fwOper

Release 0.0.4

ALIASGAR [ALI]

Apr 12, 2022

CONTENTS:

CHAPTER

ONE

FWOPER

1.1 Welcome to fwOper's documentation!

1.1.1 What is fwOper?

fwOper is an open python project to help working with various task for cisco Firewall configurations. Delta configuration changes can be generated based on the input rule change request.

Caution: It is solely users responsibility to review the configuration generated by the fwOper.

Owner of the package or package will not be liable in any manner for any mishap happen then after.

CHAPTER

TWO

INSTALLATION & REQUIREMENTS

2.1 Requirements

1. python >= 3.7

2.2 Installation

Install the fwOper package:

```
pip install --upgrade fwOper
```

2.3 Inherited python packages

- nettoolkit

CHAPTER

THREE

USER DOCUMENTATION!

3.1 fw_Oper.acl User documentation!

3.1.1 Cisco Firewall Access-Lists - How To ?

Use the acl module of fw_oper package to get the necessary changes for the ACL on Cisco Firewall.

Tip: Build your own script in order to get the change delta script generated using this package.

High-level steps:

1. Make a firewall change request excel, csv sheet. read it thru Pandas or other package.
2. Read thru each add/del request.
3. Convert request to dictionary format as required by this package.
4. Execute appropriate request on eligible ACL.
5. At last get the delta changes.

See Also: Sample Execution Steps!

3.1.2 High-level Overview

1. Define inputs
2. Import package, modules
3. select firewall, acls, acl Objects
4. Operate and View acl

3.1.3 Detailed How To

1. Define inputs:

```
file = 'running-config log captuerd file for fw.log'    # fw log

new_entry_to_add = {            # acl detail to add
        'acl_type': 'extended',
        'action': 'permit',
        'protocol': 'tcp',
        'source': '10.10.10.0 255.255.255.0',
        'destination': 'host 2.2.2.2',
        'ports': 'eq 2222',
        'log_warning': True,
        'remark': 'Remark if any',
}

old_entry_to_del = {    # acl detail to del
        'action': 'permit',
        'protocol': 'tcp',
        'source': '158.98.23.194 255.255.255.255',
        'destination': 'host 210.89.6.101',
        'ports': 'eq ssh',
}
```

2. Import necessary package/modules:

```
import fwOper as fw
```

3. Create Firewall Object:

```
with open(file, 'r') as f:
        flst = f.readlines()

insts = fw.get_object(fw.Instances, conf_list=flst)
print(insts)                    # set of instances
```

4. Refereance to Instance Access-lists (set of ALCs)

```
acls = insts['instance_name'].acls
acls = insts.instance_name.acls
print(acls)                     # set of acls
```

   instance_name can be accepted in either **bracket** or **dotted** format. Use of bracket format is must if space/special characters involved in *instance_name*.

5. Select an ACL from set of ACLS

```
acl = acls['acl_name']
acl = acls.acl_name
```

   acl_name can be accepted in either **bracket** or **dotted** format. Use of bracket format is must if space/special characters involved in *acl_name*.

6. Set ACL Numbering enable/disable on given acl:

```
acl.sequence = True      # set sequence numbering enabled (default=disable)
```

7. Operations on ACL

    1. acl views, properties:

    ```
    print(acl)                      # full acl
    print(acl[8:13])                # get range of acl lines
    print(acl.min, acl.max)         # least & maximm acl sequence number.
    ```

    2. add:

    ```
    acl1 = acl + new_entry_to_add   # append new entry, create a new ACL
    acl += new_entry_to_add         # append new entry, same acl
    print(acl.append(new))          # same as above.
    ```

    3. delete:

    ```
    acl1 = acl - old_entry_to_del   # create a new ACL by deleting an old entry.
    acl -= old_entry_to_del         # delete an old entry from existing ACL.
    print(acl.delete(old_entry_to_del))  # same as above
    acl1 = acl - 10                 #  delete acl sequence number `10`
    print(acl.delete(10))           # same as above
    del(acl[210:212])               # delete range of lines from acl.
    print(acl.delete(200, 210, 2))  # delete range of lines from acl, with jump␣
    ↪step.
    print(acl.removals)             # verify, get - deleted entries
    ```

    4. insert:

    ```
    print(acl.insert(10, new))      # insert new entry at position (10)
    ```

    5. verifications:

    ```
    print(old_entry_to_del in acl)  # bool: entry found in acl
    print(acl.contains(old_entry_to_del))# set: of line numbers containing␣
    ↪attributes (sparse matche).
    print(acl.exact(old) )          # set: of line numbers matching attributes␣
    ↪(exact matches)
    ```

