OnApp API Guide

v.2.0

Table of Contents

Introduction

OnAPP API is RESTful.
All function calls respond to xml and JSON requests.
All function calls need authentication (Basic HTTP).

Users

Get the list of users

GET onapp.com/users.xml
GET onapp.com/users.json
Outputs all existed users with their details.

Get a particular user details

GET onapp.com/users/{ID}.xml
GET onapp.com/users/{ID}.json

Example:
curl -X GET -H 'Accept: application/json' -H 'Content-type: application/json' -u user:pass http://demo.onapp.com/users/164.json

Outputs:


```
{"user":{"disk_space_available":54,"used_cpus":0,"created_at":"2010-08-
17T19:25:05+03:00","activated_at":"2010-08-17T19:25:05+03:00","used_cpu_shares":0,"roles":
[{"role":{"label":"User","created_at":"2010-05-26T16:34:58+03:00","updated_at":"2010-08-
20T17:35:45+03:00","id":2,"identifier":"user"}},{"role":{"label":"lvtestrole", "created_at":
"2010-07-19T13:57:59+03:00","updated_at":"2010-07-
20T12:04:04+03:00","id":11,"identifier":"r2lmvvus6fo3as"}}],"remember_token_expires_at":null
,"used_ip_addresses":[],"updated_at":"2010-08-19T12:31:39+03:00", "deleted_at":null,
"used_memory":0,"activation_code":null,"id":164,"group_id":1,"memory_available":512,"remembe
r_token":null,"last_name":"Last","time_zone":null,"used_disk_size":0,"total_amount":"423472.
0","payment_amount":"0.0","login":"test005_4@test.com","outstanding_amount":"423472.0","firs
t_name":"First","email":"apidoc@onapp.com"}}
```


Where:
created_at: the date in the [YYYY][MM][DD]T[hh][mm]Z format
activated_at: the date when the User was activated in the [YYYY][MM][DD]T[hh][mm]Z format
crypted_password: the user password
updated_at: the date when the User was updated in the [YYYY][MM][DD]T[hh][mm]Z format
group_id: the ID of the group this group belongs to

Note: *When user with requested ID is not found, HTTP 404 response returns.*

Example:

```
curl -X POST -d "{user:{login:'theone', email:'theone@onapp.com',
password_confirmation:'H7YgiU6B', first_name:'Joe', last_name:'Doe',
password:'H7YgiU6B', group_id:10 }}" -H 'Accept: application/json' -H 'Content-
type: application/json' -u user:pass http://demo.onapp.com/users.json
```

Returns HTTP 200 on successful creation, or HTTP 422 if a user with such login/email already exists.

Delete an existing user

```
DELETE onapp.com/users/{ID}.xml
DELETE onapp.com/users/{ID}.json
```

Example:

```
curl -X DELETE -H 'Accept: application/json' -H 'Content-type: application/json' -u
user:pass http://demo.onapp.com/users/101.json
```

Returns HTTP 200 response on successful deletion; HTTP 404 response when a user with provided ID is not found.

User's limits

Get user's limits

```
GET onapp.com/users/{ID}/resource_limit.xml
GET onapp.com/users/{ID}/resource_limit.json
```

Example:

```
curl -i -X GET -u user:pass http://demo.onapp.com/users/164/resource_limit.json
```

Outputs:

```
{"resource_limit":{"cpus":4,"created_at":"2010-08-17T19:25:05+03:00", "cpu_shares":100,
"updated_at":"2010-08-17T19:26:16+03:00","memory":512, "disk_size":54,"id":191,
"user_id":164}}
```

Set user's limits

```
PUT onapp.com/users/{ID}/resource_limit.xml
PUT onapp.com/users/{ID}/resource_limit.json
```

Example:

```
curl -i -X PUT -u user:pass http://demo.onapp.com/users/1/resource_limit.json -d
"{'resource_limit':{'memory':256, 'cpus':2, 'cpu_shares':10, 'disk_size':6}}" -H
'Accept: application/json' -H 'Content-type: application/json'
```

Returns HTTP 200 response on successful limits setting; HTTP 404 response when a user with

provided ID is not found.

