ComputeNext REST API Documentation

Release 2

ComputeNext

June 04, 2015

Contents

Contents:

CHAPTER 1

Introduction

The ComputeNext Web Services REST API is an Application Programming Interface (API) designed to search for, provision, manage and control cloud computing resources from the ComputeNext Marketplace.

1.1 Concepts

Customers can search and browse Cloud Resources available in the marketplace from multiple cloud providers.

The resources can be provisioned as Instances. Applications are typically deployed as dependent services.

A Workload is a grouping or collection of multiple cloud resources that captures the description of the services a customer needs. It includes the details and configuration of the services, their dependencies and desired service-level expectations. The workload is also the entity that facilitates management and control of provisioned instances.

1.2 API endpoint

The ComputeNext API is available on the base URL:

```
<https://cws.computenext.com/api/>
```

The section Resources gives examples of making resource queries based on this base URL above.

1.3 API Versions

The latest API version is 2.0. It is accessible via the base URL shown above though user authentication is required.

1.4 Authentication

The API supports HTTP Basic Authentication method using API Key and Secret Key. See *Obtaining API Keys* for information on creating API Keys for your ComputeNext account.

HTTP Basic Authentication is done with the *Authorization* header. The value of the header is base64 encoded value with a colon-separated, concatenated string of API Key and Secret Key. For example the API key 11111111-1111-1111-1111-111111111111 and Secret key of 22222222-2222-2222-2222-222222222222, the base64 encoding generates the following header entry:

```
Authorization: Basic MTExMTExMTEtMTExMS0xMTExLTExMTEtMTExMTExMTExMTExOjIyMjIyMjIyLTIyMjItMjIyMi0yMjIy
```

1.5 Obtaining API Keys

The API Keys are currently generated by the ComputeNext Support Team. To get a key please send an email requesting an API Key to support@computenext.com

CHAPTER 2

Resources

2.1 Introduction

For information on how to access the API and API authorization see: *Obtaining API Keys*.

The resource service has a GET API that is used to query data in the store.

The GET command will retrieve the resource definitions.

With the introduction of the *runcws* nodejs module (see *Getting Started with Workloads*) it is possible to use the runcws tool to query resources and get much of the same information as the GET API methods provide. This interface will be preferred when working with workloads.

Resource service is based on ontological concepts.

The ontology of a resource service can be described by its data schema. The schema defines which attributes are returned during a query search and which attributes can be added to a resource during CRUD operations. The ontology is described in various .ttl files. The documentation below provides some examples of API calls and return values. For a full list of all possible return elements see *Resource Schema*.

2.2 Resource API Methods

2.2.1 Query Resources

This method is used to search and discover available computing resources based on a wide variety of filtering options.

In general, query resource requests have the following formats:

HTTP GET Request:

GET /resourceQuery/query/{*resourceType*}

or in a browser window:

```
<https://cws.computenext.com/api/resourceQuery/query/{*resourceType*}>
```

or using runcws at the command-line in a terminal window:

>node runcws.js query {*resourceType*}

The current *resourceType* identifiers for use in the above tools/interfaces are:

- virtualMachine

- volumeStorage
- keyPair
- securityGroup
- image
- instanceType
- softwareType
- loadBalancer - for load distribution across running instances using selectable algorithms
- package - software packages provisionable by Chef on various operating system platforms

Note: See the list of resourceTypes for instances in section *Instances*.

Example: To query all image resources, one can use either of the following methods:

Using an HTTPS request in a browser:

https://cws.computenext.com/api/resourceQuery/query/virtualMachine

or using the *runcws* command (*Getting Started with Workloads*) we get the following output for querying the images:

```
>node runcws.js query virtualMachine
```

Response:


```
[
        {
                "zone": "singapore",
                "tier": "4",
                "slaSummary": "http://www.cloudiro.com/tos.html",
                "platform": "OnApp",
                "numberOfNines": "4",
                "location": "Singapore, Singapore",
                "connectorType": "onapp.compute",
                "uri": "vx/cloudiro/singapore/9bc2d15622d3c26242a32c436a665e78",
                "Benefits": "<p>We love technology and helping our customers. Cloudiro is a team of c
                "description": "Cloudiro Small VM with OpenSUSE 12.1 64 Bit",
                "DetailedDescription": "<p>openSUSE is a free and Linux-based operating system for yo
                "name": "Cloudiro Small VM with OpenSUSE 12.1",
                "providerResourceId": "9bc2d15622d3c26242a32c436a665e78",
                "ShortDescription": "<p><strong>Minimum System Requirements:</strong></p><p><strong>C
                "instanceTypeName": "Small VM",
                "instanceTypeUri": "vm/cloudiro/singapore/small",
                "imageName": "OpenSUSE 12.1",
                "imageUri": "image/cloudiro/singapore/201",
                "costPerUnit": "25 USD",
                "costUnit": "per month",
                "isAvailable": "1",
                "provider": "Cloudiro",
                "region": "Singapore",
                "id": "vx_cloudiro_singapore_9bc2d15622d3c26242a32c436a665e78",
                "providerId": "cloudiro",
                "rank": "vx_cloudiro_singapore_9bc2d15622d3c26242a32c436a665e78",
                "created": "2014-11-03T11:39:14.357Z",
                "updated": "2014-11-03T11:39:14.357Z",
                "atReadLimit": 300
        }
]
```

Returns:

A list of resource objects that match the query parameters.

Important: This method uses pagination and the example above is the last entry in the list.

> If "atReadLimit": 200 appears as it does above then there are more results. To view more results pass in the created parameter as show below to view the next page of entries.

Example:

https://cws.computenext.com/api/resourceQuery/query/virtualMachine?from=2014-11-03T11:39:14.357Z

2.2.2 Retrieve Region Details

This method retrieves the list of details on all the available provider regions.

Request:

GET /api/resourceQuery/region

Or using the *runcws* command (see *Getting Started with Workloads*): >node runcws.js region

Example:

https://cws.computenext.com/api/resourceQuery/region

Or using the runcws command (see *Getting Started with Workloads*):

```
node runcws.js region
```


```
[
    {
        "benefits": "<p>* CloudSigma platform powered IaaS<br />* High reliability and high performar
        "capabilities": "https://www.computenext.com/cloudfederation/instance#Resource-Capabilities-c
        "connectorType": "ComputeNext.CloudFederation.ProviderGateway.CloudSigma.CloudSigmaProvider",
        "description": "Zurich, Switzerland",
        "detailedDescription": "<p>CloudSigma is an innovative Infrastructure-as-a-Service (IaaS) pro
        "featured": "true",
        "id": "dababfcf-0769-4dae-a248-2b1b337fedee",
        "isAvailable": "1",
        "longDescription": "{\"Provider\": \"CloudSigma-Zurich\", \"Provider ID\": \"f4231561-7011-4e
        "name": "Zurich",
        "numberOfNines": "4",
        "platform": "KVM",
        "provider": "CloudSigma-Zurich",
        "providerInfoId": "f4231561-7011-4e1b-ad0b-dc492abbccd0",
        "slaSummary": "CloudSigma is a provider of managed hosting, colocation and managed services t
        "supportedActions": "http://www.computenext.com/cloudfederation/instance#Resource-Actions-dab
        "tier": "3",
        "zone": "Zurich",
        "type": "http://www.computenext.com/cloudfederation#Region"
    },
    {
        ......
    },
    {
        ......
    }
]
```

Returns:

The details of every provider region

2.2.3 Retrieve Region Properties

This method retrieves a lists of properties that are related to region information.

