Using Satellite API

Satellite 6.2.6

Pablo Hess
phess@redhat.com

Waldirio M. Pinheiro
waldirio@redhat.com
2017-02-07

Table of Contents

Consuming API **3**

Checking Available API Methods **3**

User Method **3**

Getting User information via webUI 3

Getting User information via CLI 3

Getting User information via CLI - formatted output 3

Creating new User passing all parameters 4

Via CLI with all parameters 4

External file can be used in different formats, according below 4

Creating new User passing all parameters via *external file* 5

Life Cycle **5**

Getting Life Cycle information via webUI 5

Getting Life Cycle information via CLI 5

Getting Life Cycle information via CLI - formatted output 5

Creating one new Life Cycle Environment lfc_api_dev 6

You can create the external json file with structure below 6

Calling the Curl passing the external json file 6

Creating another Life Cycle *lfc_api_qa* with prior the first Life Cycle *lfc_api_dev* 7

Life Cycle via webUI 7

Puppet Classes **8**

Getting a smart class parameter id 8

Example: the motd module and the content parameter id 8

Getting info about a specific smart class parameter 9

Modifying a smart class parameter using an external file 10

Consuming API

Here will be possible to check how to query the actual values on Satellite via API, this is very useful because you will be able to check the json structure and the use it in the future.

Checking Available API Methods

The link to API documentation on your own Satellite Server
https://<satellite-server>/apidoc

On the link above, you will be able to navigate and check all Methods and parameters necessary to consume the API on your own Satellite Server, so in some examples here you have to pass the mandatory parameters although but feel free if you need to pass additional parameters.

Official documentation: https://access.redhat.com/documentation/en/red-hat-satellite/6.2/single/api-guide

User Method

Here we will use the User method to show how to query and create a new user on Satellite Environment via API.

Getting User information via webUI

https://<satellite_server>/api/users

Getting User information via CLI

```
$ curl -X GET -s -k -u admin:redhat https://<satellite_server>/api/users
{
  "total": 1,
  "subtotal": 1,
  "page": 1,
  "per_page": 20,
  "search": null,
  "sort": {
        "by": null,
        "order": null
  },
  "results":
[{"firstname":"Admin","lastname":"User","mail":"root@example.rh","admin":true,"auth_source_id":1,"auth_source_name"
:"Internal","timezone":null,"locale":null,"last_login_on":"2017-01-24 19:26:50 UTC","created_at":"2016-12-16
15:41:13 UTC","updated_at":"2017-01-24 19:26:50
UTC","id":3,"login":"admin","default_location":null,"locations":[],"default_organization":{"id":1,"name":"ACME","ti
tle":"ACME","description":""},"organizations":[]}]
}
$
```

Getting User information via CLI - formatted output

```
$ curl -X GET -s -k -u admin:redhat https://<satellite_server>/api/users | python -mjson.tool
{
        "page": 1,
        "per_page": 20,
        "results": [
        {
```

```
                "admin": true,
                "auth_source_id": 1,
                "auth_source_name": "Internal",
                "created_at": "2016-12-16 15:41:13 UTC",
                "default_location": null,
                "default_organization": {
                "description": "",
                "id": 1,
                "name": "ACME",
                "title": "ACME"
                },
                "firstname": "Admin",
                "id": 3,
                "last_login_on": "2017-01-24 19:30:07 UTC",
                "lastname": "User",
                "locale": null,
                "locations": [],
                "login": "admin",
                "mail": "root@example.rh",
                "organizations": [],
                "timezone": null,
                "updated_at": "2017-01-24 19:30:07 UTC"
        }
        ],
        "search": null,
        "sort": {
        "by": null,
        "order": null
        },
        "subtotal": 1,
        "total": 1
}
$
```