    6. comparisions:

    ```
    acl1 = acls.another_acl_name    # select another ACL
    print(acl > acl1)               # acl1 entries missing in acl, diff in two acls
    print(acl < acl1)               # acl entries missing in acl1, diff in two acls
    print( acl == acl1 )            # bool: compare two acls / (exact match)
    print(acl.difference(acl1))     # differences: from acl to acl1
    print(acl1.same_elements(acl))  # bool: compare two acl elements == (sparse␣
    ↪match)
    ```

    7. get delta:

    ```
    print(acls.changes("adds"))     # get additions for all ACLs after apply changes
    print(acls.changes("removals")) # get removals for all ACLs after apply changes
    ```

Warning: Be extra careful on implementatin steps if sequence numbering used.

3.1.4 Extra Nuggets

- **In delta modification dictionary, `source` and `destinations` accepts all three variants of addressing format. And no-mask**

    1. 10.10.10.1 255.255.255.255
    2. host 10.10.10.1
    3. 10.10.10.1/32

- **Multiple `source` and/or `destinations` can be supplied in sets, as below.**
    - 'source': {'1.1.1.1', '1.1.1.2', '1.1.1.3', '1.1.1.4'}
    - 'destination': {'2.1.1.1', '2.1.1.2', '2.1.1.3', '2.1.1.4'},

3.2 fw_Oper.acg User documentation!

3.2.1 Cisco Firewall Object Groups - How To ?

Use the acg module of fw_oper package to get the necessary changes for the Object Groups on Cisco Firewall.

acg stands for Access Control Group (object-group)

Tip: Build your own script in order to get the change delta script generated using this package.

High-level steps:

1. Make a firewall change request excel, csv sheet. read it thru Pandas or other package.
2. Read thru each add/del request.
3. Convert request to dictionary format as required by this package.
4. Find eligible Object Group that requires changes.
5. Execute appropriate change request on eligible Object Group.
6. At last get the delta changes.

3.2.2 High-level Overview

1. Define inputs
2. Import package, modules
3. select firewall, acgs, acg Objects
4. Operate and View group/changes

3.2.3 Detailed How To

1. Define inputs:

```
file = 'running-config log captuerd file for fw.log'    # fw log

a_member = "1.1.1.1 255.255.255.255"
setof_members = {"1.1.1.0 255.255.255.0", "2.2.2.2 255.255.255.255"}
```

2. Import necessary package/modules:

```
import fwOper as fw
```

3. Create Firewall Object:

```
with open(file, 'r') as f:
        flst = f.readlines()

insts = fw.get_object(fw.Instances, conf_list=flst)
print(insts)                    # set of instances
```

4. Refereance to Instance to object-group (set of object-groups)

```
grps = insts['instance_name'].obj_grps
grps = insts.instance_name.obj_grps
print(grps)                     # set of obj_grps
```

   instance_name can be accepted in either **bracket** or **dotted** format. Use of bracket format is must if space/special characters involved in *instance_name*.

5. Select an object-group from set of object-groups

```
grp = grps['group_name']
grp = grps.group_name
```

   group_name can be accepted in either **bracket** or **dotted** format. Use of bracket format is must if space/special characters involved in *group_name*.

6. Operations on object-group

   1. object-group views, properties:

```
print(grp)              # full object-group
print(grp.keys())       # object group MEMBER_TYPES ex: network-object, port-
↪object...
print(grp.values())     # object group MEMBERs ex: address, ports, objgrp␣
↪refereance...
print(len(grp1))        # count of members.
print(grp.description)  # object group description
print(grp['network-object'])    # set of members of given member type.
```

   2. add:

```
print(grp.add(a_member))        # add a member to group, inline
grp += a_member                 # same as above
print(grp.add(setof_members))   # add set of members, inline
```

(continues on next page)

(continued from previous page)

```
grp += setof_members              # same as above
grp1 = grp + setof_members        # creates new group; i.e. copy+add
```

3. delete:

```
print(grp.delete(a_member))       # remove a member from group, inline
grp -= a_member                   # same as above
print(grp.delete(setof_members))# remove set of members, inline
grp -= setof_members              # same as above
grp1 = grp - setof_members        # creates new group; i.e. copy+delete
```

4. verifications:

```
print(a_member in grp)                    # bool: member found in group
print(grp == grp1)                        # bool: checks equality of two groups
print(grp.over(acls))                     # check for acl entries containing␣
↪group.
print(grp1.has(grp))                      # check for grp1 members containing grp.
```

5. comparisions:

```
print(grp > grp1)                         # difference in two group members
print(grp < grp1)                         # difference in two group members
```

6. get delta:

```
print(grp.add_str())    # members added to a group, string
print(grp.del_str())    # members added from a group, negating string
print(grps.changes('adds'))     # add strings for all groups
print(grps.changes('removals')) # negating strings from all groups
```

Warning: Be extra careful on implementatin steps, if group is applied to multiple access-lists.

