

## Bash+ Introduction

Bash+ is a set of functions written in Linux shell itself that integrates APIs of various providers into the command line of the shell. You can also say that bash+ is a method to command-line the API.

The goal of bash+ is to provide system admins/devops with an easy way to script operations vs different providers. Ideally, even a person with the most basic command of the Linux shell should be able to script operations.

Bash+ achieves its goal by:

1. Taking care of such tasks as managing authentication/authorization tokens and limit rates
2. Downloading and caching, when necessary, sites/accounts information
3. Providing built-in commands for most basic/common tasks
4. Providing tools that allow users to easily construct custom API requests by just copy-pasting relevant instructions from API manuals and without having to know the intricacies of accessing the given API

Bash+ can be seen as a revival of the original Unix/Linux philosophy of “Keep it simple - Keep it in a pipe”. That is, complex tasks should be carried out by a bunch of small simple commands organized in a pipe because there is beauty in simplicity.

Contrary to the common misconception, integrating APIs into the Linux shell is very easy. And bash+ already provides enough examples, conventions and common functions to make such an effort a breath. If every one of you writes a bash+ extension for an API you work with, the end result of our collective effort will be a super-shell that can do everything.

## Summary:

1. [Getting started with the bash+ Incapsula extension](#)
2. [Introduction into bash+ for Incapsula by example](#)
3. [Listing Incapsula sites,accounts and configuration](#)
4. [Printing original configuration json](#)
5. [Notes on incap updatedb](#)
6. [Making API requests with incap api](#)

## Getting started with the bash+ Incapsula extension

You start by sourcing files with bash+ functions. These files also include functions for Incapsula

```
[oskars@lab1 bash+]$  
[oskars@lab1 bash+]$ ls *functions -l  
common.functions  
dnsme_api.functions  
dnsme.functions  
dnsme_sh.functions  
dyndns.functions  
dyndns_sh.functions  
incap_api.functions  
incap.functions  
incap_help.functions  
incap_sh.functions  
[oskars@lab1 bash+]$  
[oskars@lab1 bash+]$ while read i ; do source $i ; done < <(ls *functions -l)  
[oskars@lab1 bash+]$
```

Once you have the functions in your shell and you run the command *incap* for the first time, you will be prompted for credentials.

These are saved as variables inside your shell until you leave the current session

```
[oskars@ bash+]$  
[oskars@ bash+]$ incap  
API ID:  
API Key:  
[oskars@ bash+]$
```

Finally, you download your Incapsula configuration by running *incap updatedb*

```
[oskars@ bash+]$  
[oskars@ bash+]$ incap updatedb  
Accounts: 6 [REDACTED] 69  
63 [REDACTED]: 0  
Account: 6 [REDACTED] 3 Page: 0  
[4] 22782 [REDACTED]  
Account: 3 [REDACTED] 7 Page: 0  
[5] 22794 [REDACTED]  
[5]+ Done { incap_api "account_id=${account}&page_size  
Account: 6 [REDACTED] 7 Page: 0  
[5] 23039 [REDACTED]  
[3] Done { incap_api "account_id=${account}&page_size  
Account: 3 [REDACTED] 9 Page: 0  
[6] 23060 [REDACTED]  
[4] Done { incap_api "account_id=${account}&page_size  
Account: 3 [REDACTED] 4 Page: 0  
[7] 23182 [REDACTED]  
█
```

*incap updateb* will be explained in more detail later in one of the next sections

