netconf_client Documentation

ADTRAN

Jan 25, 2019

Contents

CHAPTER 1

Quick Start

In order to connect to a NETCONF server we can use one of the `connect` methods from the `netconf_client.connect` module. In our example we will connect to a NETCONF server running over SSH, so we will use the `connect_ssh` function.:

```
from netconf_client.connect import connect_ssh

with connect_ssh(host='192.0.2.1',
                 port=830,
                 username='admin',
                 password='password') as sesssion:
    # TODO: Do things with the session object
    pass
```

The object returned from any of the `connect` functions is a `Session` object. It acts as a context manager, and as such it should generally be used alongside a `with` statement. It is importat to either use a `with` statement with the object, or to manually call `Session.close` in order to free the sockets associated with the connection.

The `Session` object can be used to send and receive raw messages. However, a higher-level API is desireable in most circumstances. For this we can use the `Manager` class.

The `Manager` class is from the `netconf_client.ncclient` module. The module overall attempts to mimick the most common uses of the ncclient API (another, Open Source, Python NETCONF client). Most of the common NETCONF operations such as performing an `<edit-config>` or a `<get>` are implemented as functions of this class.

In this example we will perform an `<edit-config>` for a single node, and then run a `<get-config>` to see the change.:

```
from netconf_client.connect import connect_ssh
from netconf_client.ncclient import Manager

with connect_ssh(host='192.0.2.1',
                 port=830,
                 username='admin',
```

(continues on next page)

(continued from previous page)

```
                password='password') as sesssion:
    mgr = Manager(session, timeout=120)
    mgr.edit_config(target='running', '''
        <config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
          <service-alpha xmlns="http://example.com">
            <simple-string>Foo</simple-string>
          </service-alpha>
        </config>''')
    print(mgr.get_config(source='running').data_xml)
```

An instance of the `Manager` class should be a drop-in replacement for a manager object from `ncclient`.

CHAPTER 2

Migrating

To migrate from the `ncclient` API to the `netconf_client` API you can generally follow these two steps.

First change your imports. For example, convert from:

```
from ncclient.operations import RPCError
from ncclient.xml_ import to_ele
```

To this:

```
from netconf_client.ncclient import RPCError, to_ele
```

Then you will need to migrate your connection code. For example, if your old connection method looked like this:

```
def mgr():
    from ncclient import manager, operations
    m = manager.connect_ssh(host='localhost', port=830,
                            username='root', password='password',
                            hostkey_verify=False,
                            timeout=120,
                            )
    m.raise_mode = operations.RaiseMode.ALL
    return m
```

Then your new connection code should look like this:

```
def mgr():
    from netconf_client.connect import connect_ssh
    from netconf_client.ncclient import Manager

    s = connect_ssh(host='localhost', port=830,
                    username='root',  password='password')
    return Manager(s, timeout=120)
```

As long as the existing code isn't doing anything too crazy, these should be the only changes needed.

CHAPTER 3

netconf_client API

3.1 netconf_client.connect

3.2 netconf_client.ncclient

3.3 netconf_client.session

3.4 netconf_client.error

CHAPTER 4

Indices and tables

- genindex
- modindex
- search