Get user's limits per hypervisor

```
GET onapp.com/users/{USER_ID}/hypervisors/{ID}/limits.xml
GET onapp.com/users/{USER_ID}/hypervisors/{ID}/limits.json
GET onapp.com/users/{USER_ID}/hypervisors/limits.xml
GET onapp.com/users/{USER_ID}/hypervisors/limits.json
```

Outputs the limitations which a particular hypervisor is reached.
If no hypervisor ID is provided, the maximum hypervisor limits are output.

Examle:
`curl -u user:pass -G http://demo.onapp.com/users/170/hypervisors/1/limits.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<limits>
  <cpu-shares type="integer">12</cpu-shares>
  <cpu type="integer">12</cpu>
  <memory type="integer">1000</memory>
  <disk-space type="integer">585</disk-space>
</limits>
```

Billing Groups

Get the list of groups

```
GET onapp.com/groups.xml
GET onapp.com/groups.json
```

XML Output Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<groups type="array">
  <group>
    <label>Test Group'</label>
    <created_at type="datetime">2010-06-30T09:24:46Z</created_at>
    <updated_at type="datetime">2010-08-04T13:37:11Z</updated_at>
    <price_disk_size type="decimal">12.0</price_disk_size>
    <id type="integer">1</id>
    <price-memory type="decimal">42.0</price_memory>
    <identifier nil="true"></identifier>
    <price_cpu type="decimal">34.0</price_cpu>
    <price_cpu_share type="decimal">32.0</price_cpu_share>
  </group>
```

Where:
created_at: the date in the [YYYY][MM][DD]T[hh][mm]Z format
updated_at: the date when the Group was updated in the [YYYY][MM][DD]T[hh][mm]Z format
price_disk_size: the price for the Disk size per hour per GB
price-memory: the price for the Memory per hour per MB

price_cpu: the price for the CPU usage per hour per CPU core
price_cpu_share: the price for the CPU Priority per hour per CPU Priority

Get a particular group details

```
GET onapp.com/groups/{ID}.xml
GET onapp.com/groups/{ID}.json
```

XML Output Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<group>
  <label>Edit Test Group</label>
  <created_at type="datetime">2010-06-30T09:24:46Z</created_at>
  <updated_at type="datetime">2010-08-04T13:37:11Z</updated_at>
  <price_disk_size type="decimal">12.0</price_disk_size>
  <id type="integer">1</id>
  <price-memory type="decimal">42.0</price_memory>
  <identifier nil="true"></identifier>
  <price_cpu type="decimal">34.0</price_cpu>
  <price_cpu_share type="decimal">32.0</price_cpu_share>
</group>
```

Add new group

```
POST onapp.com/groups.xml
POST onapp.com/groups.json
```

Example:

```
 <?xml version="1.0" encoding="UTF-8"?>
 <group>
    <label>{LABEL}</label>
    <price_cpu>{PRICE1}</price_cpu>
    <price_cpu_share>{PRICE2}</price_cpu_share>
    <price_disk_size>{PRICE3}</price_disk_size>
    <price_memory>{PRICE4}</price_memory>
 </group>
```

Edit existing group

```
PUT onapp.com/groups/{ID}.xml
PUT onapp.com/groups/{ID}.json
```

Example:

```
 <?xml version="1.0" encoding="UTF-8"?>
 <group>
     <label>{LABEL}</label>
     <price_cpu>{PRICE}</price_cpu>
     <price_cpu_share>{PRICE}</price_cpu_share>
     <price_disk_size>{PRICE}</price_disk_size>
     <price_memory>{PRICE}</price_memory>
 </group>
```

Delete existing group

```
DELETE onapp.com/groups/{ID}.xml
DELETE onapp.com/groups/{ID}.json
```

Roles

This class represents the roles assigned to the users in this OnApp installation. With our API you can get the list of roles, get a particular role details, and add a new role.

Get the roles list

This method gets the list of all the roles available in the system.

