
netconf Documentation

Release

Christian E. Hopps

Jan 17, 2018

Contents

1	Installation	3
2	Usage	5
2.1	Command Line Tool	5
2.2	Development	5
3	Reference	7
3.1	The <code>netconf.base</code> Module	7
3.2	The <code>netconf.client</code> Module	8
3.3	The <code>netconf.error</code> Module	8
3.4	The <code>netconf.ncclient</code> Module	9
3.5	The <code>netconf.server</code> Module	9
3.6	The <code>netconf.util</code> Module	11
	Python Module Index	13

This package supports creating both netconf clients and servers. Additionally a CLI netconf client is included. The following modules are present:

- `base` - Shared netconf support classes.
- `error` - Netconf error classes.
- `client` - Netconf client classes.
- `server` - Netconf server classes.
- `util` - Netconf utility functions.

netconf uses `_sshutil` and thus supports your SSH agent and SSH config when using the client.

Contents:

CHAPTER 1

Installation

At the command line:

```
$ pip install sshutil
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv sshutil  
$ pip install sshutil
```


2.1 Command Line Tool

To get the capabilities of a server:

```
$ netconf-client --hello --host example.com
```

To request config from a server that has your key:

```
$ netconf-client --host example.com <<<"<get-config/>"
```

2.2 Development

To use netconf in a project:

```
import netconf
```

2.2.1 Netconf Client

To open a session to server:

```
from netconf.client import NetconfSSHSession

session = NetconfSSHSession(host, port, username, password)
```

To send and RPC to a server:

```
rpcout = session.send_rpc("<my-rpc/>")
```

2.2.2 Netconf Server

To create a simple server listening on port 830 that handles one RPC <my-cool-rpc>:

```
from netconf import nsmmap_update, server
import netconf.util as ncutil

MODEL_NS = "urn:my-urn:my-model"
nsmmap_update({'pfx': MODEL_NS})

class MyServer (object):
    def __init__ (self, user, pw):
        controller = server.SSHUserPassController(username=user, password=pw)
        self.server = server.NetconfSSHServer(server_ctl=controller, server_
↪methods=self)

    def nc_append_capabilities(self, caps):
        ncutil.subelm(caps, "capability").text = MODEL_NS

    def rpc_my_cool_rpc (self, session, rpc, *params):
        data = ncutil.elm("data")
        data.append(ncutil.leaf_elm("pfx:result", "RPC result string"))
        return data

# ...
server = MyServer("myuser", "mysecert")
# ...
```

3.1 The `netconf.base` Module

```
class netconf.base.NetconfFramingTransport (stream, max_chunk, debug)
    Bases: netconf.base.NetconfPacketTransport
    Packetize an ssh stream into netconf PDUs – doesn't need to be SSH specific
    close()
    is_active()
    receive_pdu (new_framing)
    send_pdu (msg, new_framing)
class netconf.base.NetconfPacketTransport
    Bases: object
    receive_pdu (new_framing)
    send_pdu (msg, new_framing)
class netconf.base.NetconfSession (stream, debug, session_id, max_chunk=16384)
    Bases: object
    Netconf Protocol Server and Client
    close()
    is_active()
    reader_exits()
    reader_handle_message (msg)
    send_hello (caplist, session_id=None)
    send_message (msg)
```

```
class netconf.base.NetconfTransportMixin
```

```
    Bases: object
```

```
    close()
```

```
    connect()
```

```
netconf.base.chunkit(msg, maxsend, minsend=0, pad=u'\n')
```

chunkit iterates over a msg returning chunks of at most maxsend size, and of at least minsend size if non-zero. Padding will be added if required. This function currently requires that maxsend is at least large enough to hold 2 minsend chunks.

```
netconf.base.lookahead(iterable)
```

Return an element and an indication if it's the last element

3.2 The netconf.client Module

```
class netconf.client.NetconfClientSession(stream, debug=False)
```

```
    Bases: netconf.base.NetconfSession
```

```
    Netconf Protocol
```

```
    close()
```

```
    is_reply_ready(msg_id)
```

```
        Check whether reply is ready (or session closed)
```

```
    reader_exits()
```

```
    reader_handle_message(msg)
```

```
        Handle a message, lock is already held
```

```
    send_rpc(rpc, timeout=None)
```

```
    send_rpc_async(rpc, noreply=False)
```

```
    wait_reply(msg_id, timeout=None)
```

```
class netconf.client.NetconfSSHSession(host, port=830, username=None, password=None,
                                       debug=False, cache=None, proxycmd=None)
```

```
    Bases: netconf.client.NetconfClientSession
```

```
class netconf.client.Timeout(timeout)
```

```
    Bases: object
```

```
    is_expired()
```

```
    remaining()
```

