Canned Beef

What is Canned Beef?

"Canned beef" is a jargon term, meaning SA Script session, canned in form of portable container along with script input and result. All CLI and SNMP interactions of the script are stored in the can.

Canned beef can be unpacked by NOC testing suite and can be used to reproduce the script session, even if the original box is absent. Canned beef used for testing and debugging issues, allowing NOC to presume personality of different software versions and hardware platforms, making sure no future changes broke an existing behavior.

Second use of the beef, is to provide NOC developers access to the personality of the box, not giving shell access to it.

Third, just like a real world beef, NOC's canned beef can be conserved from completely dead meat. Failed script session can be still canned, allowing developers to reproduce lifetime of the session and find an exact reason of failure

What is Canned Beef Repo

Beefs are organized in the separate mercurial repos. The number of repos are available. Repo can be marked as either public or private. Public repos contains beef which store no sensitive information and can be made publically available. In contrary, private repos can contain sensitive information and designated to use in closed environment.

Beef repos are configured in *etc/beef.conf*. Each repo is a separate config section like

```
[sa_public]
enabled = true
type = sa
private = false
repo = https://bitbucket.org/nocproject/noc-sa-beef-public

[my_repo]
enabled = true
type = sa
private = true
repo = ssh://hg@bitbucket.org/nocproject/my_repo
```

Where:

- sa_public - repo name
- *enabled* - true or false. To temporary disable beef repo mark it as *enabled = false*
- *type* - Always *sa*
- *private* - true or false. Private repos denoted as *private = true*
- *repo* - Mercurial repo path

To check current repo status:

```
/opt/noc$ ./noc beef --list-repo
E P T  Name                 Repo
+ N sa sa_public            https://bitbucket.org/nocproject/noc-sa-beef-public
+ Y sa my_repo              ssh://hg@bitbucket.org/nocproject/my_repo
```

Where:

- *E* - Status of repo
- *P* - Repo is private or not
- *T* - Type of repo
- *Name* - Name of repo
- *Repo* - Mercurial repo path

NOC project maintains public repo (*sa_public*) one and separate private repos per *subscriber*. You can create additional repos to support your customers and local installations.

How to prepare a Canned Beef?

Acting as *noc* user:

```
/opt/noc$ ./noc script --debug <script name> <access url> -o <output>
```

where *script name* is a common script name, like *Cisco.IOS.get_version* or *get_version*. *access url* can be one of:

- path to JSON file with credentials
- managed object name
- managed object id

resulting beef can be stored in *output*.

Examples:

Run script without any parameters, when: user name "foo", user password "bar", and enable password "superpassword":

```
/opt/noc$ ./noc script --debug Cisco.IOS.get_version cred.json -o output.json
```

Where *cred.json* contain:

```
{
   "scheme": "ssh",
   "address": "192.168.0.2",
   "port": 1022,
   "credentials": {
       "user": "foo", "password": "bar", "super_password": "superpassword"
   }
}
```

Run script for existing managed object *router1*

```
 /opt/noc$ ./noc script --debug get_version router1 -o output.json
```

Run script for existing managed object with id *10*

```
/opt/noc$ ./noc script --debug get_version 10 -o output.json
```

Run script with SNMP RO community *public*:

```
/opt/noc$ ./noc script --debug Cisco.IOS.get_version cred.json -o output.json
```

`Where cred.json contain:`

```
{
   "scheme": "ssh",
   "address": "192.168.0.2",
   "port": 1022,
   "credentials": {
       "user": "foo", "password": "bar", "super_password": "superpassword", "snmp_ro": "public"
   }
}
```

Run script with parameter *address*:

```
/opt/noc$ ./noc script --debug Cisco.IOS.ping cred.json address="127.0.0.1" -o output.json
```

Create private beef

```
/opt/noc$ ./noc script --debug get_version router1 -o output.json --private
```

Each beef accepts unique GUID number, which can be used as unique identifier.

If you got error like "*Failed to load script DLink.DxS.get_spanning_tree: syntax error*", try this to find error:

```
/opt/noc$ ./noc shell
Python 2.7.11 (default, Apr 26 2016, 03:06:47)
Type "copyright", "credits" or "license" for more information.
IPython 4.2.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: import sa.profiles.DLink.DxS.get_spanning_tree
```

Testing Canned Beef

Before testing canned beed, pull changes to all enabled repos

```
/opt/noc$ ./noc beef --pull
```

Perform full unittests (including beef)

```
/opt/noc$ ./noc test
```

Test only beefs

```
/opt/noc$ ./noc test noc.stop
```

Test single repo (i.e. *sa_public*)

```
/opt/noc$ ./noc test --beef=local/repos/sa/sa_public/ noc.stop
```

To analyze beef code coverage run tests with *--coverage-html-out* key

```
/opt/noc$ ./noc test noc.stop --coverage-html-out=/tmp/coverage
```

then refer to */tmp/coverage/index.html* for details

Beef inventory

Check which beef do you have

```
/opt/noc$ ./noc beef --list
Repo,Profile,Script,Vendor,Platform,Version,GUID
sa_public,3Com.SuperStack,get_mac_address_table,3Com,SuperStack 3 Switch 3300XM,2.72,3aa6ef90-3c69-4bed-a2c7-fdeedf056f5a
sa_public,3Com.SuperStack,get_mac_address_table,3Com,SuperStack 3 Switch 3300XM,2.72,5f32307b-abb3-4ba4-bb5f-a257512231c0
sa_public,3Com.SuperStack,get_mac_address_table,3Com,SuperStack II Switch 1100,2.72,a0eb3fcf-29cc-44ad-af0e-848112c2fa62
sa_public,3Com.SuperStack,get_version,3Com,3Com,2.72,861c1a72-036e-4d3a-a94c-d52e13f3d369
sa_public,3Com.SuperStack,get_version,3Com,SuperStack II Switch 1100,2.72,d9a6dde9-327c-4efc-a3cd-ba7e080539d8
sa_public,3Com.SuperStack,get_version,3Com,SuperStack 3 Switch 3300XM,2.72,eb958499-e678-4192-9a46-aaafdaa6aed7
sa_public,3Com.SuperStack,get_vlans,3Com,SuperStack II Switch 610,2.72,2f141b13-1f18-4fee-98f6-2823cac63f3d
sa_public,3Com.SuperStack,get_vlans,3Com,SuperStack 3 Switch 3300XM,2.72,5c5e7b4b-18ea-49a2-bea1-e7df1338198b
sa_public,Alcatel.AOS,get_arp,Alcatel,OS6850-U24X,6.4.4.343.R01,03e63aeb-cddc-4cb8-bec3-8e96cf6e6b16
...
```

Contribute the Beef

Finally submit an issue with you beef, we will gracefully include it into the next releases. Send it to *beef@nocproject.org*.

DO NOT FORGET TO SET "PRIVATE" FLAG IF YOUR BEEF MAY CONTAIN SENSITIVE INFORMATION