```
GET onapp.com/roles.xml
GET onapp.com/roles.json
```

XML Output Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<roles>
  <role>
    <label>Administrator</label>
    <created_at>2010-05-26T13:34:58Z</created_at>
    <updated_at>2010-07-18T21:16:14Z</updated_at>
    <id>1</id>
    <identifier>admin</identifier>
    <permissions>
      <permission>
        <label>Any action on virtual machines</label>
        <created_at>2010-05-26T13:34:58Z</created_at>
        <updated_at>2010-05-26T13:34:58Z</updated_at>
        <id>1</id>
        <identifier>virtual_machines</identifier>
      </permission>
    </permissions>
  </role>
</roles>
```

Where:
created_at: the date in the [YYYY][MM][DD]T[hh][mm]Z format
updated_at: the date when the Group was updated in the [YYYY][MM][DD]T[hh][mm]Z format
permission: the permissions assigned to this role with their Label and ID.

Get a particular role details

This method will output the details for a particular user role.

```
GET onapp.com/roles/{ID}.xml
GET onapp.com/roles/{ID}.json
```

XML Output Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<role>
```

```
      <identifier>{IDENTIFIER}</identifier>
      <label>{LABEL}</label>
</role>
```

Where:
Identifier – the role Identifier
Label – the role label

Note: *The role for a particular user is outputs on /users/{ID} request*

Edit existing role

```
PUT onapp.com/settings/roles/{ID}.xml
PUT onapp.com/settings/roles/{ID}.json
```

XML Output Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<role>
      <identifier>{IDENTIFIER}</identifier>
      <label>{LABEL}</label>
</role>
```

Where:
Identifier – the role Identifier
Label – the role label

Add a new role

```
POST onapp.com/settings/roles.xml
POST onapp.com/settings/roles.json
```

The following parameters should be sent:
label – the new role label (required)
identifier - the new role ID (required)
permission-id – the ID of the permission you would like to assign this role with (optional)

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<role>
  <identifier>test12312345</identifier>
  <label>LABELtest12345</label>
  <permission-ids type="array">
      <permission_id>48</permission_id>
  </permission_ids>
</role>
```

Delete role

```
DELETE onapp.com/settings/roles/{ID}.xml
DELETE onapp.com/settings/roles/{ID}.json
```

Edit a user role

You can set a list of role ids to change a role(s) for an existed user. E.g.:

```
curl -X PUT -H 'Accept: application/json' -H 'Content-type: application/json' -u
user:pass http://demo.onapp.com/users/9.json -d "{'user':{role_ids:[2,3]}}"
```

Returns HTTP 200 response if roles are changed, HTTP 404 if no user found.

Virtual Machines

Get the list of VMs

```
GET onapp.com/virtual_machines.xml
GET -H 'Accept: application/json' -H 'Content-type: application/json'
onapp.com/virtual_machines.json
```

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<virtual_machines type="array">
  <virtual_machine>
  <cpus>2</cpus>
  <label>centos-5.5-x64</label>
  <created_at>2010-07-12T12:31:41+03:00</created_at>
  <operating_system_distro>rhel</operating_system_distro>
  <cpu_shares>100</cpu_shares>
  <operating_system>linux</operating_system>
  <template_id>14</template_id>
  <allowed_swap>true</allowed_swap>
  <local_remote_access_port>5917</local_remote_access_port>
  <memory>256</memory>
  <updated_at>2010-08-19T15:51:47+03:00</updated_at>
  <allow_resize_without_reboot>true</allow_resize_without_reboot>
  <recovery_mode nil="true"></recovery_mode>
  <hypervisor_id>1</hypervisor_id>
  <id>206</id>
  <xen_id>292</xen_id>
  <user_id>1</user_id>
  <booted>true</booted>
  <hostname>centos-5-5-x64</hostname>
  <template_label>CentOS 5.5 x64</template_label>
  <identifier>qlzurxiu5e6hd7</identifier>
  <initial_root_password>6omu54lvvi6i</initial_root_password>
  <min_disk_size>5</min_disk_size>
  <remote_access_password>zm9cal</remote_access_password>
  <built>true</built>
  <locked>false</locked>
  <ip_addresses>
    <ip_address>
```

```
      <address>109.123.91.94</address>
      <netmask>255.255.255.192</netmask>
      <created_at>2010-04-27T19:58:01+03:00</created_at>
      <broadcast>109.123.91.127</broadcast>
      <network_address>109.123.91.64</network_address>
      <network_id>1</network_id>
      <updated_at>2010-04-27T19:58:01+03:00</updated_at>
      <id>29</id>
      <gateway>109.123.91.65</gateway>
    </ip_address>
  </ip_addresses>
  <monthly_bandwidth_used>0</monthly_bandwidth_used>
</virtual_machine>
</virtual-machines>
```

