Introduction to oVirt
RESTful API
SDK and CLI

01 Nov 2011

Michael Pasternak
HTTP Background

The Hypertext Transfer Protocol (HTTP) is a networking protocol for distributed, collaborative, hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web.

- HTTP is an Application Layer protocol
  (The protocol definitions presume a reliable Transport Layer)
- client-server computing model
- HTTP Resources are identified and located on the network by Uniform Resource Identifiers
- HTTP functions as a request-response
HTTP methods

- GET

Requests a representation of the specified resource. Requests using GET (and a few other HTTP methods) "SHOULD NOT have the significance of taking an action other than retrieval".

- HEAD

Asks for the response identical to the one that would correspond to a GET request, but without the response body. This is useful for retrieving meta-information written in response headers, without having to transport the entire content.
HTTP methods con.

- **POST**
  Submits data to be processed to the identified resource. The data is included in the body of the request.

- **PUT**
  Uploads a representation of the specified resource.

- **DELETE**
  Deletes the specified resource.

- **TRACE**
  Echoes back the received request, so that a client can see what (if any) changes or additions have been made by intermediate servers.
HTTP methods con.

- **OPTIONS**
  Returns the HTTP methods that the server supports for specified URL. This can be used to check the functionality of a web server by requesting '*' instead of a specific resource.

- **CONNECT**
  Converts the request connection to a transparent TCP/IP tunnel, usually to facilitate SSL-encrypted communication (HTTPS) through an unencrypted HTTP proxy.

- **PATCH**
  Is used to apply partial modifications to a resource.
HTTP response codes

- 1xx Informational
  - 102 Processing
  ...

- 2xx Success
  - 200 OK
  - 201 Created
  - 202 Accepted
  ...

- 3xx Redirection
  ...

- 4xx Client Error
  - 400 Bad Request
  - 401 Unauthorized
  - 404 Not Found
  ...

- 5xx Server Error
  - 500 Internal Server Error
  - 503 Service Unavailable
  ...

oVirt kick-off workshop
REST Background

- REST is Representational State Transfer
- The term Representational State Transfer was introduced and defined in 2000 by Roy Fielding in his doctoral dissertation

(Fielding is one of the principal authors of the Hypertext Transfer Protocol (HTTP) specification versions 1.0 and 1.1)
REST Concepts

- Client–server
- Stateless
- Cacheable
- Uniform interface
REST Concepts

- Identification of resources
- Manipulation of resources through representations
- Self-descriptive
- Hypermedia as the engine of application state

Clients make state transitions only through actions, a client does not assume that any particular actions will be available for any particular resources beyond those described in representations previously received from the server.
Media types

- **XML**

```xml
<vms>
  <vm id="xxx">
    <name>yyy</name>
  </vm>
</vms>
```

- **JavaScript Object Notation (JSON)**

```json
{
  "vms" : [
    "vm" : {
      "id" : "xxx",
      "name" : "yyy"
    }
  ]
}
```

- **YAML**

```yaml
- vms:
  - id: "xxx"
    name: yyyy
```
SOAP vs. REST

• REST advantages:
  - Lightweight - not a lot of extra xml markup
  - Human Readable Results
  - Easy to build - no toolkits required

• SOAP advantages:
  - Easy to consume (sometimes)
  - Rigid - type checking, adheres to a contract
  - Development tools
SOAP vs. REST

SOAP (WS-*)

| SMTP  | HTTP POST | MQ... |

1 Endpoint URI

Application

(Many) Resource URI

Application

HTTP GET

HTTP POST

HTTP PUT

HTTP DEL

AtomPub

JSON

POX...
oVirt-API as a RESTful API

- Container: JBOSS 5.1
- Framework: RESTeasy 2.2.2GA
- http(s)://server:port/api/
oVirt-API URI structure

http(s)://server:port/api/aaa/xxx-xxx/bbbb/yyy-yyy

1. protocol
2. server details
3. entry point (base resource)
4. collection
5. resource
6. sub-collection
7. sub-resource
oVirt-API How-to (the methods)

- To list all collection resources, use GET.
  
  GET http(s)://server:port/api/vms

- To retrieve specific resource, use GET.
  
  GET http(s)://server:port/api/vms/xxx

- To create a resource, use POST.
  
  POST http(s)://server:port/api/vms
  <vm>...</vm>

- To update the resource, use PUT.
  
  PUT http(s)://server:port/api/vms/xxx
  <vm><name>aaa</name></vm>

- To remove the resource, use DELETE.
  