Once the Incapsula configuration downloaded, you can list all your sites by running *incap ls sites*

```
[oskars@ bash+]$  
[oskars@ bash+]$ incap ls sites  
Site Id      Site Name      Account      IPs      CNAME  
17 177       www[redacted].com 6[redacted]7 27[redacted] ve[redacted]ncapdns.net  
14 12        www[redacted].com 6[redacted]7 10[redacted] 7[redacted]ncapdns.net  
10 11         www[redacted]on.com 6[redacted]7 27[redacted] h[redacted]ncapdns.net  
14 13         www[redacted].com 6[redacted]7 21[redacted] t[redacted]ncapdns.net  
10 18         www[redacted].com 6[redacted]7 21[redacted] k[redacted]ncapdns.net  
12 16         www[redacted].s.com 6[redacted]7 21[redacted] 9[redacted]ncapdns.net  
14 18         www[redacted].s.com 6[redacted]7 13[redacted] y[redacted]ncapdns.net  
11 18         www[redacted].com 6[redacted]7 hk[redacted]com f[redacted]ncapdns.net  
15 14         www[redacted]tion.com 6[redacted]7 21[redacted] 3[redacted]ncapdns.net  
11 11         spo[redacted]na[redacted]m 3[redacted]9 21[redacted] y[redacted]ncapdns.net  
11 10         spo[redacted]lil[redacted]y.com 3[redacted]9 21[redacted] z[redacted]ncapdns.net  
11 17         www[redacted].com 6[redacted]7 10[redacted] 6[redacted]ncapdns.net  
19 13         new[redacted].com 6[redacted]7 21[redacted] r[redacted]ncapdns.net
```

You can filter the list using `grep`.

```
oskars@ bash+]$  
oskars@ bash+]$ incap ls sites | grep oskar  
oskars@ bash+]$  
oskars@ bash+]$
```

There is no site whose name or origin includes oskar at this point.

Next, lets create a new Incapsula site using command *incap site mk*

```
[oskars@ bash+]$  
[oskars@ bash+]$ incap site mk help  
  
<function> <account id> <site name> <origin ip>  
  
[oskars@ bash+]$  
[oskars@ bash+]$ incap site mk 309 demo.oskar-test.com 192.1.1.1  
Creating site demo.oskar-test.com  
"OK"  
Setting ignore_ssl to read the incapdns record:  
"OK"  
[oskars@ bash+]$  
[oskars@ bash+]$ incap ls sites | grep oskar  
11652521      demo.oskar-test.com  309      192.1.1.1      yk35q6z.x.incapdns.net  
[oskars@ bash+]$
```

This time when we filter *incap ls sites* with *grep*, the new site shows up

# Introduction into bash+ for Incapsula by example

This is a real world case of the Incapsula extension of bash+ in action.  
The request came to disable TCP-pre-pooling on all sites under a certain Incapsula account (Incapsula account is a group of sites).  
So you go to Incapsula online [API Doc Manual](#) and find relevant instructions

## Advanced Caching Settings

Use this operation to modify advanced caching settings.

```
/api/prov/v1/sites/performance/advanced
```

Parameters:

Name	Description	Optional
api_id	API authentication identifier	
api_key	API authentication identifier	
site_id	Numeric identifier of the site to operate on.	
param	Name of configuration parameter to set, see table below	
value	According to the param value, see table below	

Name	Description
async_validation	Sets Async validation. Pass "true" or "false" in the value parameter

tcp_pre_pooling	TCP <b>Pre-Pooling</b> , Pass "true" or "false" in the value parameter
-----------------	--

Basically, to disable pre pooling you need to send a POST request to <https://my.incapsula.com/api/prov/v1/sites/performance/advanced>

Your request should look like this:

*api\_id=<Your API id>&api\_key=<Your API key>&site\_id=<Your site id>&param=tcp\_pre\_pooling&value=false*

To run such a request you use command *incap api*. The command simplifies your task. You only need to copy relevant arguments from the API Manual. Your command would look like this:

*incap api "site\_id=<Your site id>&param=tcp\_pre\_pooling&value=false"*  
*/api/prov/v1/sites/performance/advanced*

The only parameter missing with this command is the site id. In the next page you will find the ids of all sites of that account



You can list your sites and their configuration using command *incap ls*.  
For this example, you run *incap ls accounts* to receive the list of your accounts

```
[oskars@ bash+]$  
[oskars@ bash+]$  
[oskars@ bash+]$ incap ls accounts  
Account ID      Account Name  
6 1             Customers  
3 1             TradeRush  
6 1             WordPress  
3 1             API  
3 1             A  
3 1             potStreamer  
3 1             WW  
3 1             Special  
2 1             CRM  
[oskars@ bash+]$  
[oskars@ bash+]$
```

