contrail-api-cli-extra Documentation

Release 0.5.9

Jean-Philippe Braun, Antoine Eiche, Edouard Thuleau, Mathieu Tu

Mar 19, 2019

Contents

CHAPTER 1

contrail_api_cli_extra package

This package contains contrail-api-cli commands to ease operating/fixing a contrail installation.

Commands are grouped in different packages with different purposes:

- *contrail_api_cli_extra.clean*: commands to detect and remove bad resources
- *contrail_api_cli_extra.fix*: commands to detect and fix bad resources
- *contrail_api_cli_extra.migration*: commands to handle data migration when upgrading contrail to a new major version
- *contrail_api_cli_extra.misc*: general purpose commands
- *contrail_api_cli_extra.provision*: commands used to provision/configure a contrail installation

1.1 Installation

In the contrail-api-cli-extra directory run:

```
python setup.py install
```

If all goes well you will see new commands in contrail-api-cli:

```
contrail-api-cli --host 1.2.3.4 shell
1.2.3.4:/> help
Available commands: schema shell exec edit tree cat relative ln kv rm python du
ls find-orphaned-projects fix-subnets fix-vn-id fix-fip-locks reschedule-vm fix-sg
rpf provision dot exit help cd
```

1.2 Subpackages

1.2.1 contrail_api_cli_extra.clean package

Module containing commands to clean bad resources.

Since these commands are sensitive as they will probably remove resources the `contrail_api_cli.clean` namespace must be loaded explicitely to be able to run the commands:

```
contrail-api-cli --ns contrail_api_cli.clean <command>
```

contrail_api_cli_extra.clean.orphaned_acl module

class contrail_api_cli_extra.clean.orphaned_acl.**OrphanedACL**(*name*)
Bases: contrail_api_cli.command.Command

Removes stale ACLs.

ACL is considered as stale if it has no parent:

```
contrail-api-cli --ns contrail_api_cli.ns clean-orphaned-acl --cassandra-servers
↪<ip1> <ip2>
```

Note: Because of an API server limitation the ACLs are removed directly from the cassandra cluster. Thus, the cassandra cluster nodes IPs must be provided.

contrail_api_cli_extra.clean.project module

class contrail_api_cli_extra.clean.project.**FindOrphanedProjects**(*name*)
Bases: contrail_api_cli.command.Command

Command to find projects that are still in contrail but no more in keystone.

Run:

```
contrail-api-cli find-orphaned-projects
```

class contrail_api_cli_extra.clean.project.**PurgeProject**(*name*)
Bases: *contrail_api_cli_extra.utils.ConfirmCommand*

Command to purge a project. All related resources are deleted.

Warning: This command is experimental and not fully tested.

This command works recursively by first trying to remove the project. If other resources are linked to the project the API will return a 409 response with all linked resources. The command will then try to delete these resources and so on until the project resource can be deleted.

Because of this no dry-run mode is possible.

To run the command:

```
contrail-api-cli --ns contrail_api_cli.clean purge-project project/uuid
```

contrail_api_cli_extra.clean.refs module

class contrail_api_cli_extra.clean.refs.**CleanRefs**(*name*)

Bases: *contrail_api_cli_extra.utils.CheckCommand*

Clean references in Contrail DB.

Broken refs can be found with gremlin.

The command expects a list of ids where the first ID is a valid resource while the second is a missing resource still referenced.

For example:

```
> g.V().hasLabel('routing_instance').not(has('_missing')).in().hasLabel('virtual_
↪machine_interface').has('_missing').path().by(id).unfold().fold()
[ri_id, vmi_id, ri_id, vmi_id, ...]
```

This return a list of broken refs between RIs and VMIs where VMIs don't exists or are incomplete.