3.3 fw_Oper.route User documentation!

3.3.1 Cisco Firewall Routes - How To ?

Use the route module of fw_oper package to get the necessary matching entries for the Routes on Cisco Firewall.

Note: For now route changes are not implemented.

3.3.2 High-level Overview

1. Define inputs
2. Import package, modules
3. select firewall, routes, route Objects
4. perform necessary search operations

3.3.3 Detailed How To

1. Define inputs:

```
file = 'running-config log captuerd file for fw.log'    # fw log

prefix = '10.10.10.0/24'
```

2. Import necessary package/modules:

```
import fwOper as fw
```

3. Create Firewall Object:

```
with open(file, 'r') as f:
        flst = f.readlines()

insts = fw.get_object(fw.Instances, conf_list=flst)
print(insts)                    # set of instances
```

4. Refereance to Instance to routes

```
routes = insts['instance_name'].routes
routes = insts.instance_name.routes
print(routes)                   # all routes
```

   instance_name can be accepted in either **bracket** or **dotted** format. Use of bracket format is must if space/special characters involved in *instance_name*.

5. Operations on routes:

```
print(routes)                           # all routes
print(prefix in routes)         # bool: is prefix match any route.
print(routes.prefix(prefix))    # matching route for given prefix
print(routes.prefix(prefix).ifdesc)     # route description for matching route
print(routes.prefix(prefix).route_line) # string prop ( object work without it )
```

3.4 Sample Execution Steps!

Here below is the sample code from my desk, alter it as per your need to get the desired result.

```
# ------------------------------------------------------------------------------
# Imports
# ------------------------------------------------------------------------------
import os
import pandas as pd
from collections import OrderedDict
from pprint import pprint
import fwOper



#  sample DATABASE request.xlsx FORMAT
# ------------------------------------------------------------------------------------------------
↪---------------------------------------------------------------------------- #
# |request_type|firewall_name|firewall_instance|acl_
↪name|action|source|destination|protocol|ports|remark|insert_at|source_grp|destination_
↪grp|ports_grp|protocol_grp|
# ------------------------------------------------------------------------------------------------
↪---------------------------------------------------------------------------- #


# ------------------------------------------------------------------------------
#  global vars
# ------------------------------------------------------------------------------

REQUEST_TYPES = ('del', 'add')
GROUPBY_SEQUENCE = ['request_type', 'firewall_name', 'firewall_instance', 'acl_name']

# ------------------------------------------------------------------------------
#                               FUNCTIONS
# ------------------------------------------------------------------------------

def get_file_name(folder, hostname):
        """file name for the given hostname in folder
        assumed that firewall logs are stored with hostname as filename

        Args:
                folder (str): Folder path
                hostname (str): hostname

        Returns:
                str: filename containing hostname in given folder.
        """
        for file in os.listdir(folder):
                if file.lower().find(hostname.lower()) > -1:
                        return file


def filter_request(request):
```

(continues on next page)

(continued from previous page)

```
        """filters request dictionary for the mentioned fields only

        Args:
                request (dict): input request attributes

        Returns:
                dict: filtered request attributes
        """
        fields = {'source', 'destination', 'protocol', 'ports', 'action', 'remark',
↪'insert_at'}
        return {field: request[field] for field in fields if request[field]}


def execute_req(req_type, Firewalls, req_grp):
        """Excute the ACL Change request

        Args:
                req_type (str): request type (either 'add', 'del')
                Firewalls (dict): Firewall objects dictionary
                req_grp (dict): grouped input requests

        Returns:
                s: delta change for the execution of request
        """
        # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
        # ~~~~ go sequencial on REQUEST_TYPES ('del', 'add') ~~~~
        if req_grp['request_grp']['request_type'] != req_type: return ''
        # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
        # ~~~~ set group request variable ~~~~
        grp_request = req_grp['req_grp']
        # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
        # ~~~~ set fw parameters/variables ~~~~
        fw = Firewalls[req_grp['request_grp']['firewall_name']]
        fw_inst = fw.instances[req_grp['request_grp']['firewall_instance']]
        acl = fw_inst.acls[req_grp['request_grp']['acl_name']]
        # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
        # ~~~~ execute request based on request type ~~~~
        if req_type == 'del':  return execute_del_req(acl, grp_request)
        if req_type == 'add':  return execute_add_req(acl, grp_request)
        # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

def execute_del_req(acl, grp_request):
        """execution of grouped delete requests

        Args:
                acl (ACL): access-list object
                grp_request (list): grouped input request attributes to be deleted on␣
↪given ACL

        Returns:
                str: delta change(s) for given delete request
        """
```

