC address, and DHCP enabled:

```
next = objects.IPAllocation.next_available_ip_from_cidr('default', '10.0.0.0/24')
my_ip = objects.IP.create(ip=next, mac='aa:bb:cc:11:22:33', configure_for_dhcp=True)
host = objects.HostRecord.create(conn, name='some.valid.fqdn', view='Internal', ip=my_
↪ip)
```

Reply from NIOS is parsed back into objects and contains next data:

```
In [22]: hr
Out[22]: HostRecordV4: _ref=record:host/
↪ZG5zLmhvc3QkLjQuY29tLm15X3pvbmUubXlfaG9zdF9yZWNvcmQ:my_host_record.my_zone.com/my_
↪dns_view, name=my_host_record.my_zone.com, ipv4addrs=[<infoblox_client.objects.IPv4␣
↪object at 0x7f7d6b0fe9d0>], view=my_dns_view
```

Create a new fixed address, with a MS server DHCP reservation:

```
obj, created = objects.FixedAddress.create_check_exists(connector=conn,
                                                        ip='192.168.100.100',
                                                        mac='aa:bb:cc:11:22:33',
                                                        comment='My DHCP reservation',
                                                        name='My hostname',
                                                        network_view='default',
                                                        ms_server={'_struct':
↪'msdhcpserver',
                                                                   'ipv4addr': '192.
↪168.0.0'})
```

1.2.3 High level API, using InfobloxObjectManager

Create a new fixed address, selecting it from the next available IP in a CIDR:

```
from infoblox_client.object_manager import InfobloxObjectManager

new_address = InfobloxObjectManager(conn).create_fixed_address_from_cidr(netview=
↪'default', mac='aa:bb:cc:11:22:33', cidr='10.0.0.0/24', extattrs=[])
```

What you get back is a `FixedAddressV4` object.

1.3 Objects Interface

All top level objects support interface for CRUD operations. List of supported objects is defined in next section.

- **create(cls, connector, check_if_exists=True, update_if_exists=False, **kwargs)** Creates object on NIOS side. Requires connector passed as the first argument, `check_if_exists` and `update_if_exists` are optional. Object related fields are passed in as kwargs: `field=value`, `field2=value2`.
- **search(cls, connector, return_fields=None, search_extattrs=None, force_proxy=False, **** Search single object on NIOS side, returns first object that match search criteria. Requires connector passed as the first argument. `return_fields` can be set to retrieve particular fields from NIOS, for example `return_fields=['view', 'name']`. If `return_fields` is `[]` default `return_fields` are returned by NIOS side for current `wapi_version`. `search_extattrs` is used to filter out results by extensible attributes. `force_proxy` forces search request to be processed on Grid Master (applies only in cloud environment)
- **search_all(cls, connector, return_fields=None, search_extattrs=None, force_proxy=False** Search all objects on NIOS side that match search criteria. Returns a list of objects. All other options are equal to `search()`.
- **update(self)** Update the object on NIOS side by pushing changes done in the local object.
- **delete(self)** Deletes the object from NIOS side.

1.4 Supported NIOS objects

All NIOS Objects are supported in the 0.6.0 verison release. check infoblox_client/objects.py for description of the objects. Newly supported objects

- `AAAADtcRecord`
- `AAAARecord`
- `AAAASharedRecord`
- `ADtcRecord`
- `ADtcRecordBase`
- `ARecord`
- `ARecordBase`
- `ASharedRecord`
- `ASharedRecordBase`
- `AdAuthServer`
- `AdAuthService`
- `Addressac`
- `Admingroup`
- `Adminrole`
- `Adminuser`
- `AliasRecord`
- `Allendpoints`
- `Allnsgroup`
- `Allrecords`

- Allrpzrecords
- AnyMember
- Approvalworkflow
- Authpolicy
- Awsrte53Task
- Awsrte53Taskgroup
- Awsuser
- BaseObject
- Bfdtemplate
- Bgpas
- Bulkhost
- Bulkhostnametemplate
- CNAMEDtcRecord
- CNAMERecord
- CNAMESharedRecord
- CaaRecord
- Cacertificate
- Capacityreport
- CapacityreportObjectcount
- Captiveportal
- CaptiveportalFile
- CertificateAuthservice
- Changedobject
- CiscoiseEndpoint
- Clientsubnetdomain
- Csvimporttask
- DHCPLease
- DHCPRoamingHost
- DNSView
- DNSZone
- DNSZoneDelegated
- DNSZoneForward
- DbObjects
- Dbsnapshot
- DdnsPrincipalcluster
- DdnsPrincipalclusterGroup

