

---

# **infoblox Documentation**

***Release 1.1.1***

**Gavin M. Roy**

May 06, 2015



<b>1</b>	<b>Requirements</b>	<b>3</b>
<b>2</b>	<b>Installation</b>	<b>5</b>
<b>3</b>	<b>API Documentation</b>	<b>7</b>
<b>4</b>	<b>CLI Usage</b>	<b>9</b>
<b>5</b>	<b>Contents</b>	<b>11</b>
5.1	infoblox API . . . . .	11
<b>6</b>	<b>Indices and tables</b>	<b>19</b>



An unofficial python library for interfacing with Infoblox NIOS. This library is not affiliated with [Infoblox, Inc.](#) in any way.



---

## Requirements

---

Python 2.6, 2.7, 3.2, 3.3

**External dependencies:**

- [argparse](#) (Python 2.6 only)
- [requests](#)





---

## Installation

---

infoblox is available on the [Python Package Index](#) and can be installed with *easy\_install* or *pip*:

```
pip install infoblox
```



---

## API Documentation

---

The following classes are available for interaction with the Infoblox NIOS device:

- *infoblox.Session*
- *infoblox.Host*
- *infoblox.HostIPv4*
- *infoblox.HostIPv6*



---

## CLI Usage

---

```
usage: infoblox-host      usage: infoblox-host [-h] [--version] [--debug] [-u USERNAME] [-p PASSWORD]
      <Infoblox Address> {add,remove} <FQDN> [IPv4 Address] [COMMENT]
```

Add or remove a host from the Infoblox appliance

positional arguments:

<Infoblox Address>	The Infoblox hostname
{add,remove}	Specify if you are adding or removing a host
<FQDN>	The FQDN for the host
[IPv4 Address]	The IPv4 address for the host
[COMMENT]	A comment set on the host when adding.

optional arguments:

-h, --help	show this help message and exit
--version	show program's version number and exit
--debug	Enable debug output
-u USERNAME, --username USERNAME	The username to perform the work as. Default: admin
-p PASSWORD, --password PASSWORD	The password to authenticate with. Default: infoblox



---

## Contents

---

### 5.1 infoblox API

To interact with an Infoblox device, you must first create a Session object instance that will be passed to any object you create. The following example shows how to create a host.:

```
import infoblox

session = infoblox.Session('127.0.0.1', 'admin', 'infoblox')
host = infoblox.Host(session)
host.name = 'foo.bar.net'
host.add_ipv4addr('10.0.0.1')
if host.save():
    print('Host saved')
```

**class** `infoblox.Session` (*host, username=None, password=None, https=True*)  
Central object for managing HTTP requests to the Infoblox appliance.

**class** `infoblox.Host` (*session, reference\_id=None, name=None, \*\*kwargs*)  
Implements the host record type.

Example:

```
session = infoblox.Session(infoblox_host,
                           infoblox_user,
                           infoblox_password)
host = infoblox.Host(session, name='foo.bar.net')
```

**add\_ipv4addr** (*ipv4addr*)

Add an IPv4 address to the host.

**Parameters** **ipv4addr** (*str*) – The IP address to add.

**Raises** `ValueError`

**add\_ipv6addr** (*ipv6addr*)

Add an IPv6 address to the host.

**Parameters** **ipv6addr** (*str*) – The IP address to add.

**Raises** `ValueError`

**as\_dict** ()

Return this object as a dict value.

**Return type** `dict`

**clear()**

Clear all set attributes in the mapping.

**delete()**

Remove the item from the infoblox server.

**Return type** bool

**Raises** AssertionError

**Raises** ValueError

**Raises** infoblox.exceptions.ProtocolError

**dirty**

Indicate if the mapping has changes from it's initial state

**Return type** bool

**dumps()**

Return a JSON serialized version of the mapping.

**Return type** strunicode

**fetch()**

Attempt to fetch the object from the Infoblox device. If successful the object will be updated and the method will return True.

**Return type** bool

**Raises** infoblox.exceptions.ProtocolError

**from\_dict(values)**

Assign the values from the dict passed in. All items in the dict are assigned as attributes of the object.

**Parameters** **values** (*dict*) – The dictionary of values to assign to this mapping

**get(key, default=None)**

Get the value of key, passing in a default value if it is not set.

**Parameters**

- **key** (*str*) – The attribute to get
- **default** (*mixed*) – The default value

**Return type** mixed

**items()**

Return a list of attribute name and value tuples for this mapping.

**Return type** list

**iteritems()**

Iterate through a list of the attribute names and their values.

**Return type** listiterator

**iterkeys()**

Iterate through the attribute names for this mapping.

**Return type** listiterator

**itervalues()**

Iterate through a list of the attribute values for this mapping.