(continues on next page)

(continued from previous page)

```
        ### here some where group check will get insert ### [TBD]
        acl.sequence = False
↪               # Enable if require sequence number in delta output
        s = ''
        for gr in grp_request:
                s += acl.delete(gr)
        return s


def execute_add_req(acl, grp_request):
        """execution of grouped add/insert requests
        'insert_at'-attribute needed per request for inserting, otherwise request will be
↪considered as add(append).

        Args:
                acl (ACL): access-list object
                grp_request (list): grouped input request attributes to be added/
↪inserted on given ACL

        Returns:
                str: delta change(s) for given add/insert request
        """
        acl.sequence = True
        s = ''
        for gr in grp_request:
                if gr.get('insert_at'):
                        n = int(gr['insert_at'])
                        del(gr['insert_at'])
                        s += acl.insert(n, gr)
                else:
                        s += acl.append(gr)
        return s



def check_exact_group()
        source_grp=fwOper.NetworkObject()
        item = 'destination'
        values = set()
        for gr in grp_request:
                values.add(gr['destination'])
        dum_grp = fwOper.dummy_group(source_grp, item, values)



# ------------------------------------------------------------------------------
#                                   CLASSES
# ------------------------------------------------------------------------------

# ------------------------------------------------------------------------------
# REQUEST Parameters
# ------------------------------------------------------------------------------
```

(continues on next page)

(continued from previous page)

```
class Request():
        """Firewall change request (Excel) method, properties, exections
        """

        def __init__(self, request_input_file, sheet_name='Sheet1'):
                """provide excel input file

                Args:
                        request_input_file (str): input request file
                """
                self.request_input_file = request_input_file
                self.get_dataframe(sheet_name)

        def get_dataframe(self, sheet_name):
                """creates data frame (requests), and firewalls name-list in the requests
                """
                self.requests = pd.read_excel(self.request_input_file, sheet_name=sheet_
↪name).fillna("")
                self.firewalls = self.requests.firewall_name.unique()
                self.requests = self.requests.groupby( GROUPBY_SEQUENCE )

        def group_members(self, group):
                """convert the grouping members in dictionary format ( from tuples␣
↪(group, df) ),
                also updaets missing firewall_instance with default 'system'.

                Args:
                        group (dict of tulpes): group members

                Returns:
                        dict: grouping members
                """
                members = {}
                for i, gm in enumerate(GROUPBY_SEQUENCE):
                        if gm == 'firewall_instance' and group[0][i] == '':
                                members[gm] = 'system'
                        else:
                                members[gm] = group[0][i]
                return members

        def create_request_group(self, df):
                """create the requests list based on group

                Args:
                        df (pandas.DataFrame): filtered (member) DataFrame to create a␣
↪list of group request

                Returns:
                        list: input requests
                """
                req_grp = []
                d = df.T.to_dict()
```

(continues on next page)

(continued from previous page)

```
                for i, req in d.items():
                        req_grp.append(filter_request(req))
                return req_grp

        def gen_request_id_groups(self):
                """group input request and get the grouped request format= {id:req_grp_
↪dict}

                Returns:
                        dict: grouped request
                """
                request_id_grp = {}
                for i, group in enumerate(self.requests):
                        request_grp = self.group_members(group)                 # get␣
↪request type
                        request_df = group[1]
                        req_grp = self.create_request_group(request_df) # get group of␣
↪requests.
                        request_id_grp[i] = {'request_grp':request_grp, 'req_grp':req_
↪grp}
                return request_id_grp

# ------------------------------------------------------------------------------
# Firewall Object
# ------------------------------------------------------------------------------
class Firewall(object):
        """A Firewall object

        Args:
                object (object): default
        """

        def __init__(self, folder, firewall):
                """provide folder and firewall name for which Firewall object to be␣
↪created

                attributes:
                        instances: instances of the firewall

                Args:
                        folder (str): folder path where config backup stored
                        firewall (str): firewall name with which backup is stored
                """
                file = get_file_name(folder, firewall)
                self.read(folder, file)

        def read(self, folder, file):
                """reads firewall configuration file from provided folder.

                Raises:
                        Exception: MissingInput
```