Request:

GET /api/resourceQuery/region/distinct/<params>

The current parameters that this function accepts are:

- provider
- CloudPlatformType
- location
- regionUri
- SlaSummary
- ConnectorClassname
- Capabilities

Example:

https://cws.computenext.com/api/resourceQuery/region/distinct/provider

Response:

```
[
        "cacloud",
        "cloudoye",
        "cloudiro",
        "cloudprovider",
        "computerline",
        "gmocloud",
        "internap",
        "datacate",
        "gandi"
]
```

2.2.4 Retrieve Image Restrictions

Returns the restrictions object for an image. The restrictions describe what properties must match in order to make a valid Image+Instance choice.

- Provider - The display name of the provider the instance must be offered by
- Region - The display name of the provider region the instance must be offered by
- Platform - The name of the cloud platform the instance must be offered by (i.e OpenStack, vCloud, etc.)
- cpuCount - The minimum number of cpus needed in an instance in order to support the image (The value will be in range format. i.e. [2,] means the instanceType must have at least 2 cpu cores)
- CPUSpeed - The minimum amount of cpuSpeed (in GHZ) needed in an instance in order to support the image (The value will be in range format)

- Local storage - The minimum amount of local storage (in GB) needed in an instance in order to support the image (The value will be in range format)
- RAM - The minimum amount of ram (in GB) needed in an instance in order to support the image (The value will be in range format)

Request:

GET /api/resourceQuery/restrictions/<imageURI>

Or using the *runcws* command (see *Getting Started with Workloads*):

```
>node runcws.js restrictions <imageURI>
```

Example:

https://cws.computenext.com/api/resourceQuery/restrictions/image/enocloud/montreal/a4b1ca48-b3ce-4961-b36c-49777b9115c0

Or use the node runcws command (see *Getting Started with Workloads*) to retrieve restrictions for the image:

```
>node runcws.js restrictions /image/cloudiro/singapore/201
```

Response


```
{
        "cpuCountMin": 1,
        "cpuSpeedMin": 1,
        "localStorageMin": 20,
        "ramMin": 1,
        "cpuCount": "[1,]",
        "cpuSpeed": "[1,]",
        "localStorage": "[20,]",
        "ram": "[1,]",
        "provider": "cloudiro",
        "region": "singapore",
        "platform": "OnApp"
}
```


2.2.5 Retrieve Resource Capabilities

Returns the capabilities object associated with the resources region. The list of capabilities will either be "true" or "false". resourceId can either be an instanceType, VM, VS, image, or a region.

- createImageSupported - True or false value determining whether or not the provider supports the creation of private images from existing virtual machines
- keypairRequired - True or false value determining whether or not the provider supports the creation of a key pair
- passwordRequired - True or false value determining whether or not the provider gives a password string for logging into a deployed virtual machine (used most commonly with Microsoft Windows images)
- securityGroupRequired - True or false value determining whether or not the provider supports the creation of a network security group
- userDataSupported - Whether user data supported for user in the provider.

Request:

GET /api/resourceQuery/capabilities/<resourceURI>

Or using the runcws command: *>node runcws.js capabilities <resourceURI>*

Example:

https://cws.computenext.com/api/resourceQuery/capabilities/vm/enocloud/montreal/9

Or using the runcws command (see *Getting Started with Workloads*):

```
>node runcws.js capabilities /vm/enocloud/montreal/9
```

Response:

```
{
  "createImageSupported": "false",
  "keypairRequired": "true",
  "passwordRequired": "true",
  "securityGroupRequired": "true",
  "userDataSupported": "true"
}
```

2.2.6 Resource Details

Returns the details of a resource based on its URI.

Request Body: GET /api/resourceUri/{uri}

Or using the runcws command (see *Getting Started with Workloads*):

```
>node runcws resource <resourceURI>
```

Returns:

A list of details of the given resource along with information about the provider.

Example:

https://cws.computenext.com/api/resourceUri/vm/enocloud/montreal/9

Or using the runcws command (see *Getting Started with Workloads*):

```
>node runcws.js resource /vm/enocloud/montreal/9
```

2.2.7 Retrieve Resource Actions

An array of objects containing the given input, as well as a list of actions for each resource

- All actions - All the possible actions that can be performed on this resource
- Available actions - A list of valid actions that can be performed on the resource given its current state (a subset of AllActions)

Request:

POST /api/resourceQuery/action

Request Body

```
[
    {
        "uri":"vm/enocloud/montreal/9",
        "State":"running"
    }
]
```

Returns:

Given a list of resourceIds and their states, returns a list of AvailableActions and AllActions for each resourceId. See below subpages for more specific details.

Example:

https://cws.computenext.com/api/resourceQuery/action

Or using the *runcws* command (see *Getting Started with Workloads*), a JSON file provided with parameters for the resource being queried:

```
>node runcws.js action <JSON file>
```

Response:


```
[
     {
         "uri": "vm/enocloud/montreal/9",
         "State": "running",
         "AllActions":
          [
              {
                  "id": "create",
                  "name": "Create"
              },
              {
                  "id": "delete",
                  "name": "Delete"
              },
              {
                  "id": "stop",
                  "name": "Stop"
              },
              {
                  "id": "reboot",
                  "name": "Reboot"
              },
              {
                  "id": "createImage",
                  "name": "CreateImage"
              },
              {
                 "id": "start",
                 "name": "Start"
             },
             {
                  "id": "delete-ip",
                  "name": "DeleteIp"
             },
             {
                  "id": "create-ip",
                  "name": "CreateIp"
             }
      ],
     "AvailableActions":
        [
             {
                  "id": "delete",
                  "description": "Delete your instance",
```

```
                "name": "Delete"
            }
        ]
    }
]
```

2.2.8 Validate Resource

It is used to validate the options chosen for a configurable instance and whether it meets the requirements of a given image. It can also be used to validate the chosen options for a configurable volume store or simply the existance of an image/instance resource. The API is automatically called by the workload API when adding a VM or VS to a workload. The configurable resource options to choose for user requirements are:

- cpuCount - number of CPUs
- RAM - amount of GB of RAM
- localStorage - amount of GB of local Storage

Request:

POST /api/resourceQuery/validate

Using the *runcws* command (see *Getting Started with Workloads*) with a supplied JSON file containing parameters for the resource being validated, enter a command on the command-line:

```
>node runcws.js validate <JSON file>
```

The body of the request is returned as confirmation of what it received as shown in the following example to validate an image/instance from one provider. Because it is a configurable resource, values of ram, cpuCount, and local storage are being verified to be within range limits and if valid, a price is returned in the response.

```
validate (validate resource)
options: {
  "url": "http://cws.computenext.com/api/resourceQuery/validate",
  "method": "post",
  "json": {
    "instanceTypeUri": "vm/hegerys/computenext/configurable",
    "imageUri": "image/hegerys/computenext/vapptemplate-59726d51-b021-44ca-918d-33659452b1b1",
    "ram": "2",
    "cpuCount": "2",
    "localStorage": "16"
  },
  "auth": {
   xxx
   xxx
  }
}
```

Returns:

Values for the chosen resource given in the request json file. For image+instance configuration, it validates instance config choices with image restrictions. For configurable resources with ranges to choose from, the selection is validated and if successful, the *price* for that selected configuration is also returned.