**Return type** listiterator



**keys()**

Return a list of attribute names for the mapping.

**Return type** list

**loads(value)**

Load in a serialized value, overwriting any previous values.

**Parameters** **value** (*strunicode*) – The serialized value

**reference\_id()**

Return a read-only handle for the `reference_id` of this object.

**remove\_ipv4addr(ipv4addr)**

Remove an IPv4 address from the host.

**Parameters** **ipv4addr** (*str*) – The IP address to remove

**remove\_ipv6addr(ipv6addr)**

Remove an IPv6 address from the host.

**Parameters** **ipv6addr** (*str*) – The IP address to remove

**save()**

Update the infoblox with new values for the specified object, or add the values if it's a new object all together.

**Raises** `AssertionError`

**Raises** `infoblox.exceptions.ProtocolError`

**set(key, value)**

Set the value of key.

**Parameters**

- **key** (*str*) – The attribute to set
- **value** (*mixed*) – The value to set

**Raises** `KeyError`

**values()**

Return a list of values for this mapping in attribute name order.

:rtype list

**class** `infoblox.HostIPv4(session, reference_id=None, ipv4addr=None, **kwargs)`

Implements the `host_ipv4addr` record type.

**as\_dict()**

Return this object as a dict value.

**Return type** dict

**clear()**

Clear all set attributes in the mapping.

**delete()**

Remove the item from the infoblox server.

**Return type** bool

**Raises** `AssertionError`

**Raises** `ValueError`

**Raises** infoblox.exceptions.ProtocolError

**dirty**

Indicate if the mapping has changes from it's initial state

**Return type** bool

**dumps** ()

Return a JSON serialized version of the mapping.

**Return type** strunicode

**fetch** ()

Attempt to fetch the object from the Infoblox device. If successful the object will be updated and the method will return True.

**Return type** bool

**Raises** infoblox.exceptions.ProtocolError

**from\_dict** (*values*)

Assign the values from the dict passed in. All items in the dict are assigned as attributes of the object.

**Parameters** **values** (*dict*) – The dictionary of values to assign to this mapping

**get** (*key*, *default=None*)

Get the value of key, passing in a default value if it is not set.

**Parameters**

- **key** (*str*) – The attribute to get
- **default** (*mixed*) – The default value

**Return type** mixed

**items** ()

Return a list of attribute name and value tuples for this mapping.

**Return type** list

**iteritems** ()

Iterate through a list of the attribute names and their values.

**Return type** listiterator

**iterkeys** ()

Iterate through the attribute names for this mapping.

**Return type** listiterator

**intervalues** ()

Iterate through a list of the attribute values for this mapping.

**Return type** listiterator

**keys** ()

Return a list of attribute names for the mapping.

**Return type** list

**loads** (*value*)

Load in a serialized value, overwriting any previous values.

**Parameters** **value** (*strunicode*) – The serialized value

**reference\_id()**

Return a read-only handle for the reference\_id of this object.

**save()**

Update the infoblox with new values for the specified object, or add the values if it's a new object all together.

**Raises** AssertionError

**Raises** infoblox.exceptions.ProtocolError

**set** (*key*, *value*)

Set the value of key.

**Parameters**

- **key** (*str*) – The attribute to set
- **value** (*mixed*) – The value to set

**Raises** KeyError

**values()**

Return a list of values for this mapping in attribute name order.

:rtype list

**class** infoblox.**HostIPv6** (*session*, *reference\_id=None*, *ipv6addr=None*, *ipv6bits=None*,  
*ipv6prefix\_bits=None*, *\*\*kwargs*)

Implements the host\_ipv6addr record type.

**as\_dict()**

Return this object as a dict value.

**Return type** dict

**clear()**

Clear all set attributes in the mapping.

**delete()**

Remove the item from the infoblox server.

**Return type** bool

**Raises** AssertionError

**Raises** ValueError

**Raises** infoblox.exceptions.ProtocolError

**dirty**

Indicate if the mapping has changes from it's initial state

**Return type** bool

**dumps()**

Return a JSON serialized version of the mapping.

**Return type** strunicode

**fetch()**

Attempt to fetch the object from the Infoblox device. If successful the object will be updated and the method will return True.

**Return type** bool

**Raises** infoblox.exceptions.ProtocolError

**from\_dict** (*values*)

Assign the values from the dict passed in. All items in the dict are assigned as attributes of the object.

**Parameters** **values** (*dict*) – The dictionary of values to assign to this mapping

**get** (*key*, *default=None*)

Get the value of key, passing in a default value if it is not set.

**Parameters**

- **key** (*str*) – The attribute to get
- **default** (*mixed*) – The default value

**Return type** *mixed*

**items** ()

Return a list of attribute name and value tuples for this mapping.

**Return type** *list*

**iteritems** ()

Iterate through a list of the attribute names and their values.