We can clean then by running:

```
contrail-api-cli --ns contrail_api_cli.clean clean-refs --ref-type backref --
↪target-type virtual_machine_interface ri_id vmi_id ri_id vmi_id ...
```

Or directly from a file:

```
contrail-api-cli --ns contrail_api_cli.clean clean-refs --ref-type backref --
↪target-type virtual_machine_interface --resources-file file
```

Other examples:

```
# RIs without any parent VN
> g.V().hasLabel('routing_instance').not(has('_missing')).out('parent').hasLabel(
↪'virtual_network').has('_missing').path().by(id).unfold().fold()
> contrail-api-cli clean-refs --ref-type parent --target-type virtual_network ...

# ACLs without any SG
> g.V().hasLabel('access_control_list').not(has('_missing')).out('parent').
↪hasLabel('security_group').has('_missing').path().by(id).unfold().fold()
> contrail-api-cli clean-refs --ref-type parent --target-type security_group ...
```

contrail_api_cli_extra.clean.rt module

class contrail_api_cli_extra.clean.rt.**CleanRT**(*name*)

Bases: *contrail_api_cli_extra.utils.CheckCommand*, *contrail_api_cli_extra.utils.ZKCommand*, *contrail_api_cli_extra.utils.PathCommand*

Removes stale route-targets.

RTs that are not linked to a logical-router or a routing-instance are considered as staled and will be removed. If a ZK lock exists for the RT it will be removed:

```
contrail-api-cli --ns contrail_api_cli.clean clean-route-target --zk-server <ip>␣
↪[route-target/uuid]
```

If no route-target path is provided all RTs are considered. `--check` and `--dry-run` options are available.

You can exclude RT from the cleaning process:

```
contrail-api-cli --ns contrail_api_cli.clean clean-route-target --exclude <RT_
↪FQNAME> --exclude <RT_FQNAME> [...]
```

contrail_api_cli_extra.clean.si module

class contrail_api_cli_extra.clean.si.**CleanSIScheduling**(*name*)

Bases: *contrail_api_cli_extra.utils.CheckCommand*, *contrail_api_cli_extra.utils.PathCommand*

On some occasion a SI VM can be scheduled on multiple virtual-routers.

In such case the command will remove extraenous VR links on the SI VM:

```
contrail-api-cli --ns contrail_api_cli.clean clean-si-scheduling [service-
↪instance/uuid]
```

`--check` and `--dry-run` options are available.

class contrail_api_cli_extra.clean.si.**CleanStaleSI**(*name*)

Bases: *contrail_api_cli_extra.utils.CheckCommand*, *contrail_api_cli_extra.utils.PathCommand*

Clean stale service instances.

SIs are considered stale when:

- LBaaS SI has no pool
- LBaaS SI pool has no VIP
- LBaas SI VIP has no instance-ip
- SNAT SI has no logical-router

To run the command:

```
contrail-api-cli --ns contrail_api_cli.ns clean-stale-si [service-instance/uuid]
```

`--check` and `--dry-run` options are available.

contrail_api_cli_extra.clean.subnet module

class contrail_api_cli_extra.clean.subnet.**CleanSubnet**(*name*)

Bases: *contrail_api_cli_extra.utils.CheckCommand*

Command to fix subnets KV store.

This command works by first retrieving all the KV store, then it builds a dict which tends to reproduce a clean KV store based on the virtual-network back-refs objects present in the API. Finally it compares the clean KV store with the current one and retrieves the keys of the stale entries to remove it in the current KV store.

To run the command:

```
contrail-api-cli --ns contrail_api_cli.clean clean-subnet
```

1.2.2 contrail_api_cli_extra.fix package

Module containing commands to fix bad resources.

contrail_api_cli_extra.fix.fix_fip_locks module

contrail_api_cli_extra.fix.fix_sg module

class contrail_api_cli_extra.fix.fix_sg.**FixSg**(*name*)

Bases: *contrail_api_cli_extra.utils.CheckCommand*, *contrail_api_cli_extra.utils.PathCommand*, *contrail_api_cli_extra.utils.ConfirmCommand*

Fix multiple default security groups on projects.