  DELETE http(s)://server:port/api/vms/xxx
oVirt-API methods (behind the scene)

HTTP Client (Web Browser)

POST /order
301 Location: /order/612
PUT /order/612

Web Server Application Server

Database

SELECT *
FROM books
WHERE isbn=222

INSERT INTO orders

UPDATE orders
WHERE id=612
**Method::Any**
- Accept: application/xml, yaml, json (mandatory) *
- Authorization: Basic ... (mandatory)
- Accept-Language: de | nl | it

**Method::GET**
- details = statistics | disks | nics | tags ...

**Method::POST**
- Expect: 201-created
oVirt-API resource structure

GET http(s)://server:port/api/vms/xxx

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<vm id="xxx" href="/api/vms/xxx">  identification details
  <name>vm1_iscsi</name>
  <status>DOWN</status>
  <memory>10737418240</memory>
  <cpu>
    <topology cores="1" sockets="1"/>
  </cpu>
  <start_time>2011-07-13T12:05:34.931Z</start_time>
  <creation_time>2011-05-31T16:47:51+03:00</creation_time>
  <actions>
    <link rel="start" href="/api/vms/xxx/start"/>
    <link rel="stop" href="/api/vms/xxx/stop"/>
  </actions>
  <link rel="disks" href="/api/vms/xxx/disks"/>
  <link rel="nics" href="/api/vms/xxx/nics"/>
  <cluster id="zzz" href="/api/clusters/zzz"/>
  <template id="yyy" href="/api/templates/yyy"/>
</vm>
```

- **Metadata**
- **Resource details**
- **Actions**
- **Links to related resources**
Clients / Tools

- Any HTTP library/client can be used as a client for RHEVM-API
- Common used clients are:
  - FF REST Client
  - REST-Client (Google)
  - Linux: curl / wget
  ...
REST-Client (Google)
Examples

- GET http(s)://server:port/api

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<api>
    <link rel="capabilities" href="/api/capabilities"/>
    ...
    <link rel="vms" href="/api/vms" />
    <link rel="vms/search" href="/api/vms?search={query}" />
    <special_objects>
        <link rel="templates/blank" href="/api/templates/00000000-0000-0000-0000-000000000000" />
        <link rel="tags/root" href="/api/tags/00000000-0000-0000-0000-000000000000" />
    </special_objects>
    <product_info>
        <version revision="0" build="0" minor="0" major="3" />
    </product_info>
    <summary>
        <vms>
            <total>5</total>
            <active>1</active>
        </vms>
        <hosts>
            <total>3</total>
            <active>2</active>
        </hosts>
        <users>
            <total>6</total>
            <active>2</active>
        </users>
        <storage_domains>
            <total>7</total>
            <active>8</active>
        </storage_domains>
    </summary>
</api>
```
Example GET

Get:

GET http(s)://server:port/api/vms/xxx

Get with 'curl':

curl -v -u "user@domain:password" -H "Content-type: application/xml" -X GET http(s)://server:port/api/vms/xxx
Example CREATE

- **Create VM:**

  POST http(s)://server:port/api/vms
  <vm>
    <name>my_new_vm</name>
    <cluster id="xxx" />
    <template id="yyy" />
  </vm>

- **Create with 'curl'**

  curl -v -u "user@domain:password"
  -H "Content-type: application/xml"
  -d '<vm>
    <name>my_new_vm</name>
    <cluster><name>cluster_name</name></cluster>
    <template><name>template_name</name></template>
  </vm>'
  'http(s)://server:port/api/vms'
Example UPDATE

- **Update:**
  
  PUT http(s)://server:port/api/vms/xxx
  <vm>
    <name>new_name</name>
  </vm>

- **Update with 'curl':**
  
  echo "<vm><name>new_name</name></vm>" > /tmp/upload.xml
  curl -v -u "user@domain:password"
    -H "Content-type: application/xml"
    -T /tmp/upload.xml
  'http(s)://server:port/api/vms/xxx'
Example DELETE

- **Delete:**

  DELETE http(s)://server:port/api/vms/xxx

- **Delete with 'curl':**

  curl -v -u "user@domain:password" -X DELETE http(s)://server:port/api/vms/xxx
Python SDK: (The concepts):

- Complete protocol abstraction.
- Full compliance with the oVirt api architecture.
- Auto-completion.
- Self descriptive.
- Intuitive and easy to use.
- Auto-Generated.
RSDL:
(The RESTful Services Description Language)

- Why?
  - No way to know how to create the resource [1].
  - No way to know which actions available on collection [1].
  - No way to know which parameters to pass [1]:
    - mandatory/optional/read-only.
    - type.
    - overloads.
  - If resource is yet not created:
    - No way to know which actions available on it [1].
    - No way to know which sub-collections available [1].
    - No way to know how the resource representation
      looks like [1].
[1] other than reading documentation.
RSDL:
(RESTful Services Description Language)

- How?