Where:
booted - true if booted. Otherwise false
built - true if built. Otherwise false
cpus - the number of CPUs
cpu_shares - the number of CPU Shares
created_at - the date in the [YYYY][MM][DD]T[hh][mm]Z format
hostname - the name of your host
hypervisor_id - the ID of the hypervisor used by this VM
id - the virtual machine ID
identifier - the VM identifier
initial_root_password - the VM root password
label - the VM label
local_remote_access_port - the port ID used for console access
locked - true if the VM is locked. Otherwise false
memory - the memory size
monthly_bandwidth_used - the bandwidth used this month
primary_disk_size - the size of the primary disk
recovery_mode - true if recovery mode allowed. Otherwise false
remote_access_password - the password for the remote access
swap_disk_size - the size of the swap disk
template_id - the ID of the template the VM is based on
updated_at - the date when the Virtual Machine was updated in the [YYYY][MM][DD]T[hh][mm]Z format
xen_id - the ID of the Xen virtualizing this VM

Get a particular VM details

```
GET onapp.com/virtual_machines/{ID}.xml
GET onapp.com/virtual_machines/{ID}.json
```

Shows the same attributes of VM described in 'Get the list of VMs' request.

Create a new VM

```
POST onapp.com/virtual_machines.xml
POST -H 'Accept: application/json' -H 'Content-type: application/json'
onapp.com/virtual_machines.json
```

The following parameters should be sent:

```
memory – amount of RAM assigned
cpus –  number of CPUs
label – user-friendly description
template_id
hypervisor_id
initial_root_password
```

To build a VM automatically after creation, set required_virtual_machine_build to 1.

Json Request Example:

```
{"virtual_machine":{"primary_network_id":"1", "cpus":"1", "label":"Pizza",
"cpu_shares":"10", "template_id":"1", "swap_disk_size":"1", "memory":"256",
"required_virtual_machine_build":"1", "hypervisor_id":"1",
"required_ip_address_assignment":"1", "rate_limit":"10", "primary_disk_size":"5",
"hostname":"pizza.it", "initial_root_password":"w1thp1neapplessl1ces"} }
```

XML Request Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<virtual_machine>
  <cpu_shares>{NUMBER}</cpu_shares>
  <cpus>{NUMBER}</cpus>
  <hostname>{HOSTNAME}</hostname>
  <hypervisor_id>{ID}</hypervisor_id>
  <initial_root_password>{PASSWORD}</initial_root_password>
  <memory>{SIZE}</memory>
  <template_id>{ID}</template_id>
  <primary_disk_size>{SIZE}</primary_disk_size>
  <swap_disk_size>{SIZE}</swap_disk_size>
</virtual_machine>
```

XML output example:

```
<?xml version="1.0" encoding="UTF-8"?>
<radar_virtual_machine>
  <booted type="boolean">false</booted>
  <built type="boolean">false</built>
  <cpu-shares type="integer" nil="true"></cpu_shares>
  <cpus type="integer">1</cpus>
  <created_at type="datetime">2010-05-20T15:42:45Z</created_at>
  <hostname>qwerty</hostname>
  <hypervisor-id type="integer">1</hypervisor_id>
  <id type="integer">17</id>
  <identifier>1qrev5r4gn2koz</identifier>
  <initial_root_password>qwerty</initial_root_password>
  <label>qwerty label</label>
  <local_remote_access_port type="integer" nil="true"></local_remote_access_port>
  <locked type="boolean">false</locked>
  <memory type="integer">100</memory>
  <recovery-mode type="boolean" nil="true"></recovery_mode>
  <remote_access_password nil="true"></remote_access_password>
  <template-id type="integer">3</template_id>
  <updated_at type="datetime">2010-05-20T15:42:45Z</updated_at>
  <xen-id type="integer" nil="true"></xen_id>
</radar_virtual_machine>
```

Edit VM parameters

```
PUT onapp.com/virtual_machines/{ID}.xml
```

```
PUT -H 'Accept: application/json' -H 'Content-type: application/json'
onapp.com/virtual_machines/{ID}.json
```

The parameters are the same as you can post with the Create function.
If a VM was modified successfully, an HTTP 200 response is returned.

Destroy VM