It appears sometimes several default security groups have been created. Normally, only one default security group should be created. When there are several security groups, some of them doesn't have the right project name in their fq_name. These security groups are not legitimate.

The check mode of this script detects theses security groups and marks default security groups of a tenant as good or bad. Good default security group is legitimate while bad are not. Moreover, if it exists bad security groups, the scripts returns 1 otherwise, it returns 0.

Concerning normal mode and dry-run mode, the script tries to delete non used bad default security groups. "Non used" means no VMIs are attached to them.

To run the command:

```
contrail-api-cli fix-sg [project/uuid]
```

If no project path is provided all projects are considered.

contrail_api_cli_extra.fix.fix_subnets module

class contrail_api_cli_extra.fix.fix_subnets.**FixSubnets**(*name*)

Bases: *contrail_api_cli_extra.utils.CheckCommand*

Fix subnet/vn association in kv store.

When the API server is not properly started the hooks that populates the kv store on subnet creation are not properly run. As a result doing neutron net-list will lead to the following error:

```
404-{u'NeutronError': {u'message': u'Subnet 646f986a-67c9-4e1b-bf13-59f18f787068␣
↪could not be found', u'type': u'SubnetNotFound', u'detail': u''}}
```

This command check all the IPAM subnet informations and verifies that proper kv store keys exists for each subnet.

This command assumes there is only one IPAM in the contrail installation (`default-domain:default-project:default-network-ipam`).

To check all subnets run:

```
contrail-api-cli fix-subnets --check
```

To fix all subnets run:

```
contrail-api-cli fix-subnets [--dry-run]
```

Or to fix a particular subnet run:

```
contrail-api-cli fix-subnets <subnet_uuid> [--dry-run]
```

The version of the schema is required for this command:

```
contrail-api-cli --schema-version 3.2 fix-subnets
```

contrail_api_cli_extra.fix.fix_vn_id module

class contrail_api_cli_extra.fix.fix_vn_id.**FixVnId**(*name*)

Bases: *contrail_api_cli_extra.utils.PathCommand*, *contrail_api_cli_extra.utils.ZKCommand*, *contrail_api_cli_extra.utils.CheckCommand*, *contrail_api_cli_extra.utils.ConfirmCommand*

Compare and fix virtual network IDs in Zookeeper and the API server.

Checks that the ZK lock for a VN has the correct index. Checks also that the VN has a lock in ZK.

To check all VNs run:

```
contrail-api-cli fix-vn-id --check
```

To fix all VNs or a particular VN run:

```
contrail-api-cli fix-vn-id [--dry-run] [vn_uuid]
```

contrail_api_cli_extra.fix.fix_zk_ip module

class contrail_api_cli_extra.fix.fix_zk_ip.**FixZkIP**(*name*)

Bases: *contrail_api_cli_extra.utils.ZKCommand*, *contrail_api_cli_extra.utils.CheckCommand*, *contrail_api_cli_extra.utils.PathCommand*, *contrail_api_cli_extra.utils.ConfirmCommand*

Remove or add ZK locks based on the IPAM configuration.

Sometimes, when an instance-ip or a floating-ip is created or deleted, its associated zookeeper node isn't managed properly.

This led to situation where, no IPs are reserved from the Contrail API standpoint and nevertheless, you are not able to get one, or the same floating IP address is reserved for several users/tenants.

This command list all zookeeper nodes for a given network (which may contain one or several subnet) and compare the nodes with the IPs found with the contrail API. For each IP found in zookeeper and not in the contrail API (abusive lock scenario), the command delete the associated zookeeper node. Then, for each IP found in API, and not in Zookeeper, the command creates the appropriate lock.