- DeletedObjects
- DhcidRecord
- DhcpOptionDefinition
- DhcpOptionDefinitionV4
- DhcpOptionDefinitionV6
- DhcpOptionSpace
- DhcpOptionSpaceV4
- DhcpOptionSpaceV6
- DhcpStatistics
- Dhcpddns
- Dhcpfailover
- Dhcpmember
- Dhcpoption
- Discovery
- DiscoveryAutoconversionsetting
- DiscoveryCiscoapicconfiguration
- DiscoveryClicredential
- DiscoveryDevice
- DiscoveryDevicecomponent
- DiscoveryDeviceinterface
- DiscoveryDeviceneighbor
- DiscoveryDevicesupportbundle
- DiscoveryDiagnostictask
- DiscoveryGridproperties
- DiscoveryIfaddrinfo
- DiscoveryMemberproperties
- DiscoveryNetworkinfo
- DiscoveryPort
- DiscoveryScaninterface
- DiscoverySeedrouter
- DiscoverySnmp3Credential
- DiscoverySnmpcredential
- DiscoveryStatus
- DiscoveryVlaninfo
- DiscoveryVrf
- DiscoveryVrfmappingrule

- Discoverytask
- Discoverytaskport
- Discoverytaskvserver
- Distributionschedule
- DnameRecord
- Dns64Group
- DnskeyRecord
- Dnsseckey
- Dnssectrustedkey
- DsRecord
- Dtc
- DtcAllrecords
- DtcCertificate
- DtcLbdn
- DtcMonitor
- DtcMonitorHttp
- DtcMonitorIcmp
- DtcMonitorPdp
- DtcMonitorSip
- DtcMonitorSnmp
- DtcMonitorSnmpOid
- DtcMonitorTcp
- DtcObject
- DtcPool
- DtcPoolConsolidatedMonitorHealth
- DtcPoolLink
- DtcServer
- DtcServerLink
- DtcServerMonitor
- DtcTopology
- DtcTopologyLabel
- DtcTopologyRule
- DtcTopologyRuleSource
- DtclbdnRecord
- DxlEndpoint
- DxlEndpointBroker

- EA
- EADefinition
- Exclusionrange
- Exclusionrangetemplate
- ExtensibleattributedefListvalues
- Extserver
- Extsyslogbackupserver
- Fileop
- Filterfingerprint
- Filtermac
- Filternac
- Filteroption
- Filterrelayagent
- Filterrule
- Fingerprint
- FixedAddress
- FixedAddressTemplate
- FixedAddressTemplateV4
- FixedAddressTemplateV6
- FixedAddressV4
- FixedAddressV6
- Forwardingmemberserver
- Ftpuser
- Grid
- GridCloudapi
- GridCloudapiCloudstatistics
- GridCloudapiUser
- GridCloudapiVm
- GridCloudapiVmaddress
- GridDashboard
- GridDhcpproperties
- GridDns
- GridDnsFixedrrsetorderfqdn
- GridFiledistribution
- GridLicensePool
- GridLicensePoolContainer

- GridLicensesubpool
- GridMaxminddbinfo
- GridMemberCloudapi
- GridServicerestartGroup
- GridServicerestartGroupOrder
- GridServicerestartRequest
- GridServicerestartRequestChangedobject
- GridServicerestartStatus
- GridThreatanalytics
- GridThreatprotection
- GridX509Certificate
- GridmemberSoamname
- GridmemberSoaserial
- HostRecord
- HostRecordV4
- HostRecordV6
- Hostnamerewritepolicy
- Hotfix
- HsmAllgroups
- HsmSafenet
- HsmSafenetgroup
- HsmThales
- HsmThalesgroup
- IP
- IPAddress
- IPAllocation
- IPRange
- IPRangeV4
- IPRangeV6
- IPv4
- IPv4Address
- IPv4HostAddress
- IPv6
- IPv6Address
- IPv6HostAddress
- InfobloxObject