(continues on next page)

(continued from previous page)

```
                Args:
                        folder (str): where configuration files stored
                        file (str): filename of config file
                """
                try:
                        with open(folder+file, 'r') as f:
                                fw_lst = f.readlines()
                                self.instances = fwOper.get_object(fwOper.Instances,␣
↪conf_list=fw_lst)
                except:
                        Exception(f"MissingInput{folder+file}")


# ------------------------------------------------------------------------------
# EXECUTION
# ------------------------------------------------------------------------------
if __name__ == '__main__':
        pass
# ------------------------------------------------------------------------------

        ############## WAY OF EXECUTION ##############

        # STEP1: Provide Inputs ----------------------------------------------
        file = 'request.xlsx'
        firewall_backup_folder = '/path_to_firewall_backup_folder/'

        # STEP2: Initialize Request and Firewall inputs ----------------------
        Req = Request(file)
        rigs = Req.gen_request_id_groups()
        Firewalls = {fw: Firewall(firewall_backup_folder, fw) for fw in Req.firewalls}

        # STEP3: Iterate thru requests ---------------------------------------
        for req_type in REQUEST_TYPES:  # default (first='del', second='add')
                for i, req_id_grp in rigs.items():
                        pass
                        s = execute_req(req_type, Firewalls, req_id_grp)
                        if s: print(s)                  # This is delta output



        fw = 'testfw'                                   # provide FW Name to see update
        print(Firewalls[fw].instances.system.acls.al_PERMIT_I)  # This is updated ACL

# ------------------------------------------------------------------------------
```

Thank You!

CHAPTER

FOUR

TECHNICAL DOCUMENTATION!

4.1 acl

class fwOper.acl.**ACL**(*acl_name*, *acl_lines_list*, *objs*)

Bases: *fwOper.fwObj.Singulars*

Individual access-list object

Parameters **Singulars** (Singulars) – Inherits - individual object properties definitions

Raises

- **Exception** – MissingMandatoryParameter
- **Exception** – exact match process error

Returns a single access-list object

Return type *ACL*

Yields *tuple* – tuple of (line-number, line-attributes)

add_str()

String representation of acl recoded additions

Returns recorded acl changes (adds)

Return type str

append(*attribs*)

append a line to acl display warning message - MatchingEntryAlreadyexistAtLine, if a match already exist in acl

Parameters **attribs** (*dict*) – line attributes

Returns delta change(s) for the append of entry

Return type str

contains(*item*)

check matching attributes in acl object, and return set of matching acl line numbers for containing item (sparse match)

Parameters **item** (*dict*) – line attributes

Returns set of matching acl line numbers (sparse match)

Return type set

copy_and_append(*attribs*)

create duplicate of self, append a new acl line in new object with provided attributes

Parameters **attribs** (*dict*) – line attributes

Returns copy of ACL with attributes appended

Return type *ACL*

copy_and_delete(*attribs*)

create duplicate of self, delete a line in new acl for given line number/attributes

Parameters **attribs** (*dict*) – line attributes

Returns copy of ACL with attributes/line removed

Return type *ACL*

copy_and_insert(*line_no*, *attribs*)

create duplicate of self, insert a new acl line in new acl object, with provided attributes at given line number and return new updated object. existing object remains untouched.

Parameters

- **line_no** (*int*) – line number at which entry to be inserted
- **attribs** (*dict*) – line attributes

Returns copy of ACL with attributes/line insert

Return type *ACL*

del_str()

String representation of acl recoded deletions

Returns recorded acl changes (removals)

Return type str

delete(*attribs*, *stop=None*, *step=1*)

delete a line in acl: can be use with standard delete command as well, del(acl_name[n])

Parameters

- **attribs** (*int, dict*) – int->deletes an entry by line number, dict->delete entry which matches attribute
- **stop** (*int, optional*) – to delete a range of lines provide end sequence. Defaults to None.
- **step** (*int, optional*) – to delete line numbers in multiple of. Defaults to 1.