Example:

https://cws.computenext.com/api/resourceQuery/validate or using runcws: >node runcws validate <JSON file>

Response:

```
{
  "ram": 2,
  "cpuCount": 2,
  "localStorage": 16,
  "cpuCountLabel": 2,
  "CurrencyCode": "USD",
  "localStorageLabel": 16,
  "ramLabel": 2,
  "totalUnitPrice": "0.095333",
  "ChargeAmountUnit": "per hour"
}
```


2.2.9 Resource Errors

Status Codes:

- 400: These errors will return a descriptive message detailing why the API call failed. It is usually caused by invalid parameters given.
- 500: Please report this error to ComputeNext support (support@computenext.com) so that we can track the cause of the problem.
- 404: The resource being searched for was not found in the system.
- 403: User does not have the correct role to access for a particular resource.

CHAPTER 3

Resource Schema

This is a list of all the possible return elements from the various resource API calls which can be made. The resource field in each one below indicates on what resources these elements are present.

Tier

- description: Ranking levels from 1 - 4 according to definitions in ANSI Standard TIA-942 Telecommunications Infrastructure Standards for Data Centers.
- resource: VM, VS, Image, Instance.

softwareType

- description: The software that is on a VM.
- resource: VM.

slaSummary

- **description: A summary of the agreements made between customer and service providers generally regarding what kind**
- resource: VM, VS, Image, Instance.

ram

- description: The amount of RAM a VM or Instance has.
- resource: VM, Instance.

platform

- description: The URI of the image.
- resource: VM, VS, Image, Instance.

operatingSystemType

- description: The operating system that is installed on the resource.
- resource: VM, Image.

operatingSystemVersion

- description: The bit version of the OS: 32 or 64
- resource: VM, Image, Instance.

numberOfNines

- description: An availability rating for percentage of uptime for the resource. It is related to the tier level ranking and certification is provided by various consulting firms such as Uptime institute.

- resource: VM, VS, Image, Instance.

Location

- description: The geographical location of the resource.
- resource: VM, VS, Instance.

localStorage

- description: The amount of RAM storage local to the virtualMachine processor
- resource: VM, instance

cpuSpeed

- description: The speed of the processor (GHz)
- resource: VM, Instance.

cpuCount

- description: The URI of the image
- resource: VM, Instance

connectorType

- description: The connector that the resource uses to communicate with the provider
- resource: VM, image, instance

Benefits

- description: The value from using the resource from the provider
- resource: VM, VS, image, instance, softwareType, package

costPerUnit

- description: The URI of the image
- resource: VM, VS, Instance, loadBalancer, package

costUnit

- description: The URI of the image
- resource: VM, VS, Instance, loadBalancer

description

- description: A brief description of the resource
- resource: VM, VS, Image, Instance, loadBalancer, softwareType, package

DetailedDescription

- description: The full body description of the resource.
- resource: VM, VS, Image, Instance, Software, loadBalancer, softwareType, package

imageId

- description: The URI of the image
- resource: VM,

imageURI

- description: The URI of the image

- resource: VM

instanceTypeId

- description: The URI of the image
- resource: VM

instanceTypeUri

- description: The URI of the image
- resource: VM

isAvailable

- description: A true/false flag indicating if the resource is available.
- resource: VM, VS, image, instance, loadBalancer

name

- description: The name of the resource
- resource: VM, VS, instance, loadBalancer, keyPair, securityGroup, softwareType, package

provider

- description: The URI of the image
- resource: VM, VS, image, instance, loadBalancer, keyPair, securityGroup

providerId

- description: The id of the provider
- resource: VM, VS, image, instance, loadBalancer, keyPair, securityGroup

rank

- description: a 32 digit hexadecimal number associated with the resource
- resource: VM, VS, image, instance, loadBalancer, package

region

- **description: A geographical region is an area within a location where the resource provider offers an instance of the resou**

- resource: VM, VS, image, instance, loadBalancer, keyPair, securityGroup

ShortDescription

- description: A "short" description of the resource
- resource: VM, VS, image, instance, softwareType, package

usageInstructions

- description: Instructions on how to use the resource
- resource: image, VM,

zone

- description: See **region** above. Availability zones exist within geographical regions for a location of a resource. They are selected for separating instances and thus ensuring a higher level of fault isolation, availability and service.
- resource: VM, VS, image, instance, loadBalancer

providerResourceId

- description: The URI of the image
- resource: image, instance, loadBalancer, VS, keyPair, securityGroup

specialOffer

- description: A flag to indicate if it is a special offer.
- resource: VS, image, instance

URI

- description: The URI of the resource. See *Instances* for ComputeNext's use and definition for working with its resources.
- **resource: image, instance, VS, loadBalancer, keyPair, securityGroup, package**

size

- description: The amount or quantity of the resource. It can be the fixed amount or over a configurable range in the case of volume storage.
- resource: VS

software

- description: Software applications or services which can be installed and configured to run on instsances of VMs. It is not to be confused with *softwareType* above. software is available in the marketplace at the same level as compute, storage and network (loadBalancers).
- resource: image

id

- description: The unique identifier for the resource.
- resource: instance, image, VM, loadBalancer, softwareType, package

operatingSystem

- description: The operating system that is on the image
- resource: Image.

category

- description: The type of software services the software package pertains to.
- resource: softwareType, package

algorithm

- description: The type of load sharing algorithm selectable for a loadBalancer.
- resource: loadBalancer

noOfNodeMandatory

- description: The minimum number of nodes required by the loadBalancer.
- resource: loadBalancer

sharedLoadBalancer

- description: A flag to indicate whether the loadBalancer is shared
- resource: loadBalancer

protocol

- description: communication protocol options for the loadBalancer: {Http port}, {SSL port}, {connect port}.
- resource: loadBalancer

logoName

- description: The name of the brand logo for a product or company
- resource: softwareType

CHAPTER 4

Instances

4.1 Introduction

For information on how to access the API and API authorization see: *Obtaining API Keys*.

The instance API is a low level interface designed to allocate, monitor, control and de-allocate instances of cloud resources.

All resources have a description as defined in Cloud Resources. The instance object is an instantiation of one specific type of resource.

In the instance API, everything is considered an "instance" (not just a VM instance). We can have an instance of a key pair, an instance of a security group, an instance of a volume store, etc.

All **resources** are identified by means of a URI (Universal Resource Identifier) which uniquely identifies the resource in the ComputeNext catalog.

This URI has the form <resourceType>/<provider>/<region>/<providerResourceId>. For example: vm/hpcloud/nova/small

NOTE: By convention the URI is simply a string the ComputeNext catalog uses to uniquely identify a specific type of resource from a provider at a particular region.

The URI is always lower case to avoid case sensitivity issues.

All **instances** are identified by an **instanceId** which is a Guid.

The instance API is asynchronous. Most requests return a **requestId** which is a Guid. They will return the requestId when the request has been accepted and is in-progress, but has not completed. The user must poll with that requestId to find out the status of the request. Eventually the request will become completed or will have failed.