```
DELETE onapp.com/virtual_machines/{ID}.xml
DELETE -H 'Accept: application/json' -H 'Content-type: application/json'
onapp.com/virtual_machines/{ID}.json
```
On successful deletion an HTTP 200 response is returned.
An HTTP 404 is returned if VM's ID is not found.

Transactions (aka Logs)

This class represents the Transactions of the OnApp installation.

Get the list of transaction

```
GET onapp.com/transactions.xml
GET onapp.com/transactions.json
```

```
Json Output Example:
{"transaction":{"pid":26883,"created_at":"2010-08-09T08:29:21Z","updated_at":"2010-08-
09T08:29:39Z","actor":null,"priority":10,"parent_type":"VirtualMachine","action":"reboot_vir
tual_machine","id":4372,"dependent_transaction_id":null,"parent_id":367,"params":
{},"log_output":"Running: xm shutdown evnddau8y146ek\n", "status":"complete"}
```

Get a particular transaction details

```
GET onapp.com/transactions/{ID}.xml
GET onapp.com/transactions/{ID}.json
```

Json Output Example:
```
{"pid":22006,"created_at":"2010-05-06T12:17:48Z","updated_at":"2010-05-
06T12:17:52Z","actor":null,"priority":10,"parent_type":"Radar::VirtualMachine","action":"sto
p_virtual_machine","id":102,"dependent_transaction_id":null,"parent_id":3,"params":
{},"log_output":"Running: xm destroy 0frl82wp533d3e\n","status":"complete"}
```

Networks

All CRUD operations are possible for a particular network (requests and responses are similar to VMs)
For destroy, edit, or update operations an ID is a must.

Get the list of networks available

```
GET onapp.com/settings/networks.xml
GET onapp.com/settings/networks.json
```

XML Output Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<radar-networks type="array">
  <radar_network>
    <created_at type="datetime">2010-04-27T16:11:26Z</created_at>
    <id type="integer">1</id>
    <identifier>d3ily2rrz33nb9</identifier>
    <label>Public VLAN</label>
    <updated_at type="datetime">2010-04-27T16:11:26Z</updated_at>
    <vlan type="integer" nil="true"></vlan>
  </radar_network>
  <radar_network>
    <created_at type="datetime">2010-05-19T07:30:36Z</created_at>
    <id type="integer">3</id>
    <identifier>doq4dc2vxlf41v</identifier>
    <label>VLAN 11</label>
    <updated_at type="datetime">2010-05-19T07:30:36Z</updated_at>
    <vlan type="integer">11</vlan>
  </radar_network>
</radar_networks>
```

Where:
created_at - the date in the [YYYY][MM][DD]T[hh][mm]Z format
id - the network ID
label - the optional Network label
updated_at - the date when the Network was updated in the [YYYY][MM][DD]T[hh][mm]Z format
vlan - the VLAN this network belongs to

Edit network parameters

```
PUT onapp.com/settings/networks/{ID}.xml
PUT onapp.com/settings/networks/{ID}.json
```

XML Output Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<network>
  <label>DEV1 Public 311</label>
  <created_at type="datetime">2010-04-27T16:11:26Z</created_at>
  <updated_at type="datetime">2010-05-27T12:10:04Z</updated_at>
  <id type="integer">1</id>
  <vlan type="integer">311</vlan>
  <identifier>d3ily2rrz33nb9</identifier>
</network>
```

Add a new network

```
POST onapp.com/settings/networks.xml
POST onapp.com/settings/networks.json
```

XML Output Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<network>
  <label>{Network_Label}</label>
  <vlan>{VLAN}</vlan>
  <identifier>{ID}</identifier>
</network>
```

Delete a network

```
DELETE onapp.com/settings/networks/{ID}.xml
DELETE onapp.com/settings/networks/{ID}.json
```

IP Addresses

Edit IP

```
PUT onapp.com/settings/networks/{NETWORK_ID}/ip_addresses/{ID}/edit.xml
PUT onapp.com/settings/networks/{NETWORK_ID}/ip_addresses/{ID}/edit.json
```

The following parameters can be passed to be changed:
address, netmask, broadcast, network_address, gateway (all are strings)