3.3 The netconf.error Module

```
exception netconf.error.ChannelClosed
```

```
    Bases: netconf.error.NetconfException
```

```
exception netconf.error.FramingError
```

```
    Bases: netconf.error.NetconfException
```

```
exception netconf.error.NetconfException
```

```
    Bases: exceptions.Exception
```

```

exception netconf.error.RPCError(output, tree, error)
    Bases: netconf.error.NetconfException

    get_error_info()
    get_error_severity()
    get_error_tag()
    get_error_type()

exception netconf.error.RPCServerError(origmsg, etype, tag, **kwargs)
    Bases: netconf.error.NetconfException

    get_reply_msg()

exception netconf.error.RPCSvrBadElement(origmsg, element, **kwargs)
    Bases: netconf.error.RPCServerError

exception netconf.error.RPCSvrErrBadMsg(origmsg)
    Bases: netconf.error.RPCServerError

    If the server raises this exception the and netconf 1.0 is in use, the session will be closed

exception netconf.error.RPCSvrErrNotImpl(origmsg, **kwargs)
    Bases: netconf.error.RPCServerError

exception netconf.error.RPCSvrException(origmsg, exception, **kwargs)
    Bases: netconf.error.RPCServerError

exception netconf.error.RPCSvrInvalidValue(origmsg, **kwargs)
    Bases: netconf.error.RPCServerError

exception netconf.error.RPCSvrMissingElement(origmsg, tag, **kwargs)
    Bases: netconf.error.RPCServerError

exception netconf.error.RPCSvrUnknownElement(origmsg, element, **kwargs)
    Bases: netconf.error.RPCServerError

exception netconf.error.ReplyTimeoutError
    Bases: netconf.error.NetconfException

exception netconf.error.SessionError
    Bases: netconf.error.NetconfException

netconf.error.TimeoutError
    alias of ReplyTimeoutError

```

3.4 The netconf.ncclient Module

```
netconf.ncclient.main(*margs)
```

3.5 The netconf.server Module

```

class netconf.server.NetconfMethods
    Bases: object

```

This is an abstract class that is used to document the server methods functionality

The server return not-implemented if the method is not found in the methods object, so feel free to use duck-typing here (i.e., no need to inherit)

nc_append_capabilities (*capabilities*)

The server should append any capabilities it supports to capabilities

rpc_copy_config (*unused_session, rpc, *unused_params*)

rpc_delete_config (*unused_session, rpc, *unused_params*)

rpc_edit_config (*unused_session, rpc, *unused_params*)

rpc_get (*session, rpc, filter_or_none*)

Passed the filter element or None if not present

rpc_get_config (*session, rpc, source_elm, filter_or_none*)

Passed the source element

rpc_lock (*unused_session, rpc, *unused_params*)

rpc_unlock (*unused_session, rpc, *unused_params*)

class netconf.server.**NetconfSSHServer** (*server_ctl=None, server_methods=None, port=830, host_key=None, debug=False*)

Bases: sshutil.server.SSHServer

A netconf server

allocate_session_id()

class netconf.server.**NetconfServerSession** (*channel, server, unused_extra_args, debug*)

Bases: *netconf.base.NetconfSession*

Netconf Server-side Session Protocol

close()

handled_rpc_methods = **set**([**u**'close-session', **u**'kill-session'])

reader_exits()

reader_handle_message (*msg*)

Handle a message, lock is already held

send_rpc_reply (*rpc_reply, origmsg*)

send_rpc_reply_error (*error*)

class netconf.server.**SSHAuthController** (*users=None*)

Bases: paramiko.server.ServerInterface

check_auth_none (*unused_username*)

check_auth_password (*username, password*)

check_auth_publickey (*username, key*)

check_channel_request (*kind, chanid*)

check_channel_subsystem_request (*channel, name*)

get_allowed_auths (*username*)

get_user_auth_keys (*username*)

Parse the users's authorized_keys file if any to look for authorized keys

class netconf.server.**SSHUserPassController** (*username=None, password=None*)

Bases: paramiko.server.ServerInterface

```

check_auth_none (username)
check_auth_password (username, password)
check_channel_request (kind, chanid)
check_channel_subsystem_request (channel, name)
get_allowed_auths (username)

```

3.6 The `netconf.util` Module

```
netconf.util.elm(tag, attrib=None, **extra)
```

```
netconf.util.filter_containment_iter(fcontain_elm, dest_node, containment_nodes,
                                     leaf_elms, append_to)
```

Given a containment filter node (or None) verify that all leaf elements either match, have corresponding selection nodes (empty) or are not present.