Possible resource types for instantiable instances (see list of resourceType in section Resources of Cloud Resources) are -

- kp - key pair
- sg - security group
- vm - virtual machine
- vs - volume storage
- image - image
- snap - volume snapshot
- ip - floating (elastic) IP address

- lb - loadBalancer for distributing compute load across various running instances using selectable algorithms

There are basically two types of records that are persisted by the instance API in the database - request and instance records.

requests. These are records which keep track of the user requests the API. Each request has a **requestId** (a Guid) and a **requestStatus**, plus all the parameters and results etc.

requestStatus can be -

- in-progress
- completed
- failed

If failed there will be some error information indicating the cause of the failure. Error information will include an error code, an error message, and an error ticket (a Guid) which we can use to track down the exact source of the error from the trace files.

instances: These are the records which keep track of the instances allocated at the provider side. Each instance has an **instanceId** and an **instanceStatus**. The instanceStatus is the last status we were able to retrieve from the provider side. Values for instanceStatus are similar to (creating, created, stating, deleted. etc.) and are all lower case.

metadata. The instance API uses a URI or Guids to identify everything. Names are not used. However, every instance can have metadata (tags) added. This can be used to add names, descriptions, etc. and it is also used by the workload API to tag specific instances as being part of a particular workload or transaction by adding a workloadId or a transactionId to the instance metadata.

(**Note**: if you want to add new metadata attributes to a VM instance them you must read the existing metadata, add your new attributes to it, and then update it.)

All request parameters and responses in the instance API are in JSON format.

4.2 Actions

In the instance API, the actions that can be performed vary by the resource type.

Actions are identified in the following format: <resourceType>.<CRUD action>.<action modifier>

- <resourceType> identifies the resource type (such as “vm”, “kp”, “sg” etc. as listed above)
- <CRUD action> is one of the create, retrieve, update or delete (CRUD) actions
- <action modifier> is an additional modifier for the CRUD action (such as “start”, “stop”, “reboot” etc.)

Actions names are lower case to avoid case sentivity issues. Some examples are “vm.create”, “vs.update.attach”, “kp.delete”.

The <resourceType> is determined from the resource type of the resource or instance being acted upon - you do not have to specify this explicitly.

The CRUD action is determined from the type of REST action being performed -

- POST = create
- GET = retrieve
- PUT = update
- DELETE = delete

The <action modifier> is not required in many cases, but if it is required, it is specified as the *action* query parameter on the REST request URL *"?action=start"* for example.

A full list of the resource types, the actions they support, and the parameters for those actions can be found in Resource Types, Actions and Parameters. *Getting Started with Instances* gives samples of actions on instances using runcws commands with json file parameters with actions contained in them.

4.3 Instance Methods

Note that Basic authentication is required but is not shown here. See Authentication for details.

4.3.1 Create instance from resource

Create one instance from a resource uri and parameters. The parameters required will vary depending on the resource.

POST /api/resource/<uri>

Request Body

- metadata = some JSON to be added to the instance as metadata (tags)
- <parameters> = will vary depending on the resource type.

Example

POST /api/resource/kp/hpcloud/nova/standard (no parameters)

Body

```
{
    "metadata":
    {
          "name": "HP_KP",
          "description": "Keypair Response time"
    }
}
```

This will create a key pair in the HPCloud Nova region.

Returns

A request JSON object.

4.3.2 Retrieve request

Retrieve one request.

GET /api/request/<requestId>

Example

GET /api/request/aec5fb2d-9990-4cdf-bc24-6fbc1a13cbf4

Response

```
[
      {
            "requestId": "3780520d-9463-4273-8d52-aa492b185168",
            "instanceId": "b2b7c744-fe31-4db4-aab2-d56c118ae9ee",
```

```
            "created": "2013-07-03T07:45:56.001Z",
            "updated": "2013-07-03T07:46:00.674Z",
            "ownerId": "ae16300f-6eba-426f-b92b-dd5626f094bf",
            "requestStatus": "completed",
            "resourceUri": "kp/hpcloud/nova/standard",
            "resourceType": "kp",
            "provider": "hpcloud",
            "region": "nova",
            "connector": "openStack.compute",
            "parameters":
            {
                "action": "kp.create",
                "instanceId": "b2b7c744-fe31-4db4-aab2-d56c118ae9ee",
                "kp_providerResourceId": "standard",
                 "zone": "nova"
            },
            "metadata":
            {
                "name": "KP2",
                "description": "my first key pair"
            },
            "results":
            {
                "providerInstanceId": "b2b7c744-fe31-4db4-aab2-d56c118ae9ee",
                "keyFingerprint": "6a:c4:4f:ba:02:0b:d4:73:93:b7:08:23:d7:d2:17:40",
                "privateKey": "<HIDDEN>",
                "publicKey": "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAAAgQDBvzE7rWC9G7+bpXwwluG08im... \n",
                "instanceStatus": "created"
            }
        }
]
```


Returns

A request JSON object.

4.3.3 Retrieve multiple requests

Retrieve multiple requests. Query parameters can be provided to return just those requests that match.

GET /api/request

Query Parameters

- requestStatus = in-progress | completed | failed
- resourceType = kp | sg | vm | vs | image | snap | ip | lb
- provider = <provider name, lower case>
- region = <region name, lower case>
- cleanup = true - if this is set, then any old requests for this user that have "requestStatus = completed" or "requestStatus = failed" will be cleaned up.

If multiple query parameters are provided then you will get the logical AND effect from these parameters.

Example

GET /api/request?resourceType=vm

Returns

A JSON array of request objects.

4.3.4 Create an instance from an instance

Create one instance from another instance. At the moment this is only used for creating a new vm image instance from an existing vm instance.

POST /api/instance/<instanceId>

Example

Body

```
{
    "virtualMachineId": "456afec0-2c63-45d0-8436-df1815aabb5c",
    "metadata":
    {
        "name": "New IMAGE",
        "description": "my first image"
    }
}
```

Query Parameters

TODO:

Returns

A request JSON object.

Note: Section *Getting Started with Instances* has an example of creating an image from a vm instance using runcws in sub-section *Create a Private image from a Virtual Machine*.

4.3.5 Retrieve instance

Retrieve one instance.

GET /api/instance/<instanceId>

Query Parameters

- refresh = true - refresh the instance from the provider side.

If this is *not* set, then the instance is fetched from the database only, and no request is made to the provider side. An instance is returned.

If this *is* set, then a request is returned and a request is sent to the provider side to get the latest instance status.

Returns

If refresh is specified, a request JSON object is returned.

If refresh is not specified, an instance JSON object is returned.

Note: Section *Getting Started with Instances* has several examples of getting an instance using runcws. See sub-section *Create a Private image from a Virtual Machine*.

4.3.6 Retrieve multiple instances

Retrieve multiple instances. Query parameters can be provided to return just those instances that match.