Example XML Output:

```
<?xml version="1.0" encoding="UTF-8"?>
<ip_address>
  <address>109.123.91.66</address>
  <netmask>255.255.255.192</netmask>
  <created_at type="datetime">2010-04-27T16:58:01Z</created_at>
  <broadcast>109.123.91.127</broadcast>
  <network_address>109.123.91.64</network_address>
  <network-id type="integer">1</network_id>
  <updated_at type="datetime">2010-04-27T16:58:01Z</updated_at>
  <id type="integer">1</id>
  <gateway>109.123.91.65</gateway>
</ip_address>
```

Returns HTTP 200 on success

Note: *You can get the list of IPs assigned to a VM with GET /virtual_machines/{ID} request.*

Create a new IP address record

```
POST onapp.com/settings/networks/{NETWORK_ID}/ip_addresses.xml
POST onapp.com/settings/networks/{NETWORK_ID}/ip_addresses.json
```

Parameters are: address, netmask, broadcast, network_address, gateway (all required)

XML Output Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<ip-addresses type="array">
  <ip_address>
```

```
    <address>109.123.91.66</address>
    <netmask>255.255.255.192</netmask>
    <created_at type="datetime">2010-04-27T16:58:01Z</created_at>
    <broadcast>109.123.91.127</broadcast>
    <network_address>109.123.91.64</network_address>
    <network-id type="integer">1</network_id>
    <updated_at type="datetime">2010-04-27T16:58:01Z</updated_at>
    <id type="integer">1</id>
    <gateway>109.123.91.65</gateway>
  </ip_address>
</ip_addresses>
```

Destroy IP

```
DELETE onapp.com/settings/networks/{NETWORK_ID}/ip_addresses/{ID}.xml
DELETE onapp.com/settings/networks/{NETWORK_ID}/ip_addresses/{ID}.json
```

Hypervisors

Get the list of available hypervisors

```
GET onapp.com/settings/hypervisors.xml
GET onapp.com/settings/hypervisors.json
```

Returns the array of available hypervisors.

Get a particular hypervisor details

```
GET onapp.com/settings/hypervisors/{ID}.xml
GET onapp.com/settings/hypervisors/{ID}.json
```

XML Output Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<hypervisor>
  <called_in_at type="datetime">2010-08-09T12:55:01Z</called_in_at>
  <label>HV1</label>
  <created_at type="datetime">2010-04-27T15:34:11Z</created_at>
  <spare type="boolean">false</spare>
  <updated_at type="datetime">2010-08-09T12:55:04Z</updated_at>
  <id type="integer">1</id>
  <xen-info type="yaml" nil="true"></xen_info>
  <failure-count type="integer">0</failure_count>
  <health>
    <xm_info>{XM Info}</xm_info>
    <xm_list>{XM List}</xm_list>
    <vgdisplay>{VG Display}</vgdisplay>
    <uptime>13:54:55 up 32 days, 23:56,  1 user,  load average: 0.01, 0.45, 0.58</uptime>
  </health>
  <memory-overhead type="integer">464</memory_overhead>
  <ip_address>{IP_Address}</ip_address>
  <locked type="boolean">false</locked>
```

```
  <online type="boolean">true</online>
</hypervisor>
```

Where:
called_in_at - the date when the hypervisor was called in the [YYYY][MM][DD]T[hh][mm]Z format
created_at - the date in the [YYYY][MM][DD]T[hh][mm]Z format
failure_count - the number of failures
health - the array of the xm_info, disk, memory, and xm_list variables
id - the Hypervisor ID
ip_address - the Hypervisor IP address
label - the Hypervisor Label
locked - true if the Hypervisor is locked, otherwise false
memory_overhead - shows the memory overhead
online - true if online, otherwise false
spare - true if spare, otherwise false
updated_at - the date when the Group was updated in the [YYYY][MM][DD]T[hh][mm]Z format
xen_info - the info on the Xen

Add new hypervisor

```
POST onapp.com/settings/hypervisors.xml
POST onapp.com/settings/hypervisors.json
```

To add a new hypervisor, send the following parameters:
ip-address – the Hypervisor ID
label – the Hypervisor label

XML Output Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<hypervisor>
  <ip_address>{IP}</ip_address>
  <label>{LABEL}</label>
</hypervisor>
```

Edit existing hypervisor