Creating new User passing all parameters

Via CLI with all parameters

```
curl -H "Accept:application/json,version=2" \
        -H "Content-Type:application/json" -X POST \
        -u admin:redhat -k \
        -d
"{\"firstname\":\"Test\",\"lastname\":\"API\",\"mail\":\"test@api.com\",\"login\":\"test_api\",\"password\":\"12345
6\",\"auth_source_id\":1}" \
        https://<satellite_server>/api/users
```

External file can be used in different formats, according below

```
$ cat file.json
{"firstname":"Test","lastname":"API","mail":"test@api.com","login":"test_api","password":"123456","auth_source_id":
1}
$
```

Or

```
$ cat file.json
{
        "auth_source_id": 1,
        "login": "test_api",
        "mail": "test@api.rh",
```

```
        "password": "123456"
}
$
```

Creating new User passing all parameters via *external file*

```
curl -H "Accept:application/json,version=2" \
        -H "Content-Type:application/json" -X POST \
        -u admin:redhat -k \
        -d  "$(< file.json)" \
        https://<satellite_server>/api/users
```

Life Cycle

Here we will use the Life Cycle method to show how to query and create a new life cycle on Satellite Environment via API.

Getting Life Cycle information via webUI

https://<satellite_server>/katello/api/environments/1

Getting Life Cycle information via CLI

```
$ curl -X GET -s -k -u admin:redhat https://<satellite_server>/katello/api/environments/1

{"library":true,"id":1,"name":"Library","label":"Library","description":null,"organization":{"name":"ACME","label":
"ACME","id":1},"created_at":"2016-12-16 15:41:17 UTC","updated_at":"2016-12-16 15:41:17
UTC","prior":null,"successor":null,"counts":{"content_hosts":5,"content_views":3,"packages":12017,"puppet_modules":
3,"errata":{"security":308,"bugfix":868,"enhancement":173,"total":1349},"yum_repositories":5,"docker_repositories":
0,"ostree_repositories":0,"products":3},"permissions":{"view_lifecycle_environments":true,"edit_lifecycle_environme
nts":true,"destroy_lifecycle_environments":false,"promote_or_remove_content_views_to_environments":true}}
$
```

Getting Life Cycle information via CLI - formatted output

```
$ curl -X GET -s -k -u admin:redhat https://<satellite_server>/katello/api/environments/1 | python -mjson.tool
{
        "counts": {
        "content_hosts": 5,
        "content_views": 3,
        "docker_repositories": 0,
        "errata": {
                "bugfix": 868,
                "enhancement": 173,
                "security": 308,
                "total": 1349
        },
        "ostree_repositories": 0,
        "packages": 12017,
        "products": 3,
        "puppet_modules": 3,
        "yum_repositories": 5
        },
        "created_at": "2016-12-16 15:41:17 UTC",
        "description": null,
        "id": 1,
```

```
        "label": "Library",
        "library": true,
        "name": "Library",
        "organization": {
        "id": 1,
        "label": "ACME",
        "name": "ACME"
        },
        "permissions": {
        "destroy_lifecycle_environments": false,
        "edit_lifecycle_environments": true,
        "promote_or_remove_content_views_to_environments": true,
        "view_lifecycle_environments": true
        },
        "prior": null,
        "successor": null,
        "updated_at": "2016-12-16 15:41:17 UTC"
}
$
```

Creating one new Life Cycle Environment lfc_api_dev

```
curl -H "Accept:application/json,version=2" \
        -H "Content-Type:application/json" -X POST \
        -u admin:redhat -k \
        -d  "{\"organization_id\":1,\"name\":\"lfc_api_dev\",\"prior\":1}" \
        https://<satellite_server>/katello/api/environments

$ curl -H "Accept:application/json,version=2" \
>       -H "Content-Type:application/json" -X POST \
>       -u admin:redhat -k \
>       -d  "{\"organization_id\":1,\"name\":\"lfc_api_dev\",\"prior\":1}" \
>       https://<satellite_server>/katello/api/environments

{"library":false,"id":7,"name":"lfc_api_dev","label":"lfc_api_dev","description":null,"organization":{"name":"ACME"
,"label":"ACME","id":1},"created_at":"2017-01-24 23:41:32 UTC","updated_at":"2017-01-24 23:41:32
UTC","prior":{"name":"Library","id":1},"successor":null,"counts":{"content_hosts":0,"content_views":0},"permissions
":{"view_lifecycle_environments":true,"edit_lifecycle_environments":true,"destroy_lifecycle_environments":true,"pro
mote_or_remove_content_views_to_environments":true}}
$
```