GET /api/instance

Query Parameters

- instanceStatus = creating | created | starting | started... etc.
- resourceType = kp | sg | vm | vs | image | snap | ip | lb
- provider = <provider name, lower case>
- region = <region name, lower case>
- metadata.<parameter> = query by some metadata parameter
- workloadId = workload id (Guid)
- transactionId = transaction id (Guid)

Example

GET /api/instance?workloadId=ad5a66c3-8895-4d71-b786-1d129b33326e

Response


```
[
    {
        "instanceId": "de3ef325-7daa-4d47-aee8-4d0391b18d9b",
        "created": "2013-10-05T12:46:12.428Z",
        "updated": "2013-10-05T12:46:39.252Z",
        "ownerId": "0cfc0576-0088-486a-8416-7dbd79f2776e",
        "resourceUri": "vm/hpcloud/nova/standard.small",
        "resourceType": "vm",
        "provider": "hpcloud",
        "region": "nova",
        "providerResourceId": "Standard.small",
        "attributes":
        {
            "providerInstanceId": 2205937,
            "password": "KAcqKeQX9XNagL5a",
            "instanceStatus": "running",
            "privateIpAddress": "10.2.225.213",
            "publicIpAddress": "15.185.233.238"
        },
        "attributeTimestamps":
        {
            "password": "2013-10-05T12:46:39.244Z",
            "instanceStatus": "2013-10-05T12:46:39.245Z",
            "privateIpAddress": "2013-10-05T12:46:39.244Z",
            "publicIpAddress": "2013-10-05T12:46:39.244Z"
        },
        "metadata":
        {
            "name": "Ubuntu",
            "description": "Virtual machine",
            "workloadId": "333617b7-5ade-43b4-88eb-45703ec1b0d4",
            "transactionId": "ab827e43-be23-4a33-8e01-11925cbe64d4"
        },
        "parameters":
        {
```

```
            "imageUri": "image/hpcloud/nova/ami-00000075",
            "keyPairId": "15086ab9-bbe6-45d5-8d68-377bd4af58c4",
            "securityGroupIds":
            [
                "639a42d8-9d76-4422-919b-bcc49db0524c"
            ],
            "vm_providerResourceId": "Standard.small",
            "zone": "nova",
            "username": "ubuntu",
            "image_providerResourceId": "ami-00000075",
            "keyPairId_providerInstanceId": "15086ab9-bbe6-45d5-8d68-377bd4af58c4",
            "securityGroupIds_providerInstanceId":[398345]
        }
    },
    {
        "instanceId": "888bd60a-e8e0-46d2-9fb9-60d0596d837a",
        "created": "2013-10-05T12:47:09.376Z",
        "updated": "2013-10-05T12:47:16.946Z",
        "ownerId": "0cfc0576-0088-486a-8416-7dbd79f2776e",
        "resourceUri": "vs/hpcloud/nova/standard.10",
        "resourceType": "vs",
        "provider": "hpcloud",
        "region": "nova",
        "providerResourceId": "standard",
        "attributes":
        {
            "providerInstanceId": 675435,
            "instanceStatus": "attached",
            "virtualMachineId": "de3ef325-7daa-4d47-aee8-4d0391b18d9b",
            "virtualMachineId_providerInstanceId": 2205937
        },
        "attributeTimestamps":
        {
            "instanceStatus": "2013-10-05T12:47:16.936Z",
            "virtualMachineId": "2013-10-05T12:47:16.936Z",
            "virtualMachineId_providerInstanceId": "2013-10-05T12:47:16.937Z"
        },
        "metadata":
        {
            "description": "My first volume storage",
            "name": "vs1",
            "workloadId": "333617b7-5ade-43b4-88eb-45703ec1b0d4",
            "transactionId": "ab827e43-be23-4a33-8e01-11925cbe64d4"
        },
        "parameters":
        {
            "sizeInGBytes": 50,
            "attachTarget": "Ubuntu",
            "deviceName": "/dev/vdc",
            "virtualMachineId": "de3ef325-7daa-4d47-aee8-4d0391b18d9b",
            "vs_providerResourceId": "standard",
            "zone": "nova",
            "virtualMachineId_providerInstanceId": 2205937
        }
    }
]
```


Returns

A JSON array of instance objects.

Note: Section *Getting Started with Instances* has an example of listing multiple instances using runcws in the subsection *Getting Started With Instances*.

4.3.7 Update instance

PUT /api/instance/<instanceId>

Request Body

- metadata = some JSON to be added to the instance as metadata (tags)
- <parameters> = will vary depending on the resource type and the action.

Query Parameters

action = <some action>. Valid actions depend on the resource type.

If no action parameter is provided then default actions are as follows -

- POST = create
- PUT = update
- GET = retrieve
- DELETE = delete

Example

PUT /api/instance/ad5a66c3-8895-4d71-b786-1d129b33326e?action=stop

Response


```
[
    {
        "requestId": "27aac883-5444-4db0-acf6-795f3a9aec00",
        "instanceId": "456afec0-2c63-45d0-8436-df1815aabb5c",
        "created": "2013-10-07T09:30:13.854Z",
        "updated": "2013-10-07T09:30:14.588Z",
        "ownerId": "0cfc0576-0088-486a-8416-7dbd79f2776e",
        "requestStatus": "completed",
        "resourceUri": "vm/lunacloud/eu-west/xsmall",
        "resourceType": "vm",
        "provider": "lunacloud",
        "region": "eu-west",
        "connector": "lunaCloud.compute",
        "parameters":
        {
            "uri": "vm/lunacloud/eu-west/xsmall",
            "imageUri": "image/lunacloud/eu-west/ubuntu-12.04-x86_64",
            "providerInstanceId": "456afec0-2c63-45d0-8436-df1815aabb5c",
            "action": "vm.update.stop",
            "instanceId": "456afec0-2c63-45d0-8436-df1815aabb5c",
            "vm_providerResourceId": "{\"cpuCount\":\"1\",\"cpupower\":\"1500\" ...}
            "zone": "EU-West"
        },
        "metadata":
        {
            "name": "VM10",
            "description": "my first virtual machine"
```

```
        },
        "results":
        {
            "providerInstanceId": "456afec0-2c63-45d0-8436-df1815aabb5c",
            "instanceStatus": "stopping"
        }
    }
]
```

Returns

A request JSON object.

4.3.8 Update instance metadata

Updates the metadata field of an instance. Users will pass in a piece of JSON to update the instance's metadata with.

PUT /api/instance/<instanceId>

Request Body

- metadata = a JSON file to be added to the instance as metadata (tags)

Metadata Parameters

Common parameters of a instance in an active workload:

- name = The unique name that the system uses to identify a VM, should not be modified
- description = This attribute is used for VM name, as it appears in your workload. Can be updated by user.
- monitoringEnabled = Tells you if the ComputeNext monitoring service has been installed and enabled on the VM
- workloadId = The Id of an existing workload that the VM is attached to.
- transactionId = Do not modify this, include it in any json code you are tyring to pass through.

Users can define their one attribute in the form of "newattribute" : "testnewattibute" but must pass all previous attributes to retain them.

Example

PUT /api/instance/ad5a66c3-8895-4d71-b786-1d129b33326e

Content

```
{
   "metadata": {
                    "name": "CNWE734D_3504_E6F8_FA5A_41AA96E36E08",
                    "description": "My new name",
                    "monitoringEnabled": false,
                    "workloadId": "b6d5ed5a-29e7-4455-a95c-296072c2be1d",
                    "transactionId": "d40fe39c-855f-49a1-8341-5c0952b179c9",
                    "newattribute" : "testnewattibute"
            }
    }
```

Returns

A request JSON object.

4.3.9 Delete instance

Delete one instance. Note that "deleting an instance" means deleting (de-provisioning) the instance from the provider side. The instance record is not actually deleted from our database. The instance will transition through a "deleting" state and hopefully into a "deleted" state if all goes well.

DELETE /api/instance/<instanceId>

Example

DELETE /api/instance/ad5a66c3-8895-4d71-b786-1d129b33326e

Returns

A request JSON object.

CHAPTER 5

Resource Types, Actions and Parameters

This section gives descriptions of the instance schema for various resource instances. It describes methods, actions, and method parameters. Please refer to *Instances* for details on instances of various resources and examples using HTTP requests. Refer to *Getting Started with Instances* for examples of actions working with instances using runcws.