GET: http(s)://server:port/api?rsdl

```xml
<link rel="get" href="/api/clusters">
  <request>
    <http_method>GET</http_method>
  </request>
  <response>
    <type>Clusters</type>
  </response>
</link>
<link rel="add" href="/api/clusters/{cluster:id}/permissions">
  <request>
    <http_method>POST</http_method>
    <body>
      <type>Permission</type>
    </body>
  </request>
  <response>
    <type>Permission</type>
  </response>
</link>
<link rel="delete" href="/api/clusters/{cluster:id}/permissions/{permission:id}"
  <request>
    <http_method>DELETE</http_method>
  </request>
</link>
```
This XML file does not appear to have any style information associated with it. The document tree is shown below.

```xml
<rsdl href="/api/rsdl">
  <description>The oVirt RESTful API description language.</description>
  <link rel="get" href="/api/capabilities">
    <request>
      <http_method>GET</http_method>
    </request>
    <response>
      <type>Capabilities</type>
    </response>
  </link>
  <link rel="add" href="/api/clusters">
    <request>
      <http_method>POST</http_method>
    </request>
    <body>
      <type>Cluster</type>
    </body>
    <response>
      <type>Cluster</type>
    </response>
  </link>
  <link rel="delete" href="/api/clusters/{cluster:id}">
    <request>
      <http_method>DELETE</http_method>
    </request>
  </link>
  <link rel="get" href="/api/clusters">
    <request>
      <http_method>GET</http_method>
    </request>
  </link>
</rsdl>
```
Python SDK: (Usage)

- Creating the proxy
  ```python
  api = API(url='http://localhost:8080', username='user@domain', password='password')
  ```

- Listing all collections
  ```python
  api.vms
  ```

- Listing collection's methods
  ```python
  api.vms.list
  ```

- Querying collection with oVirt search engine.
  ```python
  vms = api.vms.list(query = 'name=python_vm')
  ```

- Querying collection by custom constraint.
  ```python
  vms = api.vms.list(memory=1073741824)
  ```

- Querying collection for specific resource.
  ```python
  vm = api.vms.get(id = '02f04a4-9738-4731-83c4-293f3f734782')
  ```

- Accessing resource methods and properties.
  ```python
  vm.start()
  ```
Python SDK: (Usage)

- Accessing resource properties and sub-collections.

- Accessing sub-collection methods.

- Querying sub-collection by custom constraint.

- Retrieving sub-collection resource.

- Accessing sub-collection resource properties and methods.

```python
vm.n

# updatere source

vm.nics.

nics = vm.nics.list(interface='e1000')

nic = vm.nics.get(name='eth0')

nic.update()
```
AVAILAIBLE COMMANDS

* action    execute an action on an object
* cd        change directory
* clear     clear the screen
* connect   connect to a RHEV manager
* console   open a console to a VM
* create    create a new object
* delete    delete an object
* disconnect disconnect from RHEV manager
* exit      quit this interactive terminal
* getkey    dump private ssh key
* help      show help
* list      list or search objects
* ping      test the connection
* pwd       print working directory
* save      save configuration variables
* set       set a configuration variable
* show      show one object
* status    show status
* update    update an object

(oVirt cli) > help connect

USAGE

    connect
    connect <url> <username> <password>

DESCRIPTION

Connect to a RHEV manager. This command has two forms. In the first form, no arguments are provided, and the connection details are re-read from their respective configuration variables (see 'show'). In the second form, the connection details are provided as arguments.

The arguments are:

* url    - The URL to connect to.
* username    - The user to connect as. Important: this needs to be in the user@domain format.
* password    - The password to use.
CLI – querying for resources

Usage

list <type> [search]... [object identifiers]

Description

List or search for objects of a certain type. There are two forms. If only <type> is provided, all objects of the specified type are returned. If a search query is given, it must be a valid RHEV-M search query. In that case objects matching the query are returned.