If all leaf criteria are met then the iterator will return a triple of (`new_filter_node`, `new_dest_node`, `new_data`). `new_filter_node` corresponds to the matched containment node which is returned in `new_dest_node`, and `new_data` will be an element corresponding to the passed in `dest_node`.

These should be processed by calling `filter_containment_iter` again.

Additionally the correct leaf data will be added to `dest_node`, and `dest_node` will be appended to `append_to` if `append_to` is not None.

This implements RFC6241 section 6.2.5

```
netconf.util.filter_leaf_allows(filter_elm, xpath, value)
```

Check the value at the xpath specified leaf matches the value.

- If `filter_elm` is None then allow.
- If there is no xpath element then allow if there are no other children.
- XXX what about xpath that has embedded predicates! perhaps what we want to call this is a normal path not an xpath.

```
netconf.util.filter_leaf_allows_add(filter_elm, tag, data, value)
```

```
netconf.util.filter_leaf_values(fcontain_elm, dest_node, leaf_elms, append_to)
```

Given a containment element (or None) verify that all leaf elements in `leaf_elms` either match, have corresponding selection nodes (empty) or are not present.

Additionally the correct leaf data will be added to `dest_node`, and `dest_node` will be appended to `append_to` if `append_to` is not None.

The return value will be True, False, or a possibly empty set of selection/containment nodes. The only failing value is False, if True is returned then the caller should include all containment sibling nodes, otherwise the caller should process the list of containment/selection nodes.

```
netconf.util.filter_list_iter(filter_list, key_xpath, keys)
```

Return key, elm pairs that are allowed by keys using the values found using the given `key_xpath`

```
netconf.util.filter_node_match(filter_node, match_elm)
```

Given a filter node element and a nodename and attribute dictionary return true if the filter element matches the `elmname`, attributes and value (if not None).

The filter element can use a wildcard namespace or a specific namespace the attributes can be missing from the filter node but otherwise must match and the value is only checked for a match if it is not None.

```
netconf.util.filter_node_match_no_value (filter_node, match_elm)
netconf.util.filter_tag_match (filter_tag, elm_tag)
netconf.util.is_selection_node (felm)
netconf.util.leaf (tag, value, attrib=None, **extra)
netconf.util.leaf_elm (tag, value, attrib=None, **extra)
netconf.util.qname (tag)
netconf.util.subelm (pelm, tag, attrib=None, **extra)
```


n

- `netconf.base`, 7
- `netconf.client`, 8
- `netconf.error`, 8
- `netconf.ncclient`, 9
- `netconf.server`, 9
- `netconf.util`, 11

A

allocate_session_id() (netconf.server.NetconfSSHServer method), 10

C

ChannelClosed, 8

check_auth_none() (netconf.server.SSHAuthController method), 10

check_auth_none() (netconf.server.SSHUserPassController method), 10

check_auth_password() (netconf.server.SSHAuthController method), 10

check_auth_password() (netconf.server.SSHUserPassController method), 11

check_auth_publickey() (netconf.server.SSHAuthController method), 10

check_channel_request() (netconf.server.SSHAuthController method), 10

check_channel_request() (netconf.server.SSHUserPassController method), 11

check_channel_subsystem_request() (netconf.server.SSHAuthController method), 10

check_channel_subsystem_request() (netconf.server.SSHUserPassController method), 11