- Interface
- IpamStatistics
- Ipv6Networksetting
- Kerberoskey
- LdapAuthService
- LdapEamapping
- LdapServer
- LicenseGridwide
- LocaluserAuthservice
- Logicfilterrule
- Lomnetworkconfig
- Lomuser
- MXRecord
- MXSharedRecord
- Macfilteraddress
- Mastergrid
- Member
- MemberDhcpproperties
- MemberDns
- MemberDnsgluerecordaddr
- MemberDnsip
- MemberFiledistribution
- MemberLicense
- MemberParentalcontrol
- MemberThreatanalytics
- MemberThreatprotection
- Memberserver
- Memberservicecommunication
- Memberservicestatus
- Msdhcpoption
- Msdhcpserver
- Msdnsserver
- Msserver
- MsserverAdsitesDomain
- MsserverAdsitesSite
- MsserverDcnsrecordcreation

- MsserverDhcp
- MsserverDns
- Mssuperscope
- Namedacl
- NaptrDtcRecord
- NaptrRecord
- Natgroup
- Network
- NetworkContainer
- NetworkContainerV4
- NetworkContainerV6
- NetworkDiscovery
- NetworkTemplate
- NetworkTemplateV4
- NetworkTemplateV6
- NetworkV4
- NetworkV6
- NetworkView
- Networkuser
- NetworkviewAssocmember
- Nodeinfo
- NotificationRestEndpoint
- NotificationRestTemplate
- NotificationRestTemplateparameter
- NotificationRule
- NotificationRuleexpressionop
- NsRecord
- Nsec3ParamRecord
- Nsec3Record
- NsecRecord
- Nsgroup
- NsgroupDelegation
- NsgroupForwardingmember
- NsgroupForwardstubserver
- NsgroupStubmember
- Nxdomainrule

- OcspResponder
- Option60Matchrule
- Orderedranges
- Orderedresponsepolicyzones
- Ospf
- OutboundCloudclient
- OutboundCloudclientEvent
- ParentalcontrolAbs
- ParentalcontrolAvp
- ParentalcontrolBlockingpolicy
- ParentalcontrolIpspacediscriminator
- ParentalcontrolMsp
- ParentalcontrolNasgateway
- ParentalcontrolSitemember
- ParentalcontrolSpm
- ParentalcontrolSubscriber
- ParentalcontrolSubscribersite
- Permission
- PtrRecord
- PtrRecordV4
- PtrRecordV6
- RadiusAuthservice
- RadiusServer
- RangeTemplate
- RangeTemplateV4
- RangeTemplateV6
- Rdatasubfield
- Recordnamepolicy
- Remoteddnszone
- Restartservicestatus
- Rir
- RirOrganization
- RpzAIpaddressRecord
- RpzARecord
- RpzAaaaIpaddressRecord
- RpzAaaaRecord

- RpzCnameClientipaddressRecord
- RpzCnameClientipaddressdnRecord
- RpzCnameIpaddressRecord
- RpzCnameIpaddressdnRecord
- RpzCnameRecord
- RpzMxRecord
- RpzNaptrRecord
- RpzPtrRecord
- RpzPtrRecordV4
- RpzPtrRecordV6
- RpzSrvRecord
- RpzTxtRecord
- RrsigRecord
- Ruleset
- SRVDtcRecord
- SRVRecord
- SRVSharedRecord
- SamlAuthservice
- Scavengingtask
- Scheduledtask
- Search
- SettingNetwork
- SettingViewaddress
- SharedNetwork
- SharedNetworkV4
- SharedNetworkV6
- Sharedrecordgroup
- SmartfolderChildren
- SmartfolderGlobal
- SmartfolderGroupby
- SmartfolderPersonal
- SmartfolderQueryitem
- Snmpuser
- Sortlist
- SubObjects
- Superhost