Available Types

The <type> parameter must be one of the following. Note: not all types implement search!

* capabilities
* clusters
* datacenters
* events
* hosts
* networks
* roles
* storagedomains
* tags
* templates
* users
* vmpools
* vms

```
(oVirt cli) > list vms
name     status
--------- ------
new_vm   down
python_vm unknown
vm1_nfs  down
vmpool-1 down
vm_test  down
vm_test2 down

(oVirt cli) > show vm new_vm
id         : 62004129-a806-4e48-9b39-f6a54c97cba6
name       : new_vm
status     : down
memory     : 1024
os         : unassigned
display    : spice
monitors   : 1
stateless  : False
template   : 94f5ad88-a12a-4f48-af9f-f2ba28b7285b
cluster    : 99408929-82cf-4dc7-a532-9d998063fa95
```
CLI – create

$ create vm --name myvm --memory 512 --type SERVER \ 
   --cluster Default --template Blank

This example does the same but now using pre-formatted input:

$ create vm << EOM
> <vm>
>   <name>myvm</name>
>   <memory>512000000</memory>
>   <type>SERVER</type>
>   <cluster><name>Default</name></cluster>
>   <template><name>Blank</name></template>
> </vm>
> EOM

(oVirt cli) > create cluster --name test
error: rhev: Incomplete parameters
detail: Cluster [dataCenter.name|id] required for add

(oVirt cli) >
(oVirt cli) >
(oVirt cli) > create vm --name test --cluster Default_nfs --template Blank
error: rhev: Operation Failed
detail: Storage Domain cannot be accessed.
### CLI – update

```bash
(oVirt cli) > list clusters

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7073b1ac-ef46-11e0-aa7c-d3e6f6b5731d</td>
<td>aa</td>
<td></td>
</tr>
<tr>
<td>80eed02c-ac7d-11e0-b702-0bf21e6d33af</td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>82b1c018-ac7d-11e0-ac42-5b88dcd7c92</td>
<td>c</td>
<td>default_iscsi</td>
</tr>
<tr>
<td>63bc09b0-8b8b-11e0-bdc2-4356942887b3</td>
<td>Default_nfs</td>
<td>The default server cluster</td>
</tr>
<tr>
<td>99408929-82cf-4dc7-a532-9d998063fa95</td>
<td>Test_iscsi</td>
<td></td>
</tr>
<tr>
<td>ffb2d112-8cf0-11e0-b34b-7f61455e6a71</td>
<td>Test_iscsi</td>
<td></td>
</tr>
<tr>
<td>ada1672a-8cf1-11e0-9d3e-b75c5a33ec19</td>
<td>Test_nfs</td>
<td></td>
</tr>
<tr>
<td>ad9bd996-a893-11e0-b174-e3232e67a091</td>
<td>Test_vlans</td>
<td></td>
</tr>
</tbody>
</table>

(oVirt cli) > update cluster aa --name bb

(oVirt cli) > list clusters

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>80eed02c-ac7d-11e0-b702-0bf21e6d33af</td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>7073b1ac-ef46-11e0-aa7c-d3e6f6b5731d</td>
<td>aa</td>
<td></td>
</tr>
<tr>
<td>82b1c018-ac7d-11e0-ac42-5b88dcd7c92</td>
<td>c</td>
<td>default_iscsi</td>
</tr>
<tr>
<td>63bc09b0-8b8b-11e0-bdc2-4356942887b3</td>
<td>Default_nfs</td>
<td>The default server cluster</td>
</tr>
<tr>
<td>99408929-82cf-4dc7-a532-9d998063fa95</td>
<td>Test_iscsi</td>
<td></td>
</tr>
<tr>
<td>ffb2d112-8cf0-11e0-b34b-7f61455e6a71</td>
<td>Test_iscsi</td>
<td></td>
</tr>
<tr>
<td>ada1672a-8cf1-11e0-9d3e-b75c5a33ec19</td>
<td>Test_nfs</td>
<td></td>
</tr>
<tr>
<td>ad9bd996-a893-11e0-b174-e3232e67a091</td>
<td>Test_vlans</td>
<td></td>
</tr>
</tbody>
</table>
```
CLI – delete