Usage:

```
contrail-api-cli fix-zk-ip --zk-server <IP> [path/to/virtual-network] [--dry-run]
```

If no virtual-network is given, all virtual-networks are considered.

class contrail_api_cli_extra.fix.fix_zk_ip.**Subnet**(*cidr*, *gateway*, *dns*)

Bases: `tuple`

cidr

Alias for field number 0

dns
Alias for field number 2

gateway
Alias for field number 1

exception contrail_api_cli_extra.fix.fix_zk_ip.**SubnetNotFound**
Bases: exceptions.Exception

exception contrail_api_cli_extra.fix.fix_zk_ip.**UnhandledResourceType**
Bases: exceptions.Exception

exception contrail_api_cli_extra.fix.fix_zk_ip.**ZkNodeNotFound**(*zk_subnets*)
Bases: exceptions.Exception

contrail_api_cli_extra.fix.ri module

class contrail_api_cli_extra.fix.ri.**FixRI**(*name*)
Bases: *contrail_api_cli_extra.utils.CheckCommand*, *contrail_api_cli_extra.utils.ZKCommand*, *contrail_api_cli_extra.utils.PathCommand*

Fix routing-instances without route-targets

contrail-api-cli fix-ri –zk-server <ip> [routing-instance/uuid]

1.2.3 contrail_api_cli_extra.migration package

contrail_api_cli_extra.migration.host_routes module

class contrail_api_cli_extra.migration.host_routes.**MigrateHostRoutes**(*name*)
Bases: *contrail_api_cli_extra.utils.CheckCommand*, *contrail_api_cli_extra.utils.PathCommand*

Command to migrate host-route implementation based on interface-route-tables to route-tables (V3 plugin):

```
contrail-api-cli --ns contrail_api_cli.migration migrate-host-routes [interface-
↪route-table/uuid]
```

contrail_api_cli_extra.migration.lb module

class contrail_api_cli_extra.migration.lb.**MigrateLB22132**(*name*)
Bases: contrail_api_cli.command.Command

Command to migrate LB created in 2.21 to 3.2.

In 2.21 the admin_state flag didn't matter, but in 3.2 if it is not set to True the haproxy config file will not be generated.

This command set admin_state to True for all lb-pools, lb-members, lb-vips, lb-hms.

contrail_api_cli_extra.migration.rt module

class contrail_api_cli_extra.migration.rt.**MigrateRT22132**(*name*)
Bases: *contrail_api_cli_extra.utils.CheckCommand*, *contrail_api_cli_extra.utils.PathCommand*

Command to migrate custom RTs from 2.21 to 3.2.

In 3.2 we can benefit from new VN properties that allows to define the list of RTs the VNs must import/export.

contrail_api_cli_extra.migration.si module

class contrail_api_cli_extra.migration.si.**MigrateSI110221**(*name*)
Bases: *contrail_api_cli_extra.utils.CheckCommand*

Migration command for SI from 1.10 version to 2.21 version

1.2.4 contrail_api_cli_extra.misc package

General purpose commands.

contrail_api_cli_extra.misc.check_bad_refs module

class contrail_api_cli_extra.misc.check_bad_refs.**CheckBadRefs**(*name*)
Bases: *contrail_api_cli_extra.utils.CheckCommand*

Check for broken references.

The command will read all objects from the cassandra DB then for each object check that references exists in the API.

References includes refs, back refs, children, parents.

To run the command:

contrail-api-cli check-bad-refs –cassandra-servers db:9160 [uuids...]

class contrail_api_cli_extra.misc.check_bad_refs.**PropertiesEncoder**(*skipkeys=False*, *ensure_ascii=True*, *check_circular=True*, *allow_nan=True*, *sort_keys=False*, *indent=None*, *separators=None*, *encoding='utf-8'*, *default=None*)
Bases: json.encoder.JSONEncoder

encode(*o*)
Return a JSON string representation of a Python data structure.

```
>>> JSONEncoder().encode({"foo": ["bar", "baz"]})
'{"foo": ["bar", "baz"]}'
```

contrail_api_cli_extra.misc.dot module

class contrail_api_cli_extra.misc.dot.**Dot**(*name*)
Bases: contrail_api_cli.command.Command

Command to create a dot file from a list of paths.