Next, you run *incap ls accounts <account id>* and receive the list of all sites under that account

```
[oskars@ bash+]$  
[oskars@ bash+]$ incap ls accounts 6803  
Site Id      Site Name      Account  IPs      CNAME  
1129        www.           6803     13       8  
1172        www.           6803     18       d  
1157        api-           6803     77       2  
1161        api.           6803     B        west-1.elb.amazona  
1125        lp.           6803     1p       ce-eu-west-1.amazon  
1183        www.           6803     98       5  
1146        goog           6803     10       6  
1157        www.           6803     b        5.eu-west-1.elb.ama  
1123        www.           6803     21       e  
1140        www.           6803     10       j  
1156        bo.           6803     L        west-1.elb.amazonaw  
1187        engi           6803     S        50.eu-west-1.elb.am  
1105        old.           6803     21       9  
1150        gen.           6803     11       1.eu-west-1.elb.ama  
1175        files.         6803     18       c  
1127        pay.           6803     98       s  
1122        sta.           6803     52       d
```

To save all site ids into the variable , you run:

```
ids=$( incap ls accounts <account id> | egrep -o ^[0-9]+)
```

And finally you run the entire command as:

```
for id in $ids ; do  
    incap api "site_id=${id}&param=tcp_pre_pooling&value=false" /api/prov/v1/sites/performance/advanced  
done
```

## Listing Incapsula sites,accounts and configuration

At the top level bash+ for Incapsula displays 4 pseudo directories: accounts, sites, origins and cache

```
[oskars@ bash+]$  
[oskars@ bash+]$ incap ls  
accounts  
sites  
origins  
cache  
[oskars@ bash+]$  
[oskars@ bash+]$
```

You have already seen how to work with sites and accounts by now.

One additional command is *incap ls sites <site id>* prints a summary of the site configuration

```
[oskars@ bash+]$  
[oskars@ bash+]$ site_id=1[REDACTED]  
[oskars@ bash+]$  
[oskars@ bash+]$ incap ls sites $site_id  
Site ID           = 18[REDACTED]  
Site Name         = www[REDACTED]  
Site Status       = pending-dns-changes  
Account ID        = 31[REDACTED]  
Origin            = tr[REDACTED]-1.elb.am  
Cache Mode        = advanced  
Block URLs        = wp-login,wp-admin  
SSL CA            = GS  
SSL San           = ww[REDACTED]om  
SSL Detection     = true  
SSL Detection Status = ok  
SSL Validation Status = done  
DNS CNAME Record  = y[REDACTED]  
DNS A Records     = 1[REDACTED]8.96  
[oskars@ bash+]$
```

Origins work on the same principle as accounts. There is a list of all origins your sites point to.

```
[oskars@ bash+]$ incap ls origins
```

```
Origin
```

```
21
21
27
21
27
27
21
21
10
27
BE 714351734.eu-west-1.elb.amazonaws.com
27
P1 -746990009.eu-west-1.elb.amazonaws.com
21
21
WE amazonaws.com
P1 8.eu-west-1.elb.amazonaws.com
LE elb.amazonaws.com
Ap elb.amazonaws.com
21
146
```

You can also print all sites that point to a certain IP or hostname.  
This is useful when you need to get the ids of all sites that should be migrated from one data center to another