- Superhostchild
- SyslogEndpoint
- SyslogEndpointServers
- Syslogserver
- TXTRecord
- TXTSharedRecord
- TacacsplusAuthservice
- TacacsplusServer
- Taxii
- TaxiiRpzconfig
- Tenant
- Tftpfiledir
- ThreatanalyticsModuleset
- ThreatanalyticsWhitelist
- ThreatinsightCloudclient
- ThreatprotectionGridRule
- ThreatprotectionNatrule
- ThreatprotectionProfile
- ThreatprotectionProfileRule
- ThreatprotectionRule
- ThreatprotectionRulecategory
- ThreatprotectionRuleset
- ThreatprotectionRuletemplate
- ThreatprotectionStatinfo
- ThreatprotectionStatistics
- Thresholdtrap
- TlsaRecord
- Trapnotification
- UnknownRecord
- Updatesdownloadmemberconfig
- Upgradegroup
- UpgradegroupMember
- UpgradegroupSchedule
- Upgradeschedule
- Upgradestatus
- Upgradestep

- Userprofile
- Vdiscoverytask
- Vlan
- Vlanlink
- Vlanrange
- Vlanview
- Vtftpdirmember
- ZoneAuthDiscrepancy
- ZoneRp
- ZoneStub
- Zoneassociation
- Zonenameserver

Until 0.4.25 this project supported

- NetworkView for 'networkview'
- DNSView for 'view'
- DNSZone for 'zone_auth'
- Member for 'member'
- Network (V4 and V6)
  - NetworkV4 for 'network'
  - NetworkV6 for 'ipv6network'
- IPRange (V4 and V6)
  - IPRangeV4 for 'range'
  - IPRangeV6 for 'ipv6range'
- HostRecord (V4 and V6)
  - HostRecordV4 for 'record:host'
  - HostRecordV6 for 'record:host'
- FixedAddress (V4 and V6)
  - FixedAddressV4 for 'fixedaddress'
  - FixedAddressV6 for 'ipv6fixedaddress'
- IPAddress (V4 and V6)
  - IPv4Address for 'ipv4address'
  - IPv6Address for 'ipv6address'
- ARecordBase
  - ARecord for 'record:a'
  - AAAARecord for 'record:aaaa'
- PtrRecord (V4 and V6)

  - PtrRecordV4 for 'record:ptr'
  - PtrRecordV6 for 'record:ptr'
- EADefinition for 'extensibleattributedef'
- CNAMERecord for 'record:cname'
- MXRecord for 'record:mx'

1.5 Search by regular expression

Search for partial match is supported only by low-level API for now. Use '~' with field name to search by regular expressions. Not all fields support search by regular expression. Refer to wapidoc to find out complete list of fields that can be searched this way. Examples:

Find all networks that starts with '10.10.':

```
conn = connector.Connector(opts)
nw = conn.get_object('network', {'network~': '10.10.'})
```

Find all host records that starts with '10.10.':

```
conn = connector.Connector(opts)
hr = conn.get_object('record:host', {'ipv4addr~': '10.10.'})
```

1.6 More examples

Utilizing extensible attributes and searching on them can easily be done with the get_object function. The default field in return_fields acts like the + does in WAPI.

> _return_fields+ Specified list of fields (comma separated) will be returned in addition to the basic fields of the object (documented for each object).

This enables you to always get the default values in return, in addition to what you specify whether you search for a network or a networkcontainer, defined as place_to_check in the code below.

```
from infoblox_client.connector import Connector


def default_infoblox_connection():
    opts = {'host': '192.168.1.10', 'username': 'admin', 'password': 'admin'}
    conn = Connector(opts)
    return conn

def search_extensible_attribute(connection, place_to_check: str, extensible_
↪attribute: str, value: str):
    """
    Find extensible attributes.
    :param connection: Infoblox connection
    :param place_to_check: Can be `network`, `networkcontainer` or `record:host` and␣
↪so on.
    :param extensible_attribute: Which extensible attribute to search for. Can be␣
↪`CustomerCode`, `Location`
    and so on.
```

(continues on next page)

