An easier approach to K5 IaaS API scripting

Current situation

The existing script templates provided for scripting K5 operations are not straightforward to use. For example, in order to list all the subnets in a K5 project, you have to perform the following:

1. Get the 32-digit hexadecimal ID for the K5 contract;

2. Get the 32-digit hexadecimal ID for the project (a.k.a tenant) within that contract;

3. Edit the `init.sh` file to save the IDs as environment variables, making sure you also have the correct K5 user name and its current password defined in there;

4. Run `get_token.sh` to retrieve an authentication token;

5. Edit the `list_Subnets.sh` script to insert the known network ID;

6. Run `list_Subnets.sh` to list the JSON output from a query of the given network.

You can repeat step 6 as desired, but after two hours you have to repeat steps 4 and 5 because the authentication token will have expired. Also, if you want to query a different contract or project, you have to start from scratch.

Revised scripts

I have produced a new set of bash scripts to make things easier. Whilst these scripts were based on the original templates, I have made the following improvements:

1. No editing required

   The scripts are designed not to need editing. They all make use of the `k5token.sh` script which prompts you at run time for the target environment, i.e. contract name, project name, user name and password – the last two only if you don't already have a current authentication token for the target environment.

2. Caching of data

   Variable data such as the last contract name and user name are cached in local files and presented as default but overridable responses, so you don't need to retype them each time. Optionally, your password can also be cached if you wish, in which case you will only be prompted for it if it is incorrect, for example if it has expired or changed. Authentication tokens are also cached per contract or project, so you can target mixed environments without having to re-authenticate each time.

3. Automatic detection of stale authentication token

   As the `k5token.sh` script saves the authentication token in a file, it can automatically detect when the token has expired based on the file timestamp. This means you don't have to remember how long ago you last authenticated.

4. Symbolic name input

Instead of having to enter a 32-digit hexadecimal ID string, you can enter the short contract name (e.g. Jvx3Tz5B) or project name.

5. Scrollable lists

To avoid having to type in a project name or ID, all project names within a contract are presented as a scrollable, cursor-controlled list. This is populated by querying the contract dynamically. This facility is also used for other data, e.g. for selecting a server flavour when creating a server.

6. Runtime parameters

The scripts also allow data to be supplied as run-time arguments, e.g.:

```
./listServers.sh --contract Jvx3Tz5B --project Alpha_Project
```

7. Minimal external dependencies

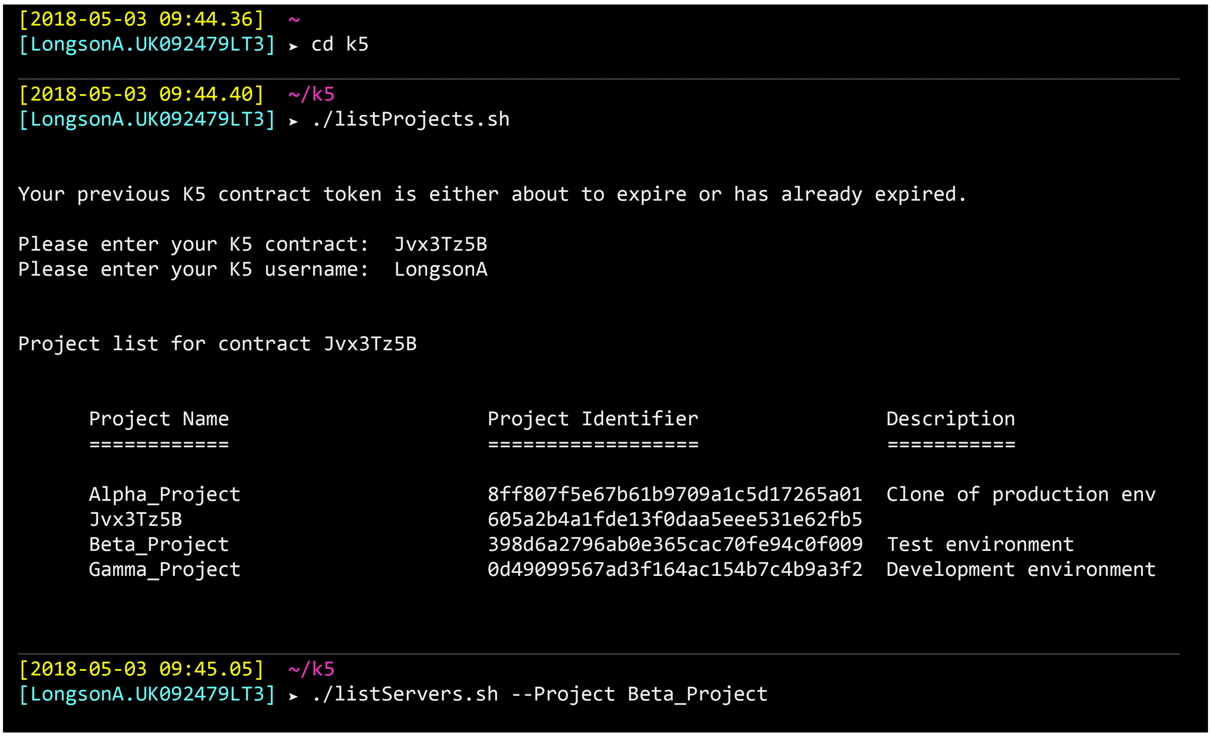
As these are all bash scripts and only require the curl and jq commands, Python need not be installed. They are designed to work under Cygwin, MobaXTerm, or pure Linux environments such as Ubuntu or Red Hat. Note that if you have issues with your particular environment, please let me know and I will try to resolve this if I can.