*incap ls origins <origin>*

```
[oskars@ bash+]$  
[oskars@ bash+]$  
[oskars@ bash+]$ incap ls origins 200.1.1.7  
Site Id      Site Name      Account Id      IPs  
7122        spotcfdf.com   9              213.141.7  
178         spotcfdf.com   9              213.141.7  
746         spotcfdf.com   9              213.141.7  
585         spotcfdf.com   9              213.141.7  
576         spotcfdf.com   9              213.141.7  
941         spotcfdf.com   8              213.141.7  
241         spotcfdf.com   9              213.141.7  
709         spotcfdf.com   9              213.141.7  
478         spotcfdf.com   9              213.141.7  
690         spotcfdf.com   9              213.141.7  
760         spotcfdf.com   9              213.141.7  
100         spotcfdf.com   9              213.141.7  
907         spotcfdf.com   9              213.141.7  
1172        spotcfdf.com   9              213.141.7  
[oskars@ bash+]$  
[oskars@ bash+]$
```

*incap ls origins* is not only easier to work with. *incap ls sites/accounts* truncates long hostnames to keep the output in a table. *incap ls origins <origin ip/hostname>* will output exactly the sites you need with no place for errors

## Printing original configuration json with *incap cat*

You can print the original json of Incapsula configuration using *incap cat* command  
*incap cat accounts* will print the json of all your accounts

```
[oskars@ bash+]$ incap cat accounts
{"accounts": [{"id": "316", "name": "oskars", "email": "oskars@bash.com", "password": "123456", "login": true, "f_id": "332", "ed": "t", "name": "oskars", "ins": "t", "id": "t", "id": "t", "ue": "t", "lan": "t", "Opti": "t", "reys": "t", "spoto": "t"}]}
```

*incap cat sites* will print jsons of all files. Not showing here

*incap cat sites <site id>* or *<site name>* will print the json of that site

```
[oskars@ bash+]$  
[oskars@ bash+]$  
[oskars@ bash+]$ site_id=1  
[oskars@ bash+]$  
[oskars@ bash+]$ incap cat sites $site_id  
{  
  "site": "1",  
  "date": "2017-07-25T10:25:00Z",  
  "et": "et",  
  "om": "om",  
  "": [],  
  "user": "user",  
  "ck Us": "ck Us",  
  "id": "id",  
  "bot_a": "bot_a",  
  "c_thr": "c_thr",  
  "s.bac": "s.bac",  
  "": "Remote File Inclusion",  
  "action": "api.threats"
```

*incap cat sites <site id/domain>* print configuration stored locally. If you want to be 100% sure that you are not missing any changes, for example done by other people thru Incapsula web site, use *incap site status <site id>*. This command will download the latest configuration of the site from Incapsula API



## Notes on *incap updatedb*

Incapsula identifies all sites by their ids and not by their names. At the moment, Incapsula API doesn't have a command that can query the site id by the site's name. Instead, you should download the entire configuration of all your sites to be able to map names to ids.

Even if such a command existed, there are still advantages to have your Incapsula configuration locally cached because this allows you to search your sites by their account or origin IP, like in the examples above, or any other parameter.

It's a good practice to sync your Incapsula configuration every once in a while. However, *bash+* generally keeps your local cache updated with the following tricks.

First of all, when you create or modify a site, Incapsula API returns the updated configuration in a json format. *incap api* automatically detects if the response contains the site configuration and updates your cache.

Two, when you delete a site using command *incap site rm <site id>*, the site is deleted from your local cache as well. Below is a few examples:

```
[oskars@ bash+]$
[oskars@ bash+]$ incap ls sites | grep oskar
11652521      demo.oskar-test.com      309  192.1.1.1  yk35q6z.x.incapdns.net
[oskars@ bash+]$
[oskars@ bash+]$ incap site ip 11652521 192.1.1.2
{
  "site_id": 11652521,
  "domain": "demo.oskar-test.com",
  "new_ips": [
    "192.1.1.2"
  ],
  "old_ips": [
    "192.1.1.1"
  ]
}
[oskars@ bash+]$
[oskars@ bash+]$ incap ls sites | grep oskar
11652521      demo.oskar-test.com      309  192.1.1.2  yk35q6z.x.incapdns.net
[oskars@ bash+]$
[oskars@ bash+]$ incap site rm 11652521
"OK"
removed `/dev/shm/oskars/incapsula/sites/demo.oskar-test.com_11652521'
[oskars@ bash+]$
[oskars@ bash+]$ incap ls sites | grep oskar
[oskars@ bash+]$
```

## Making API requests with *incap api*

You have already seen that *incap api* is evoked as *incap api <post data> <url/uri>*

It's important to keep in mind that *incap api* doesn't print output on success. It only prints error messages when requests fail.