Returns delta change(s) for the deletion of entry

Return type str

difference(*obj*)

difference between self and another ACL object elements

Parameters **obj** (ACL) – another ACL object to compare differences

Returns difference between self and another ACL object elements

Return type dict

end_point_identifiers_pos = {0: 5, 1: 7, 2: 9}

exact(*item*)

check matching attributes in acl object, and return set of matching acl line numbers for exact matches item only

Parameters **item** (*dict*) – line attributes

Raises **Exception** – exact match process error

Returns set of matching acl line numbers (exact match)

Return type set

insert(*line_no*, *attribs*)

insert a line in acl: can be use with standard way as well, aclname[line_no] = attribs display warning message - MatchingEntryAlreadyexistAtLine, if a match already exist in acl

Parameters

- **line_no** (*int*) – line number at which entry to be inserted
- **attribs** (*dict*) – line attributes

Returns delta change(s) for the insertion of entry

Return type str

mandatory_item_values_for_str = ('acl_type', 'action', 'protocol', 'source', 'destination', 'ports', 'log_warning')

property max

property min

parse(*objs*)

parse access-list-lines-list and set _repr_dic objs requires for acl lines having object-group-names

Parameters **objs** (OBJS) – object of object-groups (OBJS)

same_elements(*obj*)

compare self for similar elements with provided another ACL object.

Parameters **obj** (ACL) – another ACL object to compare elements

Returns if self and provided ACL has same elements or not

Return type bool

property sequence

str()

String representation of full acl

Returns full acl

Return type str

class fwOper.acl.**ACLS**(*config_list*, *objs=None*)

Bases: *fwOper.fwObj.Plurals*

collection of ACL objects

Parameters **Plurals** (Plurals) – Inherits - group of items properties definitions

changes(*change*)

collate the delta changes recorded in all access-lists and provide delta for that change (“ADDS”, “REMOVALS”)

Parameters **change** (*str*) – type of change for which change output requested (“ADDS”, “REMOVALS”)

Returns delta changes

Return type str

set_acl_names()

sets available access-lists names in _repr_dic (key)

Returns _repr_dict

Return type dict

set_objects(*objs*)

sets access-lists (ACL)s in _repr_dic (value)

Parameters **objs** (OBJS) – object of dictionary of object-groups

fwOper.acl.**access_list_list**(*config_list*)

extracts access-lists from provided configuration list ie.config_list.

Parameters **config_list** (*list*) – configuration list

Returns access-lists lines in a list

Return type list

fwOper.acl.**dummy_group**(*source_grp*, *item*, *values*)

create a dummy object-group with provided items, by taking template as source group

Parameters

- **source_grp** (OBJ) – source group (will be a template to create new dummy group)
- **item** (*str*) – acl line attribte name (‘source’, ‘destination’, ‘ports’, ‘protocol’)
- **values** (*str, set, tuple, list*) – set of value(s)

Returns object-group object with provided item: values

Return type *OBJ*

fwOper.acl.**update_obj_grp_str**(*item*, *what*)

update the object group and host string in acl

Parameters

- **item** (*dict*) – acl line item
- **what** (*str*) – acl line attribte name (‘source’, ‘destination’, ‘ports’, ‘protocol’)

Returns string represenation of object group or host object in acl

Return type str

4.2 acg

class fwOper.acg.**OBJ**(*obj_grp_name*, *_hash*)

Bases: *fwOper.fwObj.Singulars*

Individual group object

Parameters **Singulars** (Singulars) – Inherits - individual object properties definitions

Raises

- **Exception** – IncorrectIteminItemType
- **Exception** – InvalidGroupMemberType
- **Exception** – NoValidCandidate

Returns a single object-group object

Return type *OBJ*

add(**arg*)

add_str(*header=True*)

del_str(*header=False*)

delete(**arg*)

property grp_details

object group details in dictionary (helpful in generating copy)

Returns object-group primary/basic details

Return type dict

has(*obj*)

returns object group if self contains provided object-group.

Parameters **obj** (OBJ) – object-group object to check within

Returns object-group object if self within it else None

Return type (*OBJ*, None, False)

over(*acls*)

returns dictionary of acls with acl/line/attribute if object group present in any acls

Parameters **acls** (ACLS) – dictionary of all acls (ACLS)

Returns dictionary of acls with acl/line/attribute

Return type dict

parse()

starts parsing object-group-config-lines-list and set extended variables of instance

set_instance_primary_details(*obj_grp_details*)

set primary variable details of instance

Parameters **obj_grp_details** (*dict*) – object-group primary/basic details (candidates-list, type, service-filter)

str()

String representation of full object-group

Returns object-group

Return type str

class fwOper.acg.**OBJS**(*config_list*)

Bases: *fwOper.fwObj.Plurals*

collection of object groups

Parameters **Plurals** (Plurals) – Inherits - group of items properties definitions

changes(*change*)

collate the delta changes recorded in all object-groups and provide delta for that change ("ADDS", "REMOVALS")

Parameters **change** (*str*) – type of change for which change output requested ("ADDS", "REMOVALS")

Returns delta changes

Return type str

get_matching_obj_grps(*requests*)

matches provided (request members) in all object-groups available on device and returns dictionary of object-group names, where object-group matches same members in it.