(continued from previous page)

```
    :param value: The value you want to search for.
    :return: result
    """
    extensible_args = [
        place_to_check,
        {
            f"*{extensible_attribute}:~": value,
        }
    ]
    kwargs = {
        'return_fields': [
            'default',
            'extattrs',
        ]
    }
    result = {"type": f"{place_to_check}", "objects": connection.get_
↪object(*extensible_args, **kwargs)}
    return result

connection = default_infoblox_connection()

search_network = search_extensible_attribute(connection, "network", "CustomerCode",
↪"Infoblox")
# Print the output:
print(search_network)
{
  "type": "network",
  "objects": [
    {
      "_ref": "network/ZG5zLmhvc3QkLjQuY29tLm15X3pvbmUubXlfaG9zdF9yZWNvcmQ:192.168.1.
↪1/28/default",
      "comment": "Infoblox Network",
      "extattrs": {
        "CustomerCode": {
          "value": "Infoblox"
        }
      },
      "network": "192.168.1.0/28",
      "network_view": "default"
    }
  ]
}

search_host = search_extensible_attribute(connection, "record:host", "CustomerCode",
↪"Infoblox")
# Print the output:
print(search_host)
{
  "type": "record:host",
  "objects": [
    {
      "_ref": "record:host/ZG5zLm5ldHdvcmtfdmlldyQw:InfobloxHost",
      "extattrs": {
        "CustomerCode": {
          "value": "Infoblox"
        }
      },
```

(continues on next page)

(continued from previous page)

```
      "ipv4addrs": [
        {
          "_ref": "record:host_ipv4addr/ZG5zLm5ldHdvcmtfdmlldyQwdvcmtfdmlldyQw:192.
↪168.1.1/InfobloxHost",
          "configure_for_dhcp": false,
          "host": "InfobloxHost",
          "ipv4addr": "192.168.1.1"
        }
      ],
      "name": "InfobloxHost",
      "view": " "
    }
  ]
}
```

1.7 Features

- TODO

CHAPTER 2

Installation

At the command line:

```
$ easy_install infoblox-client
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv infoblox-client
$ pip install infoblox-client
```

CHAPTER 3

Usage

To use Infoblox Client in a project:

```
import infoblox-client
```

CHAPTER 4

Examples

4.1 Print WAPI response page by page

If you want to print WAPI response page by page, please use the `paging()` helper function as described in the following example:

```
from infoblox_client import connector, objects, utils

opts = {'host': '192.168.1.10', 'username': 'admin', 'password': 'admin'}
conn = connector.Connector(opts)

resp = objects.DNSZone.search_all(conn, view='default', paging=True)

for page in utils.paging(resp, max_results=2):
    print(page)
    input("Press enter to read more...")
```

CHAPTER 5

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/infobloxopen/infoblox-client/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

5.1.4 Write Documentation

Infoblox Client could always use more documentation, whether as part of the official Infoblox Client docs, in docstrings, or even on the web in blog posts, articles, and such.

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/infobloxopen/infoblox-client/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

5.2 Get Started!

Ready to contribute? Here's how to set up *infoblox-client* for local development.