```
PUT onapp.com/settings/hypervisors/{ID}.xml
PUT onapp.com/settings/hypervisors/{ID}.json
```

You can edit the following parameters:
ip-address – the Hypervisor ID
label – the Hypervisor label

XML Output Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<hypervisor>
  <ip_address>{IP}</ip_address>
  <label>{LABEL}</label>
</hypervisor>
```

Delete hypervisor

```
DELETE onapp.com/settings/hypervisors/{ID}.xml
```

```
DELETE onapp.com/settings/hypervisors/{ID}.json
```

Disks

Get the list of all available disks

```
GET onapp.com/settings/disks.xml
GET onapp.com/settings/disks.json
```

XML Output Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<disks type="array">
  <disk>
    <created_at type="datetime">2010-06-21T10:36:54Z</created_at>
    <add_to_linux_fstab type="boolean" nil="true"></add_to_linux_fstab>
    <disk-size type="integer">20</disk_size>
    <updated_at type="datetime">2010-08-03T14:40:03Z</updated_at>
    <primary type="boolean">true</primary>
    <required_automatic_backup type="boolean">true</required_automatic_backup>
    <data_store_id type="integer">2</data_store_id>
    <id type="integer">198</id>
    <disk_vm_number type="integer">1</disk_vm_number>
    <is-swap type="boolean">false</is_swap>
    <mount-point nil="true"></mount_point>
    <identifier>hflgp0gcjhfwdt</identifier>
    <virtual_machine_id type="integer">70</virtual_machine_id>
    <built type="boolean">true</built>
    <locked type="boolean">false</locked>
  </disk>
</disks>
```

Where:
created_at - the date when the disk was created in the [YYYY][MM][DD]T[hh][mm]Z format
disk-size – the size of a disk
updated_at – the date when the disk was updated in the [YYYY][MM][DD]T[hh][mm]Z format
primary – true if the disk is primary. Otherwise false.
required-automatic-backup – true if automatic backup is scheduled. Otherwise false.
data-store-id – the ID of the data store this disk is located
id – the disk ID
disk-vm-number – the number of virtual machines using this disk
is-swap – true if this is a swap disk. Otherwise false.
virtual-machine-id – the ID of the virtual machine using this disk
built – true if the disk is built. Otherwise false.
Locked – true if the disk is locked. Otherwise false.

Get the list of disks available for a particular VM

```
GET onapp.com/virtual_machines/{VM_ID}/disks.xml
GET onapp.com/virtual_machines/{VM_ID}/disks.json
```

XML Output Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<disks type="array">
  <disk>
    <created_at type="datetime">2010-07-12T09:31:42Z</created_at>
    <add_to_linux_fstab type="boolean" nil="true"></add_to_linux_fstab>
    <disk-size type="integer">10</disk_size>
    <updated_at type="datetime">2010-08-03T14:40:03Z</updated_at>
    <primary type="boolean">true</primary>
    <required_automatic_backup type="boolean">true</required_automatic_backup>
    <data_store_id type="integer">2</data_store_id>
    <id type="integer">437</id>
    <disk_vm_number type="integer">1</disk_vm_number>
    <is-swap type="boolean">false</is_swap>
    <mount-point nil="true"></mount_point>
    <identifier>0gdnd4bcxzwsrk</identifier>
    <virtual_machine_id type="integer">206</virtual_machine_id>
    <built type="boolean">true</built>
    <locked type="boolean">false</locked>
  </disk>
</disks>
```

Edit a disk

```
PUT onapp.com/settings/disks/{ID}.xml
PUT onapp.com/settings/disks/{ID}.json
```

Currently you can edit the *Size* parameter.