Parameters **requests** (*tuple, list, set*) – list/set/tuple with members of dict, containing 'source', 'destination', 'ports' as keys.

Raises **Exception** – Invalid Request type

Returns include all three, src, dest, port

Return type dict

matching_obj_grps(*member*)

matches provided [members] in all object-groups available on device and returns list of object-group names, where object-group matches same members in it.

Parameters **member** (*list, set, tuple*) – list/set/tuple with members

Returns singular object

Return type list

set_objects()

set extended information of each object-group.

fwOper.acg.**get_member_obj**(*member_type*, *member*, *objs*)

convert and provided string member to member object aka: Network, OBJ, Ports based on its member-type provided. objs: requires for recursive lookup for OBJ (if any)

Parameters

- **member_type** (*str*) – type of member
- **member** (*str*) – string repr of Network, OBJ, Ports etc
- **objs** (OBJS) – collection of Object-groups (OBJS object)

Raises **Exception** – InvalidMemberType

Returns Based on member type returns member object

Return type [*Network*, *Ports*, None]

4.3 instances

class fwOper.instances.**Instance**(*instance_name*, *instance_config_list*)

Bases: *fwOper.fwObj.Singulars*

a firewall instance object

Parameters **Singulars** (Singulars) – inherits properties/methods for Singulars objects

parse()

parsing thru instance configuration

str()

details of current instance with keys

Returns current instance keys

Return type str

class fwOper.instances.**Instances**(*config_list*)

Bases: *fwOper.fwObj.Plurals*

firewall instances object

Parameters **Plurals** (Plurals) – inherits properties/methods for Plural objects

changes()

to be implemented [TBD]

set_objects()

sets all individual instances

4.4 route

class fwOper.route.**ROUTE**(*route_line*)

Bases: *fwOper.fwObj.Singulars*

Individual static-route object,

Properties: (network, nexthop, ifdesc, distance)

Parameters **Singulars** (Singulars) – inherits Singulars object properties/methods

parse()

parse static route line and set route_dict

str()

return String representation of routes

class fwOper.route.**ROUTES**(*config_list*)

Bases: object

collection of object of Routes

get_route_objects()

set ROUTE objects in self-Routes instance

prefix(*network*)

check matching network in ROUTES object, return longest matching route

Parameters **network** (*str*) – ip-address/subnet

Returns matching Route object (longest match)

Return type Route

str()

string representation of self

Returns all routes

Return type str

fwOper.route.**routes_list**(*config_list*)

list of lines with static routes from given config-list

Parameters **config_list** (*list*) – firewall (instance) configuration list

Returns routes

Return type list

4.5 common

fwOper.common.**heading**(*what*, *name*, *change*)

used to get the Banner heading

Parameters

- **what** (*str*) – banner require for what? (valid options = acl, object-group)
- **name** (*str*) – filter on valid options (acl/object-group name)
- **change** (*str*) – filter on change type (valid options = adds, removals)

Returns banner for the provided requirements

Return type str

4.6 entity

class fwOper.entity.**ACL_REMARK**(*remark*)

Bases: object

ACL remark entity object

str()

class fwOper.entity.**EntiryProperties**

Bases: object

Common properties/methods for individual entities

fwOper.entity.**IcmpProtocol**

alias of *fwOper.entity.Singular*

class fwOper.entity.**Network**(*network*, *dotted_mask=None*)

Bases: *fwOper.entity.EntiryProperties*

a network/subnet object

Parameters **EntiryProperties** (EntiryProperties) – Common properties/methods for individual entities

address_it()

set addressing object and a few basic variables.

fwOper.entity.**NetworkProtocol**

alias of *fwOper.entity.Singular*

class fwOper.entity.**Ports**(*port_type*, *port*, *port_range_end=''*, *objectGroups=None*)

Bases: *fwOper.entity.EntiryProperties*

a port/range-of-ports object

Parameters **EntiryProperties** (EntiryProperties) – Common properties/methods for individual entities

split()

split the port string to a list

Returns port string splitted

Return type list

class fwOper.entity.**Singular**(*_type*)

Bases: *fwOper.entity.EntiryProperties*

a common class template to create an IcmpProtocol or NetworkProtocol object instance

Parameters **EntiryProperties** (EntiryProperties) – Common properties/methods for individual entities