5.1 Resource Types

- Virtual Machine (vm)
- Key Pair (kp)
- Security Group (sg)
- Volume Storage (vs)
- Volume Snapshot (snap)
- Floating IP Address (ip)
- Image (image)

5.2 Virtual Machine (vm) Instance Schema

5.2.1 Method: vm.create

Description: Create virtual machine.

Parameters: **imageUri**

- description: The URI of the image
- type: string
- required: true
- restrictions: minLength = 8, maxLength = 128

keyPairId

- description: The instance ID of the key pair
- type: string
- required: false

- restrictions: minLength = 36, maxLength = 36

securityGroupIds

- description: An array of security group instance ID's
- type: array
- required: false
- restrictions: minItems = 1, maxItems = 5

userData

- description: User data for the virtual machine
- type: string
- required: false
- restrictions: minLength = 2, maxLength = 4096

zone

- description: The zone for the virtual machine
- type: string
- required: false
- restrictions: minLength = 2, maxLength = 32

cpuCount

- description: The CPU count
- type: integer
- required: false
- restrictions: minimum = 1

cpuSpeed

- description: The CPU speed in GHz
- type: number
- required: false
- restrictions: none

localStorage

- description: Local storage in GBytes
- type: number
- required: false
- restrictions: none

ram

- description: RAM in MBytes
- type: number
- required: false
- restrictions: none

5.2.2 Method: vm.create.image

Description: Create an image from a virtual machine.

Parameters: No parameters

5.2.3 Method: vm.retrieve

Description: Retrieve virtual machine.

Parameters: No parameters

5.2.4 Method: vm.retrieve.password

Description: Retrieve virtual machine password.

Parameters: No parameters

5.2.5 Method: vm.update.password

Description: Update password on a virtual machine.

Parameters: **password**

- description: Password
- type: string
- required: true
- restrictions: minLength = 5, maxLength = 256

5.2.6 Method: vm.update.start

Description: Start virtual machine.

Parameters: No parameters

5.2.7 Method: vm.update.stop

Description: Stop virtual machine.

Parameters: No parameters

5.2.8 Method: vm.update.reboot

Description: Reboot virtual machine.

Parameters: **rebootType**

- description: Reboot type - hard or soft
- type: string
- required: false

- restrictions: minLength = undefined, maxLength = undefined

5.2.9 Method: vm.delete

Description: Delete virtual machine.

Parameters: No parameters

5.3 Key Pair (kp) Instance Schema

5.3.1 Method: kp.create

Description: Create key pair.

Parameters: No parameters

5.3.2 Method: kp.retrieve

Decription: Retrieve key pair.

Parameters: No parameters

5.3.3 Method: kp.delete

Description: Delete key pair.

Parameters: No parameters

5.4 Security Group (sg) Instance Schema

5.4.1 Method: sg.create

Description: Create security group.

Parameters: No parameters

5.4.2 Method: sg.retrieve

Description: Retrieve security group.

Parameters: No parameters

5.4.3 Method: sg.update.add-access

Description: Add ports to security group.

Parameters:

rules

- description: The security group rules
- type: array
- required: true
- restrictions: minItems = 1, maxItems = 20

5.4.4 Method: sg.update.remove-access

Description: Remove ports from security group.

Parameters: **ruleIndexes**

- description: The indices of the security group rules that should be removed
- type: array
- required: true
- restrictions: minItems = 1, maxItems = 20

5.4.5 Method: sg.delete

Description: Delete security group.

Parameters: No parameters

5.5 Volume Storage (vs) Instance Schema

5.5.1 Method: vs.create

Description: Create volume storage.

Parameters: **sizeInGBytes**

- description: Volume size in GBytes
- type: number
- required: true
- restrictions: none

zone

- description: The zone for the virtual machine
- type: string
- required: false
- restrictions: minLength = 2, maxLength = 32

virtualMachineId

- description: The instance id of the virtual machine
- type: string
- required: false

- restrictions: minLength = 36, maxLength = 36

deviceName

- description: The device name for the attached volume
- type: string
- required: false
- restrictions: minLength = 2, maxLength = 128

5.5.2 Method: vs.retrieve

Description: Retrieve volume storage.

Parameters: No parameters

5.5.3 Method: vs.update.attach

Description: Attach volume storage to a virtual machine.

Parameters: **virtualMachineId**

- description: The instance id of the virtual machine
- type: string
- required: true
- restrictions: minLength = 36, maxLength = 36

deviceName

- description: The device name for the attached volume
- type: string
- required: true
- restrictions: minLength = 2, maxLength = 128

5.5.4 Method: vs.update.detach

Description: Detach volume storage from a virtual machine.

Parameters: No parameters

5.5.5 Method: vs.delete

Description: Delete volume storage.

Parameters: No parameters

5.6 Volume Snapshot (snap) Instance Schema

5.6.1 Method: snap.create

Description: Create volume snapshot from volume storage.

Parameters:

volumeStorageId

- description: **MISSING**
- type: string
- required: true
- restrictions: minLength = 36, maxLength = 36

5.6.2 Method: snap.retrieve

Description: Retrieve volume snapshot.

Parameters: No parameters

5.6.3 Method: snap.delete

Description: Delete volume sanapshot.

Parmeters: No parameters

5.7 Floating IP Address (ip) Instance Schema

5.7.1 Method: ip.create

Description: Create floating ip address.

Providers: No parameters

5.7.2 Method: ip.retrieve

Description: Retrieve floating ip address.

Parameters: No parameters

5.7.3 Method: ip.update.add

Description: Add a floating ip address to a virtual machine.

Parameters: **virtualMachineId**

- description: The instance id of the virtual machine
- type: string
- required: true

- restrictions: minLength = 36, maxLength = 36

5.7.4 Method: ip.update.remove

Description: Remove a floating ip address from a virtual machine.

Parameters: No parameters

5.7.5 Method: ip.delete

Description: Delete floating ip address.

Parameters: No parameters

5.8 Image (image) Instance Schema

5.8.1 Method: image.create

Description: Create an image from a virtual machine.

Parameters:

virtualMachineId

- description: The instance id of the virtual machine
- type: string
- required: true
- restrictions: minLength = 36, maxLength = 36

5.8.2 Method: image.retrieve

Description: Retrieve image.

Parameters: No parameters

5.8.3 Method: image.delete

Description: Delete image.

Parameters: No parameters

CHAPTER 6

Workloads

6.1 Introduction

For information on how to access the API and API authorization see: *Obtaining API Keys*.

The workload API provides an API by which users can create, edit and manage workloads, and activate and deactivate them to create (allocate) or delete (de-allocate) sets of cloud resources. It also provides methods for monitoring and controlling the activation and deactivation process.

All workloads are described by means of a JSON document. This document contains workload elements, which can be added and deleted from the workload. Each workload has a name, which must be unique for the user.

Note: Names are made unique by lower casing and removing all spaces. So "VM 1" will be made unique as "vm1" which means "Vm 1" and "vM1" will be considered the same.

The workload CRUD API methods are used for editing the workload, including adding and deleting workload elements.

Once the workload is defined using the CRUD methods, it can be activated. Activation (or deactivation) is a two step process. First, the workload is planned. The planning process checks that when this workload is activated, the user will not exceed their limits for resource instances. It also checks the availability of resources at the provider, which will indicate the likelihood that this activation process will succeed or not. The output of the planning process is the workload plan, which is a set of workload steps that will need to be carried out to activate the workload.