The command will automatically add to the graph the parent, refs and back_refs of the given resources:

```
contrail-api-cli dot path/to/res1 path/to/res2 -f /tmp/output.dot
```

-e option can be used to exclude resources from the graph.

contrail_api_cli_extra.misc.graph module

contrail_api_cli_extra.misc.manage_rt module

class contrail_api_cli_extra.misc.manage_rt.**ManageRT**(*name*)
Bases: contrail_api_cli.command.Command

Command to manage VN custom RTs

contrail_api_cli_extra.misc.rpf module

class contrail_api_cli_extra.misc.rpf.**RPF**(*name*)
Bases: *contrail_api_cli_extra.utils.PathCommand*

Simple command to enable or disable RPF (Reverse Path Forwarding) on a VN.

To check if RPF is enabled or not run:

```
contrail-api-cli rpf virtual-network/uuid
```

To enable of disable RPF add `--on` or `--off` options.

contrail_api_cli_extra.misc.vm module

class contrail_api_cli_extra.misc.vm.**RescheduleVM**(*name*)
Bases: *contrail_api_cli_extra.utils.CheckCommand*

Command to move a SI VMs from one virtual-router to other virtual-routers.

This command will move all SI VMs found on the provided VR to a list of VRs. The command will make sure two VMs of the same SI will not end up on the same VR.

The command can be run in dry-run mode to see how the VMs will be moved.

To reschedule VMs run:

```
contrail-api-cli reschedule-vm virtual-router/uuid1 virtual-router/uuid2 virtual-
↪router/uuid3
```

This will move all VR/uuid1 SI VMs to VR/uuid2 and VR/uuid3 evenly.

1.2.5 contrail_api_cli_extra.provision package

A set of commands to provision/bootstrap a contrail environment.

Each command can be used separately but the more general case is to use the `provision` command that will call appropriate commands given an environment specification file.

See *contrail_api_cli_extra.provision.provision*

Only the *provision* command is available in the default namespace. If you wish to use other commands from the provision module use:

```
contrail-api-cli --ns contrail_api_cli.provision list-bgp-router
```

contrail_api_cli_extra.provision.analytics module

contrail_api_cli_extra.provision.bgp_router module

contrail_api_cli_extra.provision.common module

contrail_api_cli_extra.provision.config module

contrail_api_cli_extra.provision.dns_nameserver module

contrail_api_cli_extra.provision.encapsulation module

contrail_api_cli_extra.provision.global_asn module

contrail_api_cli_extra.provision.linklocal module

contrail_api_cli_extra.provision.lr module

contrail_api_cli_extra.provision.provision module

Given a JSON environment file provison the needed resources in contrail.

The provision command works in two steps. First it will collect informations of the running environment and the wanted environment provided in the JSON file. Then it will make a diff of the two environments and add/remove/modify resources in the running environment.

Anatomy of the environment file

The base structure of the file is as follow:

```
{
    "name": "environ",
    "namespace": "contrail_api_cli.provision",
    "defaults": {},
    "provision": {}
}
```

- `name` is the name of the environment you want to provision
- `namespace` is the command namespace to load to do the provisioning

- `defaults` is an object containing default values used for provisioning but also for collecting data of the running environment
- `provision` is an object defining the resources that needs to be provisioned in the running environment

provision

In the provision section you can defined the resource you need to provision on the installation.