You can create the external json file with structure below

```
$ cat file_lfc.json
{"organization_id":1,"name":"lfc_api_dev","prior":1}
$
```

Or

```
$ cat file_lfc.json
{
        "organization_id":1,
        "name":"lfc_api_dev",
        "prior":1
}
$
```

Calling the Curl passing the external json file

```
curl -H "Accept:application/json,version=2" \
        -H "Content-Type:application/json" -X POST \
```

```
        -u admin:redhat -k \
        -d  "$(cat file_lfc.json)" \
        https://<satellite_server>/katello/api/environments
```

```
$ curl -H "Accept:application/json,version=2" \
>       -H "Content-Type:application/json" -X POST \
>       -u admin:redhat -k \
>       -d  "$(cat file_lfc.json)" \
>       https://<satellite_server>/katello/api/environments

{"library":false,"id":9,"name":"lfc_api_dev","label":"lfc_api_dev","description":null,"organization":{"name":"ACME"
,"label":"ACME","id":1},"created_at":"2017-01-24 23:49:19 UTC","updated_at":"2017-01-24 23:49:19
UTC","prior":{"name":"Library","id":1},"successor":null,"counts":{"content_hosts":0,"content_views":0},"permissions
":{"view_lifecycle_environments":true,"edit_lifecycle_environments":true,"destroy_lifecycle_environments":true,"pro
mote_or_remove_content_views_to_environments":true}}
$
```

Creating another Life Cycle *lfc_api_qa* with prior the first Life Cycle *lfc_api_dev*

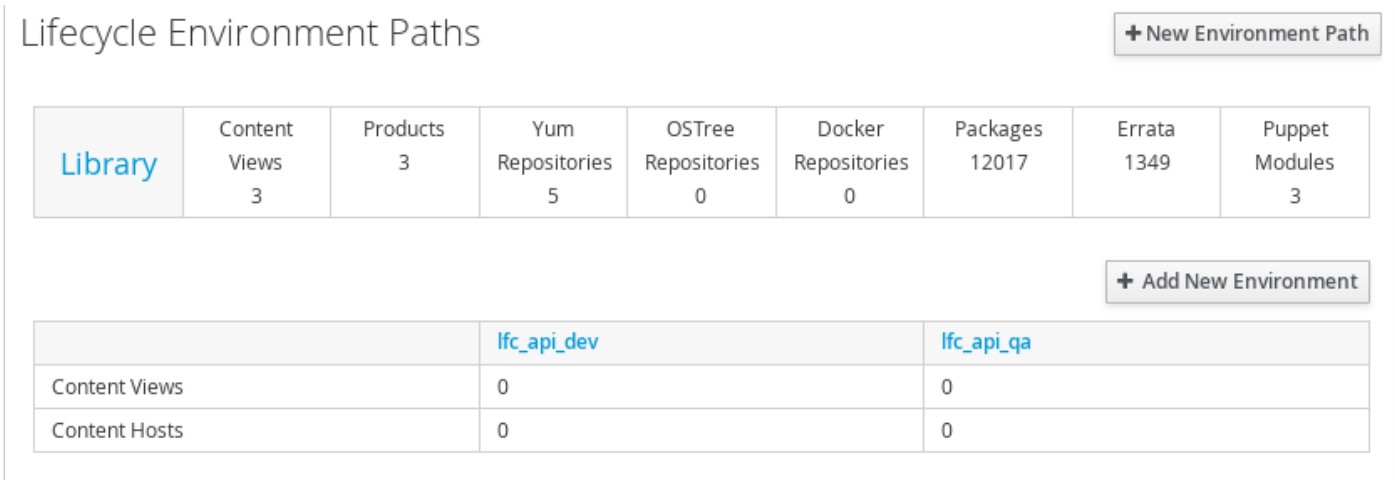
As you can see above, the lfc_api_dev was created with id 9, we will use this one as prior to our QA environment
Below the new file pointing to the new prior and the call will be the same command as above.

```
$ cat file_lfc.json
{
        "organization_id":1,
        "name":"lfc_api_qa",
        "prior":9
}
$
```

```
$ curl -H "Accept:application/json,version=2" \
>       -H "Content-Type:application/json" -X POST \
>       -u admin:redhat -k \
>       -d  "$(cat file_lfc.json)" \
>       https://<satellite_server>/katello/api/environments

{"library":false,"id":10,"name":"lfc_api_qa","label":"lfc_api_qa","description":null,"organization":{"name":"ACME",
"label":"ACME","id":1},"created_at":"2017-01-24 23:53:42 UTC","updated_at":"2017-01-24 23:53:42
UTC","prior":{"name":"lfc_api_dev","id":9},"successor":null,"counts":{"content_hosts":0,"content_views":0},"permiss
ions":{"view_lifecycle_environments":true,"edit_lifecycle_environments":true,"destroy_lifecycle_environments":true,
"promote_or_remove_content_views_to_environments":true}}
$
```