- There are two types of plan. A plan for activation, and a plan for deactivation.
- The plan is transient, because it depends on the current state of the users instances.
- The plan is not stored in the database, but it is returned as part of the workload.
- The plan is valid for a limited time, it will expire.
- The plan will contain both serial and parallel steps, which indicate the sequence in which the steps will be executed.

The planning process considers the state of the "running workload". The "running workload" is the set of instances that are running that were created for this workload. These instances are from the instance service and are tagged with the workloadId from the current workload in their metadata (a collection). To access these instances (and their status), you use the instance API, not the workload API.

During the planning process, the workload definition is checked against the running workload. For activation, if a workload element is defined in the workload definition that is not in the running workload, then the workload plan will create that instance. If a workload element was deleted from the workload definition, but is in the running workload, then the workload plan will delete that instance. So the activation plan will always try to get the running workload to

match the workload definition. For deactivation, the running workload is checked and the workload plan deletes all elements in the running workload.

If the plan looks good, you can then go ahead and execute the workload.

The steps for workload activation are:

1. Create any key pairs.
2. Create any security groups.
3. Create requested VMs.
4. Create requested VS.
5. Attach any VS to the VM's.

A transaction is the process of activating or deactivating the workload. Once that activation or deactivation has occurred, there is no longer a transaction. The transaction is only used for monitoring the activation or deactivation process and nothing else. We keep the last transaction available for each workload, but only the last transaction, and once the activation or deactivation process (the transaction) has completed (or failed) it is no longer useful - except for knowing what happened.

In general, any errors will be returned either directly as JSON from the REST call (code, message, ticket) or in some cases error information might be embedded into the workload JSON itself.

Once created, workloads have a unique workloadId (Guid), but workload elements are identified only by name within the scope of a workload, they do not have an id. Each workload element name must be unique within the scope of the workload.

To see an example of the api in action see: *Getting Started with Workloads*.

6.2 Workload Methods

6.2.1 Create workload

Create a workload.

POST /workload

Request Body

The workload JSON.

Notes

The workload JSON is validated against a basic schema defined here: *Workload Schema*.

The schema includes: name, description, metadata, parameters and elements (an array). Metadata here is metadata for the workload and can be any valid JSON.

Each workload element is validated against a basic schema defined here: *Workload Element Schema*.

For the schema validation, most properties are optional, but if they are provided, they must match the schema. Properties in the JSON which are not defined in the schema are ignored.

For create, the workload name is required. Must be unique for this user.

A user is limited in the maximum number of workloads they can have at one time. This is defined by the "user limits" for that user.

Example

POST /workload

Returns

The workload JSON. Will include the workloadId.

6.2.2 Clone workload

Clone the workload. This creates a copy of the original workload with a different name and a different workloadId.

POST /workload/clone/<workloadId>

Request Body

- name - required - the name for the new (cloned) workload. Must be unique for this user.

Example

POST /workload/ad5a66c3-8895-4d71-b786-1d129b33326e

Returns

The workload JSON. Will include the workloadId.

6.2.3 Retrieve workload

Retrieve one workload.

GET /workload/<workloadId>

Returns

The workload JSON. Will include the workloadId.

6.2.4 Retrieve multiple workloads

Retrieve multiple workloads.

GET /workload

Query Parameters

- name - optional. If you want to retrieve a workload by name, you can specify the workload name here

Note that even though only one workload will be returned (because workload names are unique) an array will always be returned from this method.

Returns

An array of workloads.

If there is only one workload returned then the full JSON for the workload will be returned.

If there is more than one workload returned then only limited information for the workloads will be returned (no elements, no parameters, no plan, etc.)

6.2.5 Update workload

Update the workload.

PUT /workload/<workloadId>

Request Body

The workload JSON.

Notes

The following properties can be updated by this method: name, description, parameters, metadata. They are replaced in their entirety. To update elements use the element CRUD methods below. Updates need to conform to the same schema requirements as the create above. If the name is changed, it must be unique for this user.

A workload cannot be updated while workload planning or execution is in progress.

When a workload is updated, any existing plan is invalid and hence is removed.

Example

PUT /workload/ad5a66c3-8895-4d71-b786-1d129b33326e

Returns

The workload JSON. Will include the workloadId.

6.2.6 Delete workload

Delete a workload.

DELETE /workload/<workloadId>

Notes

A workload cannot be deleted while workload planning or execution is in progress.

Currently you can delete the workload even if it has running instances. This will cause those running instances to effectively be "orphaned" as they will no longer belong to a workload. So before you delete a workload you should check whether it has running instances.

Example

DELETE /workload/ad5a66c3-8895-4d71-b786-1d129b33326e

Returns

{ "deleted": true } (if the workload was sucessfully deleted)

- or -

{ "deleted": false } (if the workload was already deleted)

6.2.7 Create/Update workload element

Create or update a workload element.

PUT /workload/<workloadId>/element

Request Body

The workload element JSON.

Notes

A workload element cannot be created or updated while workload planning or execution is in progress.

When a workload is updated, any existing plan is invalid and hence is removed.

The workload element JSON is validated against the schema defined here: *Workload Element Schema*.

The schema includes: name, uri, parameters, and metadata. Metadata here is metadata for the workload element and can be any valid JSON. That metadata will be attached to the instance when it is created. name and uri are required. All other parameters are optional.

Returns

The workload element JSON.

6.2.8 Delete workload element

Delete a workload element from the workload.

DELETE /workload/<workloadId>/element

Request Body

The workload element JSON. The only required property is name (to identify the workload element that is to be deleted).

Notes

A workload element cannot be deleted while workload planning or execution is in progress.

When a workload is updated (in this case by having an element deleted) any existing plan is invalid and hence is removed.

Returns

The workload element JSON.

6.2.9 Plan workload

Generate the workload plan. The plan can be for activation or deactivation.

PUT /workload/<workloadId>/plan

Query Parameters

- action = activate | deactivate

Example

PUT /workload/ad5a66c3-8895-4d71-b786-1d129b33326e/plan?action=activate

Returns

The workload element JSON. This will include the plan. The plan property starts with *serial*.

6.2.10 Execute workload plan

Execute the workload plan.

PUT /workload/<workloadId>/execute

Notes

The execute will fail if the workloadStatus is in-progress.

Returns:

```
{
        "workloadStatus": "in-progress",
        "action": "activate",
        "transactionId": "<the transaction id>"
}
```

6.2.11 Retrieve transaction steps

Retrieve transaction steps.

GET /transaction/<transactionId>/steps

Query Parameters

- begin = the beginning Log Sequence Number (LSN). Optional. If omitted then begin = 0.
- end = the ending LSN. Optional. If omitted then end = 10,000.

Notes

When a workload is executed, the results of all the steps in the workload plan are logged to a transaction log. Each entry in the log has a "Log Sequence Number" or LSN. Some of those log entries are relevant to the progress of the step and some are not. To monitor the progress of the step we are mostly interested in when the step started (step status = in-progress) and when it has finished (step status = completed or step status = failed). The *begin* and *end* query parameters can be used to limit the number of records returned. Note that once a step with a specific LSN has been returned it will never change, so the best way to poll with this method is to advance the begin LSN to the LSN of the last step that was returned in the previous call to this method.

Result

An array of log entries. These have a stepId so they can be correlated to the steps in the workload plan.