Add new disk

```
POST onapp.com/virtual_machines/{VM_ID}/disks.xml
POST onapp.com/virtual_machines/{VM_ID}/disks.json
```

To add a new disk, send the following required parameters:
Data Store – the data store ID
Size – the disk size

XML Output Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<disk>
  <disk-size type="integer">0</disk_size>
  <primary type="boolean">false</primary>
  <required_automatic_backup type="boolean">false</required_automatic_backup>
  <data_store_id type="integer" nil="true"></data_store_id>
  <disk_vm_number type="integer" nil="true"></disk_vm_number>
  <is-swap type="boolean">false</is_swap>
  <virtual_machine_id type="integer" nil="true"></virtual_machine_id>
  <built type="boolean">false</built>
  <locked type="boolean">false</locked>
</disk>
```

Delete a disk

```
DELETE onapp.com/settings/disks/{ID}.xml
DELETE onapp.com/settings/disks/{ID}.json
```

Data Storages

The DataStore class represents the Data Storages of the OnAPP installation.

Get the list of data storages

```
GET onapp.com/settings/data_stores.xml
GET onapp.com/settings/data_stores.json
```

XML Output Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<data-stores type="array">
  <data_store>
    <label>SAN1</label>
    <created_at type="datetime">2010-04-27T15:55:08Z</created_at>
    <updated_at type="datetime">2010-08-04T09:02:15Z</updated_at>
    <id type="integer">1</id>
    <local_hypervisor_id type="integer" nil="true"></local_hypervisor_id>
    <data_store_size type="integer">890</data_store_size>
    <identifier>radar-san1</identifier>
  </data_store>
</data_stores>
```

Where:
created_at - the date in the [YYYY][MM][DD]T[hh][mm]Z format
data_store_size - the size of your data store in human readable format (e.g., 1K 234M 2G)
id - the data store ID
label – the data store label
local_hypervisor_id - the ID of the Hypervisors using this Data Store
updated_at - the date when the Data Store was updated in the [YYYY][MM][DD]T[hh][mm]Z format

Get a particular data storage details

```
GET onapp.com/settings/data_stores/{ID}.xml
GET onapp.com/settings/data_stores/{ID}.json
```

XML Output Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<data_store>
  <label>SAN1</label>
  <created_at type="datetime">2010-04-27T15:55:08Z</created_at>
  <updated_at type="datetime">2010-08-04T09:02:15Z</updated_at>
  <id type="integer">1</id>
  <local_hypervisor_id type="integer" nil="true"></local_hypervisor_id>
  <data_store_size type="integer">890</data_store_size>
  <identifier>radar-san1</identifier>
</data_store>
```

Add new data storage

```
POST onapp.com/settings/data_stores.xml
POST onapp.com/settings/data_stores.json
```

XML Output Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<data_store>
   <data_store_size>{SIZE}</data_store_size>
   <label>{LABEL}</label>
</data_store>
```

Edit existing data storage

```
PUT onapp.com/settings/data_stores/{ID}.xml
PUT onapp.com/settings/data_stores/{ID}.json
```

You can edit the data store disk capacity and label.

XML Output Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<data_store>
    <data_store_size>{SIZE}</data_store_size>
    <label>{LABEL}</label>
</data_store>
```

Delete data storage

```
DELETE onapp.com/settings/data_stores/{ID}.xml
DELETE onapp.com/settings/data_stores/{ID}.json
```

Backups

Get the list of backups available for a particular VM

```
GET onapp.com/virtual_machines/{VM_ID}/backups.xml
GET onapp.com/virtual_machines/{VM_ID}/backups.json
```

An array of backups is returned. If there are no backups, an empty array is returned.

Add new backup

```
POST onapp.com/virtual_machines/{VM_ID}/backups.xml
```

```
POST onapp.com/virtual_machines/{VM_ID}/backups.json
```

Delete a backup

```
DELETE onapp.com/virtual_machines/{VM_ID}/backups.xml
DELETE onapp.com/virtual_machines/{VM_ID}/backups.json
```

Templates

Get the list of available templates

```
GET onapp.com/templates.xml
```
An array of templates is returned. If there are no templates, an empty array is returned.

Make a template public

```
POST onapp.com/templates/{ID}/make_public.xml
POST onapp.com/templates/{ID}/make_public.json
```

If a template is queued to be moved to a public list successfully, an HTTP 201 response is returned.

Download a new template to be used with OnApp

```
GET onapp.com/templates.xml?remote_template_id={REMOTE_TEMPLATE_ID}
GET onapp.com/templates.json?remote_template_id={REMOTE_TEMPLATE_ID}
```

Statistics

Get daily stats (information on the resources used by virtual machines):
```
GET onapp.com/usage_statistics.xml
GET onapp.com/usage_statistics.json
```

Only GET method is available for the statistics.