Each key correspond to a provision command. The content is a list of calls to the command. Example:

```
"provision": {
    "bgp-router": [
        {
            "router-asn": 64512,
            "router-ip": "10.0.0.1",
            "router-address-families": [
                "route-target",
                "inet-vpn"
            ]
        }
    ]
}
```

In this case the provision command will call the *add-bgp-router* command if the bgp-router is not present on the installation.

defaults

The *defaults* section is similar to the *provision* section exept that keys don't take a list but only an object with the default parameters:

```
"defaults": {
    "dns-nameserver": {
      "network-ipam-fqname": "default-domain:default-project:default-network-ipam"
    }
}
"provision": {
    "dns-nameserver": [
        {"ips": ["185.23.93.178", "185.23.93.179"]}
    ]
}
```

In this case the *network-ipam-fqname* will be used while reading the current list of dns nameservers and also when provisioning the list of nameservers.

Provisioning examples

virtual-routers:

```
"provision": {
    "vrouter": [
        {
            "vrouter-ip": 1.2.3.4,
```

(continues on next page)

(continued from previous page)

```
            "vrouter-name": "vrouter1"
        },
        {
            "vrouter-ip": 1.2.3.5,
            "vrouter-name": "vrouter2"
        }
    ]
}
```

bgp-routers:

```
"defaults": {
    "bgp-router": {
        "router-address-families": [
            "route-target",
            "inet-vpn",
            "erm-vpn"
        ]
    }
},
"provision": {
    "bgp-router": [
        {
            "router-asn": 60940,
            "router-ip": "10.2.0.1",
            "router-type": "juniper",
            "router-name": "mx",
            "router-address-families": [
                "route-target",
                "inet-vpn"
            ]
        },
        {
            "router-ip": "10.0.0.2",
            "router-asn": 64518,
            "router-name": "control1"
        },
        {
            "router-ip": "10.0.0.3",
            "router-asn": 64518,
            "router-name": "control2"
        }
    ]
}
```

Note: The two contrail controllers will use the address families listed in the *defaults* section. The mx one will use a different list.

dns nameservers:

```
"defaults": {
    "dns-nameserver": {
        "network-ipam-fqname": "default-domain:default-project:default-network-ipam"
    }
},
```

(continues on next page)

(continued from previous page)

```
"provision": {
    "dns-nameserver": {
        "ips": [
            "185.23.93.178",
            "185.23.93.179"
        ]
    }
}
```

Note: When only one item has to be provisioned you can pass directly the object instead of a list of objects.

encapsulation order:

```
"provision": {
    "encaps": {
        "modes": [
            "MPLSoUDP",
            "MPLSoGRE",
            "VXLAN"
        ]
    }
}
```

global ASN:

```
"provision": {
    "global-asn": {
        "asn": 64518
    }
}
```

linklocal:

```
"provision": {
    "linklocal": {
        "fabric-dns-service-name": "metadata.nova.example.com",
        "fabric-service-port": 8775,
        "service-name": "metadata",
        "service-ip": "169.254.169.254",
        "service-port": 80
    },
}
```

virtual-networks:

```
"defaults": {
    "vn": {
        "project-fqname": "default-domain:openstack"
    }
},
"provision": {
    "vn": {
        "virtual-network-name": "public",
        "external": true
```

(continues on next page)

(continued from previous page)

```
    }
}
```

subnets:

```
"defaults": {
    "subnets": {
        "virtual-network-fqnames": [
            "default-domain:openstack:public"
        ]
    }
},
"provision": {
    "subnets": {
        "cidrs": [
            "67.45.0.144/28",
            "67.45.1.0/25"
        ],
        "virtual-network-fqname": "default-domain:openstack:public"
    }
}
```

Note: Order matters. When you bootstrap the environment the virtual-network must be provisioned before the subnets.

route-targets:

```
"defaults": {
    "route-targets": {
        "virtual-network-fqname": "default-domain:openstack:public"
    }
}
"provision": {
    "route-targets": {
        "route-target-list": [
            "target:64518:1",
            "target:234.123.34.0:1",
        ]
    }
}
```

Running the provisioning

Once the JSON file is created you can validate the syntax with jq:

```
cat spec.json | jq .
```

Then you can run the provision command:

```
contrail-api-cli provision spec.json
```