1. Fork the *infoblox-client* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/infoblox-client.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv infoblox-client
$ cd infoblox-client/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 infoblox-client tests
$ python setup.py test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.4 and above. Check https://travis-ci.org/infobloxopen/infoblox-client/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ python -m unittest infoblox_client.tests.test_infoblox_client
```

CHAPTER 6

Credits

6.1 Development Lead

- Pavel Bondar <pbondar@infoblox.com>

6.2 Contributors

- John Belamaric <jbelamaric@infoblox.com>
- Hosung Hwang <hhwang@infoblox.com>
- Yue Ko <yko@infoblox.com>
- Robert Grant <rhgrant10@gmail.com>
- Yakau Bubnou <yakau_bubnou@epam.com>
- Jonas Krüger Svensson <jonas-ks@hotmail.com>
- Hiroyasu OHYAMA <user.localhost2000@gmail.com>

More are always welcome!

CHAPTER 7

History

7.1 0.6.0 (2022-11-18)

- Added support for Python version 3.9 #352;
- Removed support for Python version below 3.0 #352;

7.2 0.5.2 (2022-10-12)

- Added Certificate based authentication logic #330;
- Fixed use of EA inheritance in IP Objects #318;
- Fixed missing fields ('ipv4addr', 'ipv6addr') for 'class Member()' #345;

7.3 0.5.1 (2022-03-14)

- Updated connector's urlencoding logic for proper array encoding #287;
- Updated InfobloxObject's fetch method to raise *InfobloxFetchGotMultipleObjects* exception #288;
- Fix a bug when calling abstracted class from_dict with V4 & V6 subclass #282;
- Fix a bug when updating DNSZone object exception was raised and field not allowed to update #331;
- Fix a bug when ARecord and AAAARecord object skips updating the updatable fields #334, #328;
- Raised an exception while searching with non searchable fields #339;
- Fix errors generated for the client using sphinx with make docs #343;

7.4 0.5.0 (2020-05-14)

- Majorly Updated objects with around 380+ NIOS object calls supported now.(Find the infoblox_client/objects.py file to list the supported objects and its descriptions)
- Bug Fixes
- python-six dependency set to >=1.11.0

7.5 0.4.25 (2020-03-12)

- Bug Fixes

7.6 0.4.24 (2020-02-25)

- Added some extra fields(ms_server) for Fixed Address
- Supporting MX record
- Bug Fixes - PTR records now return an IP

7.7 0.4.23 (2019-09-10)

- Added some extra fields for network class
- Fixed update option for A Record
- Adding fields for fixed address

7.8 0.4.22 (2019-02-21)

- Supported returning default fileds plus user required fields reflecting WAPI
- Supporting 'aliases' parameter of HOST record for DNS

7.9 0.4.21 (2019-01-18)

- Supporting wapi version 2.10 or above

7.10 0.4.20 (2018-03-27)

- Updated default WAPI version from 1.4 to 2.1

7.11 0.4.19 (2018-02-06)

- Changed logging of failure on object search from Error to Warning

7.12 0.4.18 (2017-11-20)

- Fix bug related to temporary unavailable status code

7.13 0.4.17 (2017-11-09)

- Added pagination support for wapi calls

7.14 0.4.15 (2017-07-18)

- Changed logic of generate duid using only mac address

7.15 0.4.14 (2017-05-18)

- Add function to check object is created or reused

7.16 0.4.13 (2017-03-01)

- Add TTL field to HostRecordV*
- Add CNAME record support
- Specify return fields for an SRV record

7.17 0.4.12 (2016-12-08)

- Allow search all fields
- Remove ptrdname from PTR record search attributes

7.18 0.4.11 (2016-10-31)

- Add search HostRecords by MAC

7.19 0.4.10 (2016-10-24)

- Updated history and author

7.20 0.4.9 (2016-10-24)

- Add function to get fixed addresses by mac

7.21 0.4.8 (2016-10-10)

- Add ptrdname search option to PtrRecord objects

7.22 0.4.7 (2016-07-14)

- Add zones extensible attribute update support

7.23 0.4.6 (2016-07-01)

- Add network_view support for host records

7.24 0.4.5 (2016-06-13)

- Allow raising exception in create_ea_definition
- Add pep8 check to tox
- Add pep8 check to Travis CI
- Add examples of searching by regular expression

7.25 0.4.4 (2016-05-11)

- Pass only_ref option to update_from_dict
- Do not fail on processing unknown fields
- Fetch only object reference for service restart
- Update README with example of using EA

7.26 0.4.3 (2016-03-28)

- Add default fields for Member
- Update docstring for create_network
- Add fields to FixedAddressV4 and IPAddress

7.27 0.4.2 (2016-03-04)

- Add max_retries option to connector
- Log failure on get with Error log level

7.28 0.4.1 (2016-02-26)

- Add 'max_results' as connector option