Life Cycle via webUI



Puppet Classes

Getting a smart class parameter id

```
$ curl -H "Accept:application/json,version=2" \
        -H "Content-Type:application/json" -X GET \
        -u $user:$passwd  -k \
        "https://<satellite_server>/api/smart_class_parameters?search=puppetclass_name=<some_name>” \
        | python -m json.tool
```

Example: the motd module and the content parameter id

```
$ curl -H "Accept:application/json,version=2" \
        -H "Content-Type:application/json" -X GET \
        -u $user:$passwd  -k \
        "https://<satellite_server>/api/smart_class_parameters?search=puppetclass_name=motd” \
        | python -m json.tool
{
    "page": 1,
    "per_page": 20,
    "results": [
        {
            "avoid_duplicates": false,
            "created_at": "2017-02-06 12:37:48 UTC",
            "default_value": "",
            "description": "",
            "hidden_value": "*****",
            "hidden_value?": false,
            "id": 3,
            "merge_default": false,
            "merge_overrides": false,
            "override": false,
            "override_value_order": "fqdn\nhostgroup\nos\ndomain",
            "override_values_count": 0,
            "parameter": "content",
            "parameter_type": "string",
            "puppetclass_id": 3,
            "puppetclass_name": "motd",
            "required": false,
            "updated_at": "2017-02-07 13:08:42 UTC",
            "use_puppet_default": false,
            "validator_rule": null,
            "validator_type": ""
        },
        {
            "avoid_duplicates": false,
            "created_at": "2017-02-06 12:37:48 UTC",
            "default_value": true,
            "description": "",
            "hidden_value": "*****",
            "hidden_value?": false,
            "id": 1,
            "merge_default": false,
            "merge_overrides": false,
            "override": false,
            "override_value_order": "fqdn\nhostgroup\nos\ndomain",
            "override_values_count": 0,
```

```
            "parameter": "dynamic_motd",
            "parameter_type": "boolean",
            "puppetclass_id": 3,
            "puppetclass_name": "motd",
            "required": false,
            "updated_at": "2017-02-06 15:21:06 UTC",
            "use_puppet_default": null,
            "validator_rule": null,
            "validator_type": null
        },
        {
            "avoid_duplicates": false,
            "created_at": "2017-02-06 12:37:48 UTC",
            "default_value": "",
            "description": "",
            "hidden_value": "*****",
            "hidden_value?": false,
            "id": 2,
            "merge_default": false,
            "merge_overrides": false,
            "override": false,
            "override_value_order": "fqdn\nhostgroup\nos\ndomain",
            "override_values_count": 0,
            "parameter": "template",
            "parameter_type": "string",
            "puppetclass_id": 3,
            "puppetclass_name": "motd",
            "required": false,
            "updated_at": "2017-02-06 15:21:06 UTC",
            "use_puppet_default": null,
            "validator_rule": null,
            "validator_type": null
        }
    ],
    "search": "puppetclass_name=motd",
    "sort": {
        "by": null,
        "order": null
    },
    "subtotal": 3,
    "total": 66
}
```