4.7 fwObj

class fwOper.fwObj.**Common**

Bases: object

Commons properties/methods for Singular/Plural objects

keys()

values()

class fwOper.fwObj.**Plurals**

Bases: *fwOper.fwObj.Common*

collection of objects

Parameters **Common** (Common) – Inherits Commons properties/methods for Singular/Plural objects

changes(*what*, *change*)

collate the recorded delta changes and provide delta for that change ("ADDS", "REMOVALS")

Parameters

- **what** (*str*) – where to look for the change ('acl', 'object-group')
- **change** (*str*) – type of change for which change output requested ("ADDS", "REMOVALS")

Raises **Exception** – INCORRECTCHANGE

Returns delta changes

Return type str

abstract classmethod set_objects()

class fwOper.fwObj.**Singulars**(*name=''*)

Bases: *fwOper.fwObj.Common*

a single object

Parameters **Common** (Common) – Inherits Commons properties/methods for Singular/Plural objects

abstract classmethod parse()

fwOper.fwObj.**get_object**(*obj*, *file=None*, *conf_list=None*, ***kwargs*)

Pre-defined set of steps to get objects. (either input require file/conf_list ; preferred conf_list)

Parameters

- **obj** (OBJS, ACLS, ROUTES, *INSTANCES*) – various objects type
- **file** (*str, optional*) – file name with path. Defaults to None.
- **conf_list** (*list, optional*) – configuration content in list format. Defaults to None.

Raises **Exception** – MissingMandatoryInput

Returns object

Return type object

4.8 member

fwOper.member.**get_match_dict**(*request_parameters*, *objs*)

search for request parameters and return matching parameters dictionary. (dictionary with attributes require to match in ACL)

Parameters

- **request_parameters** (*dict*) – request paramters in dictionary
- **objs** (OBJS) – object-groups object

Returns with filtered parameters only

Return type dict

fwOper.member.**get_port_name**(*n*)

update and return well known port number for port name

Parameters **n** (*int*) – port number

Returns well-known port name else port number

Return type str

fwOper.member.**group_object_member**(*spl_line*, *objectGroups=None*)

returns object-group object from given splitted line

Parameters

- **spl_line** (*[type]*) – [description]
- **objectGroups** (*[type], optional*) – [description]. Defaults to None.

Returns object-group OBJ member object

Return type *OBJ*

fwOper.member.**icmp_group_member**(*spl_line*)

returns icmp port group member object from given splitted line

Parameters **spl_line** (*list*) – splitted line of an acl entry

Returns IcmpProtocol member object

Return type IcmpProtocol

fwOper.member.**network_group_member**(*spl_line*, *idx*, *objectGroups=None*)

returns Network group member object from given splitted line

Parameters

- **spl_line** (*list*) – splitted line of an acl entry
- **idx** (*int*) – index position to start looking for network
- **objectGroups** (OBJS, *optional*) – object-groups object. Defaults to None.

Raises **Exception** – UndefinedEndPointType

Returns Network group member object

Return type (*Network*, *OBJ*, None)

fwOper.member.**network_member**(*network*, *objs=None*)

returns Network group member object for given network, objs will require if network has object-group.

Parameters

- **network** (*str*) – ip-network string
- **objs** (OBJS, *optional*) – object-groups object. Defaults to None.

Raises **Exception** – InvalidNetwork

Returns Network group member object

Return type *Network*

fwOper.member.**port_group_member**(*spl_line*, *idx*, *objectGroups=None*)

returns Port group member object from given splitted line

Parameters

- **spl_line** (*list*) – splitted line of an acl entry
- **idx** (*int*) – index position to start looking for port(s)
- **objectGroups** (OBJS, *optional*) – object-groups object. Defaults to None.

Raises **Exception** – UndefinedPort/PortType

Returns Network group member object

Return type (*Ports*, *OBJ*, None)

fwOper.member.**port_member**(*port*, *objs*)

returns Port group member object for given port, objs will require if port has object-group

Parameters

- **port** (*str*) – port string
- **objs** (OBJS) – object-groups object

Raises **Exception** – InvalidPort

Returns Ports member object

Return type *Ports*

fwOper.member.**protocol_group_member**(*spl_line*)

returns protocol group member object from given splitted line

Parameters **spl_line** (*list*) – splitted line of an acl entry

Returns NetworkProtocol member object

Return type NetworkProtocol

fwOper.member.**update_ports_name**(*requests*)

update and return well known port number for port name in given request port

Parameters **requests** (*dict*) – acl attributes in request dictionary

Returns updated request attributes

Return type dict

4.9 static

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

f

INDEX

M

N

O

P

R

S

U

V