8. Runtime help

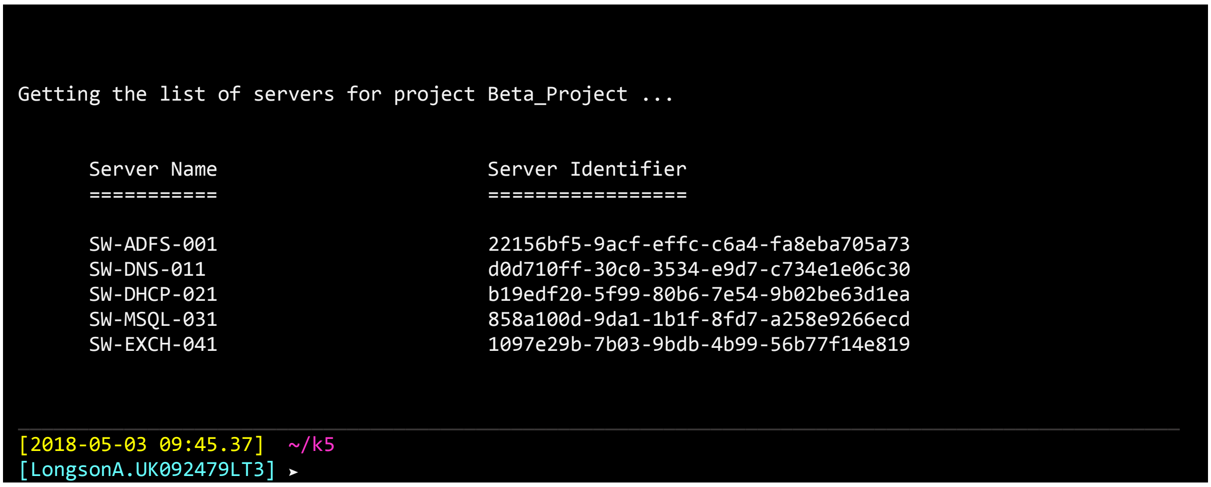
The scripts support a help parameter to show how to use them, e.g.:

```
./listServers.sh -h
```

Example session – MobaXTerm



```
[2018-05-03 09:44.36]  ~
[LongsonA.UK092479LT3] ➤ cd k5

[2018-05-03 09:44.40]  ~/k5
[LongsonA.UK092479LT3] ➤ ./listProjects.sh


Your previous K5 contract token is either about to expire or has already expired.

Please enter your K5 contract:  Jvx3Tz5B
Please enter your K5 username:  LongsonA


Project list for contract Jvx3Tz5B


      Project Name                      Project Identifier                Description
      ============                      ==================                ===========

      Alpha_Project                     8ff807f5e67b61b9709a1c5d17265a01  Clone of production env
      Jvx3Tz5B                          605a2b4a1fde13f0daa5eee531e62fb5
      Beta_Project                      398d6a2796ab0e365cac70fe94c0f009  Test environment
      Gamma_Project                     0d49099567ad3f164ac154b7c4b9a3f2  Development environment



[2018-05-03 09:45.05]  ~/k5
[LongsonA.UK092479LT3] ➤ ./listServers.sh --Project Beta_Project
```



```
Getting the list of servers for project Beta_Project ...


      Server Name                       Server Identifier
      ===========                       ================

      SW-ADFS-001                       22156bf5-9acf-effc-c6a4-fa8eba705a73
      SW-DNS-011                        d0d710ff-30c0-3534-e9d7-c734e1e06c30
      SW-DHCP-021                       b19edf20-5f99-80b6-7e54-9b02be63d1ea
      SW-MSQL-031                       858a100d-9da1-1b1f-8fd7-a258e9266ecd
      SW-EXCH-041                       1097e29b-7b03-9bdb-4b99-56b77f14e819


[2018-05-03 09:45.37]  ~/k5
[LongsonA.UK092479LT3] ➤
```

In the above example, I didn't need to type the contract name or user name, they were presented as default options so I only had to press the enter key.

Cache files

There are a number of files which the scripts use to cache data in order to simplify user interaction. These all begin with a dot so they are not listed by default when running `ls` (although using the `-A` argument will list them anyway):

1. .k5ids

   This contains a list of user names, contract names and project names, along with their 32-digit hexadecimal identifiers.

2. .k5lastcontract, .k5lastproject and .k5lastuser

   These contain the names of the last contract, project and user which were used and are used for the default values in user prompts.

3. .k5pwoption.<contract>.<username>

   Each of these files contains two lines. The first contains your password save preference (Never save / Prompt / Always save) and the second contains your base64-encoded K5 password, but only if you opted to save it.

Implementation

It's best to copy the files to a separate folder in your user profile such as the k5 folder you saw in the example session. Besides the script files themselves there is also a `servertypes.txt` file which is used by the `createServer.sh` script when prompting the user for the required server flavour (if not supplied on the command line).

Final note

These scripts don't cover every area of K5 and are far from complete, but they should provide a framework on which you can create a more automated approach to K5 configuration and administration.

Tony Longson (tony.longson@uk.fujitsu.com).

22nd May 2018.