The command will show the actions it will make and waits the input of the user to actually make the actions.

class contrail_api_cli_extra.provision.provision.**Provision**(*name*)
Bases: contrail_api_cli.command.Command

contrail_api_cli_extra.provision.route_target module

contrail_api_cli_extra.provision.service_appliance_set module

contrail_api_cli_extra.provision.subnet module

contrail_api_cli_extra.provision.vn module

contrail_api_cli_extra.provision.vrouter module

1.3 contrail_api_cli_extra.utils module

Shared functions and classes for contrail_api_cli_extra commands.

class contrail_api_cli_extra.utils.**CassandraCommand**(*name*)
Bases: contrail_api_cli.command.Command

Inherit from this class to add *–cassandra-servers* options.

Cassandra servers list value is stored in *self.cassandra_servers*.

class contrail_api_cli_extra.utils.**CheckCommand**(*name*)
Bases: contrail_api_cli.command.Command

Inherit from this class to add *–check* and *–dry-run* options.

Options values are stored in *self.check* and *self.dry_run* (True or False).

class contrail_api_cli_extra.utils.**ConfirmCommand**(*name*)
Bases: contrail_api_cli.command.Command

Inherit from this class for a command that expect a confirmation.

This will add a –yes flag to skip the confirmation. *self.yes* can be used by the command to show the confirmation or not.

confirm_message
Confirmation message to run the command

class contrail_api_cli_extra.utils.**PathCommand**(*name*)
Bases: contrail_api_cli.command.Command

Inherit from this class for a command that expect a list of resource paths.

This will add a *path* argument to the command (nargs=*).

The selected resources are available in *self.resources*.

resource_type
Type of resource the command is about.

class contrail_api_cli_extra.utils.**RouteTargetAction**(*option_strings*, *dest*, *nargs=None*, *const=None*, *default=None*, *type=None*, *choices=None*, *required=False*, *help=None*, *metavar=None*)
Bases: argparse.Action

class contrail_api_cli_extra.utils.**ZKCommand**(*name*)
Bases: contrail_api_cli.command.Command

Inherit from this class when a connection to the Zookeeper cluster is needed.

This will add a *–zk-server* option to the command.

The ZK client is available in *self.zk_client*.

contrail_api_cli_extra.utils.**format_table_ascii_delimiters**(*header*, *widths*, *data*)
Return ascii table with delimitation and lines wrapped.

usage:

```
header = ["name", "age", "job", "bag"]
widths = [20, 4, 10, 40]
data = [
         ["foo", 54, "policeman", ("pen", "knife", "glasses")],
         ["bar", 28, "fireman", ("socket", "hat")],
         ["joe", 36, "anthropologist", ("computer", "keyboard")]
       ]

format_table2(header, widths, data)
```

return:

```
+------+-----+------------+------------+
| name | age | job        | bag        |
+------+-----+------------+------------+
| foo  | 54  | policeman  | - pen      |
|      |     |            | - knife    |
|      |     |            | - glasses  |
|      |     |            |            |
| bar  | 28  | fireman    | - socket   |
|      |     |            | - hat      |
|      |     |            |            |
| joe  | 36  | anthropolo | - computer |
|      |     | gist       | - keyboard |
|      |     |            |            |
+------+-----+------------+------------+
```

contrail_api_cli_extra.utils.**ip_type**(*string*)
argparse type to validate IP adresses.

contrail_api_cli_extra.utils.**md5_type**(*value*)
argparse type to validate a md5 hash.

contrail_api_cli_extra.utils.**network_type**(*string*)
argparse type to validate network adresses.

contrail_api_cli_extra.utils.**port_type**(*value*)
argparse type to validate port numbers.

contrail_api_cli_extra.utils.**server_type**(*value*)
argparse type to validate a server:port option.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

C

Index

C

D

E

F

G

I

M

N

O

P

R

S

U

Z