chunkit() (in module netconf.base), 8

close() (netconf.base.NetconfFramingTransport method), 7

close() (netconf.base.NetconfSession method), 7

close() (netconf.base.NetconfTransportMixin method), 8

close() (netconf.client.NetconfClientSession method), 8

close() (netconf.server.NetconfServerSession method), 10

connect() (netconf.base.NetconfTransportMixin method), 8

E

elm() (in module netconf.util), 11

F

filter_containment_iter() (in module netconf.util), 11

filter_leaf_allows() (in module netconf.util), 11

filter_leaf_allows_add() (in module netconf.util), 11

filter_leaf_values() (in module netconf.util), 11

filter_list_iter() (in module netconf.util), 11

filter_node_match() (in module netconf.util), 11

filter_node_match_no_value() (in module netconf.util), 11

filter_tag_match() (in module netconf.util), 12

FramingError, 8

G

get_allowed_auths() (netconf.server.SSHAuthController method), 10

get_allowed_auths() (netconf.server.SSHUserPassController method), 11

get_error_info() (netconf.error.RPCError method), 9

get_error_severity() (netconf.error.RPCError method), 9

get_error_tag() (netconf.error.RPCError method), 9

get_error_type() (netconf.error.RPCError method), 9

get_reply_msg() (netconf.error.RPCServerError method), 9

get_user_auth_keys() (netconf.server.SSHAuthController method), 10

H

handled_rpc_methods (netconf.server.NetconfServerSession attribute), 10

I

`is_active()` (netconf.base.NetconfFramingTransport method), 7

`is_active()` (netconf.base.NetconfSession method), 7

`is_expired()` (netconf.client.Timeout method), 8

`is_reply_ready()` (netconf.client.NetconfClientSession method), 8

`is_selection_node()` (in module netconf.util), 12

L

`leaf()` (in module netconf.util), 12

`leaf_elm()` (in module netconf.util), 12

`lookahead()` (in module netconf.base), 8

M

`main()` (in module netconf.ncclient), 9

N

`nc_append_capabilities()` (netconf.server.NetconfMethods method), 10

netconf.base (module), 7

netconf.client (module), 8

netconf.error (module), 8

netconf.ncclient (module), 9

netconf.server (module), 9

netconf.util (module), 11

NetconfClientSession (class in netconf.client), 8

NetconfException, 8

NetconfFramingTransport (class in netconf.base), 7

NetconfMethods (class in netconf.server), 9

NetconfPacketTransport (class in netconf.base), 7

NetconfServerSession (class in netconf.server), 10

NetconfSession (class in netconf.base), 7

NetconfSSHServer (class in netconf.server), 10

NetconfSSHSession (class in netconf.client), 8

NetconfTransportMixin (class in netconf.base), 7

Q

`qname()` (in module netconf.util), 12

R

`reader_exits()` (netconf.base.NetconfSession method), 7

`reader_exits()` (netconf.client.NetconfClientSession method), 8

`reader_exits()` (netconf.server.NetconfServerSession method), 10

`reader_handle_message()` (netconf.base.NetconfSession method), 7

`reader_handle_message()` (netconf.client.NetconfClientSession method), 8

`reader_handle_message()` (netconf.server.NetconfServerSession method), 10

`receive_pdu()` (netconf.base.NetconfFramingTransport method), 7

`receive_pdu()` (netconf.base.NetconfPacketTransport method), 7

`remaining()` (netconf.client.Timeout method), 8

ReplyTimeoutError, 9

`rpc_copy_config()` (netconf.server.NetconfMethods method), 10

`rpc_delete_config()` (netconf.server.NetconfMethods method), 10

`rpc_edit_config()` (netconf.server.NetconfMethods method), 10

`rpc_get()` (netconf.server.NetconfMethods method), 10

`rpc_get_config()` (netconf.server.NetconfMethods method), 10

`rpc_lock()` (netconf.server.NetconfMethods method), 10

`rpc_unlock()` (netconf.server.NetconfMethods method), 10

RPCError, 8

RPCServerError, 9

RPCSvrBadElement, 9

RPCSvrErrBadMsg, 9

RPCSvrErrNotImpl, 9

RPCSvrException, 9

RPCSvrInvalidValue, 9

RPCSvrMissingElement, 9

RPCSvrUnknownElement, 9

S

`send_hello()` (netconf.base.NetconfSession method), 7

`send_message()` (netconf.base.NetconfSession method), 7

`send_pdu()` (netconf.base.NetconfFramingTransport method), 7

`send_pdu()` (netconf.base.NetconfPacketTransport method), 7

`send_rpc()` (netconf.client.NetconfClientSession method), 8

`send_rpc_async()` (netconf.client.NetconfClientSession method), 8

`send_rpc_reply()` (netconf.server.NetconfServerSession method), 10

`send_rpc_reply_error()` (netconf.server.NetconfServerSession method), 10

SessionError, 9

SSHAAuthController (class in netconf.server), 10

SSHUserPassController (class in netconf.server), 10

`subelm()` (in module netconf.util), 12

T

Timeout (class in netconf.client), 8

TimeoutError (in module netconf.error), 9

W

`wait_reply()` (`netconf.client.NetconfClientSession`
method), 8