7.29 0.4.0 (2016-02-19)

- Add max_results option to connector and objects
- Add Tenant object
- Update README.rst with more examples

7.30 0.3.9 (2016-02-18)

- Add 'configure_for_dns' field for HostRecord

7.31 0.3.8 (2016-02-17)

- Add 'extattrs' to DNSZone/DNSView return_fields

7.32 0.3.7 (2016-02-12)

- Add return_fields to NetworkView

7.33 0.3.6 (2016-01-28)

- Add support for list and tuple values to EA object
- Remove _value_to_bool

7.34 0.3.5 (2016-01-22)

- No changes

7.35 0.3.4 (2016-01-21)

- Do not override verify flag on request level

7.36 0.3.3 (2016-01-20)

- create_required_ea_definitions return created list
- Add 'start_addr', 'end_addr' to ip detection list
- Add request type to connector logger
- Flake8 fixes

7.37 0.3.2 (2016-01-19)

- Convert strings into booleans for ssl_verify
- Update AUTHORS.rst, add contributors
- Remove unused methods from utils.py

7.38 0.3.1 (2016-01-14)

- Add 'zone' to search fields of Host Record

7.39 0.3.0 (2016-01-14)

- Update development status from Pre-Alpha to Alpha
- Feature/tox testing (huge changes in testing env)
- Add 'network' to search fields of FixedAddress
- Allow domain-name-servers for ipv6
- Update existent EA for network instead of replace

7.40 0.2.3 (2016-01-06)

- Return None if search failed instead of exception
- Add ip_version as a public property for objects

7.41 0.2.2 (2015-12-23)

- Fix updating object from create method
- Rework delete_all_associated_objects logic
- Fix error handling in create_object
- Do not catch exception on create_dns_zone level
- Update feature version for member_ipv6_setting

7.42 0.2.1 (2015-12-18)

- Add InfobloxMemberAlreadyAssigned exception
- Update dns record if already exists
- Add 'log_api_calls_as_info' option for connector
- Check for empty values in EA

7.43 0.2.0 (2015-12-17)

- Deprecate network_exists method in object_manager
- Add _global_field_processing for objects
- Add parsing 'extattrs' into EA objects for all InfobloxObject childs
- Add docs badge to README.rst
- Reworked get_network in object_manager
- Move _eq_ to BaseObject
- Check if fixed address is found before delete

7.44 0.1.4 (2015-12-08)

- Field updates for Member object
- Log all api calls in connector on debug level

7.45 0.1.3 (2015-12-04)

- Add 'network' field to ip versioned fields
- Skip adding DHCP options for IPv6 network
- Do not search IPRange before creating

7.46 0.1.2 (2015-12-02)

- Do not fail if object is not found on delete
- Raise exception with details if reply is not json
- Add 'silent_ssl_warnings' option to connector

7.47 0.1.1 (2015-12-01)

- Fix unbind_name_from_record_a

7.48 0.1.0 (2015-12-01)

- Add new field type '_updateable_search_field' to objects and fix HostRecord search
- Fix 'make docs'
- Update README.rst (fixed formatting)

7.49 0.0.11 (2015-11-25)

- Fix adding second ip to HostRecord
- Fix failing in pdb
- Convert EA values into boolean if possible
- Added 'ips' allias for ip field in HostRecord

7.50 0.0.10 (2015-11-19)

- Add utility to determine supported feature
- Update README.rst with objects interface

7.51 0.0.9 (2015-11-13)

- Add allowed_object_types field for EA Definition
- Allow to return default fields for object
- Update README.rst with list of supported objects

7.52 0.0.8 (2015-11-12)

- Add Extensible Attributes Definition support
- Fixed options processing for create_network in object_manager
- Fixed missed DNSZone object in create_dns_zone

7.53 0.0.7 (2015-10-27)

- Added 'network' to IPRange search fields
- Modified *get* method of the EA class to allow return default values

7.54 0.0.6 (2015-10-26)

- Added initial support of Extensible Attributes as sub objects
- Added search by Extensible Attributes
- Improved validation in connector
- Added delete_object_by_ref to object manager

7.55 0.0.5 (2015-10-12)

- Fixed issues in working with objects
- Added missed _get_object_type_from_ref
- Added code coverage
- Updated links to point to infobloxopen repository

7.56 0.0.4 (2015-09-23)

- Added object abstraction for interacting with NIOS objects
- Added object_manager to simplify some operations on objects

7.57 0.0.3 (2015-09-15)

- Added dependencies to package.

7.58 0.0.2 (2015-09-11)

- Fixed using dashes in package directory names that prevented package import after install.

7.59 0.0.1 (2015-09-11)

- Added connector to send wapi requests to NIOS, does not includes NIOS object model at this point.
- First release on PyPI.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search