**Return type** *listiterator*

**iterkeys** ()

Iterate through the attribute names for this mapping.

**Return type** *listiterator*

**intervalues** ()

Iterate through a list of the attribute values for this mapping.

**Return type** *listiterator*

**keys** ()

Return a list of attribute names for the mapping.

**Return type** *list*

**loads** (*value*)

Load in a serialized value, overwriting any previous values.

**Parameters** **value** (*str|unicode*) – The serialized value

**reference\_id** ()

Return a read-only handle for the reference\_id of this object.

**save** ()

Update the infoblox with new values for the specified object, or add the values if it's a new object all together.

**Raises** `AssertionError`

**Raises** `infoblox.exceptions.ProtocolError`

**set** (*key*, *value*)

Set the value of key.

**Parameters**

- **key** (*str*) – The attribute to set
- **value** (*mixed*) – The value to set

**Raises** `KeyError`

**values** ()

Return a list of values for this mapping in attribute name order.

:rtype list



---

## Indices and tables

---

- `genindex`
- `modindex`
- `search`





## A

`add_ipv4addr()` (infoblox.Host method), 11  
`add_ipv6addr()` (infoblox.Host method), 11  
`as_dict()` (infoblox.Host method), 11  
`as_dict()` (infoblox.HostIPv4 method), 13  
`as_dict()` (infoblox.HostIPv6 method), 15

## C

`clear()` (infoblox.Host method), 11  
`clear()` (infoblox.HostIPv4 method), 13  
`clear()` (infoblox.HostIPv6 method), 15

## D

`delete()` (infoblox.Host method), 12  
`delete()` (infoblox.HostIPv4 method), 13  
`delete()` (infoblox.HostIPv6 method), 15  
`dirty` (infoblox.Host attribute), 12  
`dirty` (infoblox.HostIPv4 attribute), 14  
`dirty` (infoblox.HostIPv6 attribute), 15  
`dumps()` (infoblox.Host method), 12  
`dumps()` (infoblox.HostIPv4 method), 14  
`dumps()` (infoblox.HostIPv6 method), 15

## F

`fetch()` (infoblox.Host method), 12  
`fetch()` (infoblox.HostIPv4 method), 14  
`fetch()` (infoblox.HostIPv6 method), 15  
`from_dict()` (infoblox.Host method), 12  
`from_dict()` (infoblox.HostIPv4 method), 14  
`from_dict()` (infoblox.HostIPv6 method), 15

## G

`get()` (infoblox.Host method), 12  
`get()` (infoblox.HostIPv4 method), 14  
`get()` (infoblox.HostIPv6 method), 16

## H

`Host` (class in infoblox), 11  
`HostIPv4` (class in infoblox), 13  
`HostIPv6` (class in infoblox), 15

## I

`items()` (infoblox.Host method), 12  
`items()` (infoblox.HostIPv4 method), 14  
`items()` (infoblox.HostIPv6 method), 16  
`iteritems()` (infoblox.Host method), 12  
`iteritems()` (infoblox.HostIPv4 method), 14  
`iteritems()` (infoblox.HostIPv6 method), 16  
`iterkeys()` (infoblox.Host method), 12  
`iterkeys()` (infoblox.HostIPv4 method), 14  
`iterkeys()` (infoblox.HostIPv6 method), 16  
`itervalues()` (infoblox.Host method), 12  
`itervalues()` (infoblox.HostIPv4 method), 14  
`itervalues()` (infoblox.HostIPv6 method), 16

## K

`keys()` (infoblox.Host method), 12  
`keys()` (infoblox.HostIPv4 method), 14  
`keys()` (infoblox.HostIPv6 method), 16

## L

`loads()` (infoblox.Host method), 13  
`loads()` (infoblox.HostIPv4 method), 14  
`loads()` (infoblox.HostIPv6 method), 16

## R

`reference_id()` (infoblox.Host method), 13  
`reference_id()` (infoblox.HostIPv4 method), 14  
`reference_id()` (infoblox.HostIPv6 method), 16  
`remove_ipv4addr()` (infoblox.Host method), 13  
`remove_ipv6addr()` (infoblox.Host method), 13

## S

`save()` (infoblox.Host method), 13  
`save()` (infoblox.HostIPv4 method), 15  
`save()` (infoblox.HostIPv6 method), 16  
`Session` (class in infoblox), 11  
`set()` (infoblox.Host method), 13  
`set()` (infoblox.HostIPv4 method), 15  
`set()` (infoblox.HostIPv6 method), 16

## V

[values\(\) \(infoblox.Host method\), 13](#)

[values\(\) \(infoblox.HostIPv4 method\), 15](#)

[values\(\) \(infoblox.HostIPv6 method\), 16](#)