Each smart class parameter has an ID that is global for the same satellite instance. The `content` parameter of the `motd` class has `id=3` on this satellite server.

Getting info about a specific smart class parameter

Following on the example of `id=3`.

```
$ curl -H "Accept:application/json,version=2" \
        -H "Content-Type:application/json" -X GET \
        -u $user:$passwd  -k \
        "https://<satellite_server>/api/smart_class_parameters/3” \
        | python -m json.tool
```

Print the output of this command to an external file:

```
$ curl -H "Accept:application/json,version=2" \
        -H "Content-Type:application/json" -X GET \
        -u $user:$passwd  -k \
```

```
        "https://<satellite_server>/api/smart_class_parameters/3" \
        | python -m json.tool > output_file.json
```

This will print all attributes of the `content` parameter to file `output_file.json`.

Modifying a smart class parameter using an external file

Open the file above and modify the desired values.
In this example we wish to change the content parameter of the motd module, which **requires toggling the `override` item**:

```
$ cat output_file.json
{
    "avoid_duplicates": false,
    "created_at": "2017-02-06 12:37:48 UTC",
    "default_value": "",
    "description": "",
    "hidden_value": "*****",
    "hidden_value?": false,
    "id": 3,
    "merge_default": false,
    "merge_overrides": false,
    "override": false,
    "override_value_order": "fqdn\nhostgroup\nos\ndomain",
    "override_values": [],
    "override_values_count": 0,
    "parameter": "content",
    "parameter_type": "string",
    "puppetclass_id": 3,
    "puppetclass_name": "motd",
    "required": false,
    "updated_at": "2017-02-07 11:56:55 UTC",
    "use_puppet_default": false,
    "validator_rule": null,
    "validator_type": ""
}
```

Change the desired item and remember to toggle the override value.
You should also remove values mentioning time values (created_at and updated_at in this example).
Lastly, remove the override_values item as it cannot be explicitly passed an empty value.

```
$ vi output_file.json
{
    "avoid_duplicates": false,
    "created_at": "2017-02-06 12:37:48 UTC",    ← Remove this line
    "default_value": "My new value for this item",    ← New value added to this item
    "description": "",
    "hidden_value": "*****",
    "hidden_value?": false,
    "id": 3,
    "merge_default": false,
    "merge_overrides": false,
    "override": true,    ← Toggled to true
    "override_value_order": "fqdn\nhostgroup\nos\ndomain",
    "override_values": [],    ← Remove this line
    "override_values_count": 0,
    "parameter": "content",
    "parameter_type": "string",
    "puppetclass_id": 3,
    "puppetclass_name": "motd",
    "required": false,
    "updated_at": "2017-02-07 11:56:55 UTC",    ← Remove this line
```

```
    "use_puppet_default": false,
    "validator_rule": null,
    "validator_type": ""
}
```

Save the changed file to another file then apply it to the satellite server:

```
$ curl -H "Accept:application/json,version=2" \
        -H "Content-Type:application/json" \
        -X PUT -u $user:$passwd \
        -d "$(< changed_file.json)"  \
        -k "https://<satellite_server>/api/smart_class_parameters/3
```

If an error is returned, remove from the json file the items that may be causing the error and try again.