If you do need access to the output of the request, it's stored in a special reserved variable called *res*.

In the example below we access this variable to read the TXT record for ssl validation.

```
[oskars@ bash+]$
[oskars@ bash+]$ incap ls sites demo.oskar-test.com
Site ID                = 13663571
Site Name              = demo.oskar-test.com
Site Status            = pending-dns-changes
Account ID             = 3[REDACTED]
Origin                 = 192.1.1.1
Cache Mode             = advanced
Block URLs             =
SSL CA                 =
SSL San                =
SSL Detection          = true
SSL Detection Status   = ok
SSL Validation Status  =
DNS CNAME Record      = gcnz4at.x.incapdns.net
DNS A Records         =
[oskars@ bash+]$
[oskars@ bash+]$ incap site ssl 13663571
[oskars@ bash+]$
[oskars@ bash+]$ echo $res_code
0
[oskars@ bash+]$
[oskars@ bash+]$ echo $res | jq .
{
  "res": 0,
  "res_message": "OK",
  "debug_info": {
    "id-info": "9088",
    "domain_dns": "globalsign-domain-verification=cxIzH
  }
}
[oskars@ bash+]$
```

Another variable *res\_code* saves the exit code of the last *incap api* call.

*res\_code* can have the following values:

0 - Success

1 - API returned error

2 - Request failed and the error message received is not json. Most likely you misspelled the url and the response is an html page

10+ - Error codes of curl program which bash+ uses for requests. They are calculated by adding 10 to the original exit code of curl

On errors *incap api* prints *res\_code*, the history list of the last 10 functions including the last request and the arguments of the last request

```
[oskars@ bash+]$  
[oskars@ bash+]$  
[oskars@ bash+]$ incap api blablablah blahblahblah  
res_code: 2  
src: incap_api incap incapsula_wp_ssl_reminder incap_run source  
args: blablablah blahblahblah  
msgs: Request failed. Response is not json  
      The resource you requested could not be found  
      You are welcome to contact our support team  
      Thanks and our apology  
[oskars@ bash+]$  
[oskars@ bash+]$  
[oskars@ bash+]$ ■
```

## Additional commands

```
[oskars@ bash+]$  
[oskars@ bash+]$ incap site id demo.oskar-test.com  
13663571  
[oskars@ bash+]$  
[oskars@ bash+]$ incap site id 13663571  
13663571  
[oskars@ bash+]$  
[oskars@ bash+]$ incap site purge cache 13663571  
Purging site 13663571 cache...  
"OK"  
[oskars@ bash+]$  
[oskars@ bash+]$ incap site purge hostname demo.oskar-test.com  
Purging demo.oskar-test.com from hostname cache...  
"OK"  
[oskars@ bash+]$
```

bash+ also provides some auxiliary functions, such as *json2bash* for reformatting json in case you may find it more comfortable to work with json

```
[oskars@ bash+]$
[oskars@ bash+]$ incap site ssl 13663571
[oskars@ bash+]$
[oskars@ bash+]$ echo $res | jq .
{
  "res": 0,
  "res_message": "OK",
  "debug_info": {
    "id-info": "9088",
    "domain_dns": "globalsign-domain-verification=cxIzHZv5AyxGt-12DZAw0F2CkSmRka04Z-4M6_128b"
  }
}
[oskars@ bash+]$
[oskars@ bash+]$ echo $res | jq . | json2bash
res                = 0
res_message        = OK
debug_info id-info = 9088
debug_info domain_dns = globalsign-domain-verification=cxIzHZv5AyxGt-12DZAw0F2CkSmRka04Z-4M6_128b
[oskars@ bash+]$
[oskars@ bash+]$
```