(oVirt cli) > list clusters

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>80eed02c-ac7d-11e0-b702-0bf21e6d33af</td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>7073b1ac-ef46-11e0-aa7c-d3e6f6b5731d</td>
<td>bb</td>
<td></td>
</tr>
<tr>
<td>82b1c018-ac7d-11e0-ac42-5b8d8dcd7c92</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>63bc09b0-8b8b-11e0-bd52-4356942887b3</td>
<td>Default_iscsi</td>
<td></td>
</tr>
<tr>
<td>99408929-82cf-4dc7-a532-9d99063fa95</td>
<td>Default_nfs</td>
<td>The default server cluster</td>
</tr>
<tr>
<td>ffb2d112-8cf0-11e0-b34b-7f61455e6a71</td>
<td>Test_iscsi</td>
<td></td>
</tr>
<tr>
<td>ada1672a-8cf1-11e0-9d3e-b75c5a33ec19</td>
<td>Test_nfs</td>
<td></td>
</tr>
<tr>
<td>ad9bd996-a893-11e0-b174-e3232e67a091</td>
<td>Test_vlans</td>
<td></td>
</tr>
</tbody>
</table>

(oVirt cli) > delete cluster bb

(oVirt cli) > list clusters

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>80eed02c-ac7d-11e0-b702-0bf21e6d33af</td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>82b1c018-ac7d-11e0-ac42-5b8d8dcd7c92</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>63bc09b0-8b8b-11e0-bd52-4356942887b3</td>
<td>Default_iscsi</td>
<td></td>
</tr>
<tr>
<td>99408929-82cf-4dc7-a532-9d99063fa95</td>
<td>Default_nfs</td>
<td>The default server cluster</td>
</tr>
<tr>
<td>ffb2d112-8cf0-11e0-b34b-7f61455e6a71</td>
<td>Test_iscsi</td>
<td></td>
</tr>
<tr>
<td>ada1672a-8cf1-11e0-9d3e-b75c5a33ec19</td>
<td>Test_nfs</td>
<td></td>
</tr>
<tr>
<td>ad9bd996-a893-11e0-b174-e3232e67a091</td>
<td>Test_vlans</td>
<td></td>
</tr>
</tbody>
</table>
CLI - action

**USAGE**

```
action <type> <id> <action> [base identifiers] [attribute options]
```

**DESCRIPTION**

Executes the an action on an object. This command requires the following arguments:

* `type` - The type to operate on
* `id` - The name or id identifying the object
* `action` - The action to take

For more specific help on the available actions and options, use
`'help action <type> <id>'`

**AVAILABLE TYPES**

The `<type>` parameter must be one of the following:

* `cluster`
* `datacenter`
* `event`
* `host`
* `network`
* `role`
* `storagedomain`
* `tag`
* `template`
* `user`
* `vm`
* `vmpool`

**RETURN VALUES**

This command will return one of the following statuses. To see the exit status of the last command, type 'status'.

* 000 (OK)
* 001 (SYNTAX_ERROR)
* 002 (COMMAND_ERROR)
* 003 (INTERRUPTED)
* 004 (UNKNOWN_ERROR)
* 010 (REMOTE_ERROR)
* 011 (NOT_FOUND)

```bash
(oVirt cli) > show vm new_vm
id : 62004129-a806-4e48-9b39-f6a54c97cbe6
name : new_vm
status : down
memory : 1024
os : unassigned
display : spice
monitors : 1
stateless : False
template : 94f5ad88-a12a-4f48-af9f-f2ba28b7285b
cluster : 99408929-82cf-4dc7-a532-9d998063fa95
```

```bash
(oVirt cli) > action vm new_vm start
status: complete
```

```bash
(oVirt cli) > show vm new_vm
id : 62004129-a806-4e48-9b39-f6a54c97cbe6
name : new_vm
status : powering_up
memory : 1024
os : unassigned
display : spice
monitors : 1
stateless : False
template : 94f5ad88-a12a-4f48-af9f-f2ba28b7285b
cluster : 99408929-82cf-4dc7-a532-9d998063fa95
```
What next?

- Non-Admin users support
- Actions on Collection (atomic network operations)
- Pagination on collections
- Async update/delete
- Exposing additional oVirt search capabilities
- SDKs (C# / Ruby / Delphi / Java / ...)
- Clients (PowerShell / ...)

oVirt kick-off workshop
New oVirt engine features

- Quota
- New networking capabilities (bridgeless)
- Multiple storage domains
- Backup API
- Full support for Async tasks

...
THANK YOU!

Wiki: http://ovirt.org/wiki/Category:Api
ML: engine-devel
GIT: git://gerrit.ovirt.org/ovirt-engine-sdk