Each log entry is:

```
{
        "lsn": 456,
        "stepId": "<the unique step id - matches the plan>",
        "timestamp": "<UTC timestamp>",
        "status": "in-progress | completed | failed",
        "reason": "<failure reason if status = failed>",
        "elapsedTimeInSeconds": "<elapsed time in seconds for the workload step>"
}
```

6.2.12 Retrieve transaction errors

Retrieve transaction errors.

GET /transaction/<transactionId>/errors

Notes

In the workload API, errors from the provider side are not necessarily fatal. The workload API implements retries and retry delays. So it may be that an error was received from a provider, but that the operation was retried and succeeded the second time. This method can be used to see details for all errors that occur during the workload execution.

Returns

An array of log entries. These have a stepId so they can be correlated to the steps in the workload plan.

Each log entry is:

```
{
        "lsn": 456,
        "stepId": "<the unique step id - matches the plan>",
        "timestamp": "<UTC timestamp>",
        "code": "<error code>",
        "message": "<error message>",
        "ticket": "<error ticket>"
}
```

6.2.13 Retrieve transaction status

Retrieve transaction status.

GET /transaction/<transactionId>/status

Returns:

```
{
        "transactionId": "<the transaction id>",
        "workloadId": "<the workload id>",
        "status": <the transaction status: in-progress | completed | failed>",
        "reason": <failure reason if status = failed>",
        "stepId": <the stepId of the step that failed, if status = failed>",
        "started": "<UTC timestamp - start time>",
        "ended": "<UTC timestamp - end time>",
        "elapsedTimeInSeconds": "<elapsed time in seconds for transaction>"
}
```

6.2.14 Cancel transaction

Cancel a transaction that is currently in-progress.

PUT /transaction/<transactionId>/cancel

Returns:

```
{
  "transactionId": "<the transaction id>",
  "workloadId": "<the workload id>",
  "action": "cancel",
  "status": "in-progress"
}
```

CHAPTER 7

Workload Schema

Workloads are a collection of workload elements. The workload elements may specify a virtual machine (vm) or volume storage (vs).

This section specifies the *input* schema for a workload.

Note that additional properties will be added to the workload by the workload API. These are explained here: *Getting Started with Workloads*

7.1 name

The workload name. Workload names must be unique (when converted to lower case with no whitespace) for a user.

- type: string
- required for create: true
- required for update: false
- restrictions: minLength = 1, maxLength = 64

7.2 description

The workload description.

- type: string
- required: false
- restrictions: minLength = 1, maxLength = 128

7.3 metadata

The workload metadata. The metadata is a set of key/value pairs, where the key is a string and the value may be a string or a JSON object.

- type: object
- required: false

7.4 elements

The workload elements. These describe what the workload should contain. The order of the workload elements is not significant.

- type: array
- required: false
- restrictions: minItems = 1, maxItems = 100

See: *Workload Element Schema*

CHAPTER 8

Workload Element Schema

8.1 name

The workload element name. Workload element names must be unique (when converted to lower case with no whitespace) within a workload.

- type: string
- required: true
- restrictions: minLength = 1, maxLength = 64

8.2 uri

The resource URI. This identifies the resource type of the workload element (in the ComputeNext catalog).

Currently only two resource types are allowed for workload elements: vm and vs.

- type: string
- required: true
- restrictions: minLength = 1, maxLength = 256

8.3 parameters

Parameters for the workload element.

- type: object
- required: false

The parameters vary by the resource type.

In general the parameters that can be used for the equivalent "create" action can be specified here.

A virtual machine (vm) workload element can use the parameters allowed for *vm.create*

A volume storage (vs) workload element can use the parameters allowed for *vs.create*

See Resource Types, Actions and Parameters

However, there are some additional parameters available for the workload element that allow instances to be specified by *name*, not instance ID.

Virtual Machine (vm)

8.3.1 keyPair

The key pair name that should be used for this VM. Maps to keyPairId.

8.3.2 securityGroups

An array of security group names to be used for this VM. Maps to securityGroupIds.

Volume Storage (vs)

8.3.3 attachTarget

The name of the VM that this VS should be attached to. Maps to virtualMachineId.

8.4 metadata

The workload element metadata. The metadata is a set of key/value pairs, where the key is a string and the value may be a string or a JSON object. The workload element metadata will be added to the instance metadata when the instance is created.

- type: object
- required: false

There are several reserved metadata key names used by the workload API

- name (the workload element name)
- workloadId (the workload ID)
- transactionId (the transaction ID)

CHAPTER 9

Getting Started with Instances

Please read *Getting Started with Workloads* first, which explains how to install and setup the *runcws* command line tool used for the examples in this section. You should also have read and understood *Instances* with background information about instances before proceeding to do this section.

The *instance API* is a lower level API used for individual instances.

The *workload API* is a higher level API used for collections of instances. The workload API calls the instance API.

Note: To create any instances you must first enter your payment information into the ComputeNext website. If you do not have payment information you will receive a "403 Forbidden" error.

In all of the following tutorial examples, sample json files with parameter data are taken from the demo subdirectory.

9.1 Create A Key Pair

This example will show how to create (and delete) a key pair (kp) using runcws.

Create the instance:

```
>node runcws.js createi demo\kp.create.json
createi (create instance from resource)
options: {
  "url": "http://cws.computenext.com/api/resource/kp/hpcloud/nova/standard",
  "method": "post",
  "json": {
        "metadata": {
          "name": "KP1",
          "description": "my first key pair"
        }
  },
  "auth": {
        "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
        "pass": "<hidden>"
  }
}
##### setting current requestId: 3ebea4a3-0b60-49a4-b0b1-701e6acb54b8
----- RESULT -----
{
  "requestId": "3ebea4a3-0b60-49a4-b0b1-701e6acb54b8",
  "created": "2013-12-20T01:16:15.041Z",
  "updated": "2013-12-20T01:16:15.041Z",
  "ownerId": "d00f16e6-7503-4838-a658-8b60594ff285",
```

```
  "requestStatus": "in-progress",
  "resourceUri": "kp/hpcloud/nova/standard",
  "resourceType": "kp",
  "provider": "hpcloud",
  "region": "nova",
  "connector": "openStack.compute",
  "parameters": {
        "action": "kp.create",
        "instanceId": "c4c2ddc6-9366-41c9-a813-9abd88b924b7",
        "kp_providerResourceId": "standard",
        "zone": "nova"
  },
  "metadata": {
        "name": "KP1",
        "description": "my first key pair"
  }
}
```


Get the request:


```
>node runcws.js getr
getr (retrieve request)
options: {
  "url": "http://cws.computenext.com/api/request/3ebea4a3-0b60-49a4-b0b1-701e6acb54b8",
  "method": "get",
  "auth": {
        "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
        "pass": "<hidden>"
  }
}
##### setting current instanceId: c4c2ddc6-9366-41c9-a813-9abd88b924b7
----- RESULT -----
[
  {
        "requestId": "3ebea4a3-0b60-49a4-b0b1-701e6acb54b8",
        "instanceId": "c4c2ddc6-9366-41c9-a813-9abd88b924b7",
        "created": "2013-12-20T01:16:15.041Z",
        "updated": "2013-12-20T01:16:15.942Z",
        "ownerId": "d00f16e6-7503-4838-a658-8b60594ff285",
        "requestStatus": "completed",
        "resourceUri": "kp/hpcloud/nova/standard",
        "resourceType": "kp",
        "provider": "hpcloud",
        "region": "nova",
        "connector": "openStack.compute",
        "parameters": {
          "action": "kp.create",
          "instanceId": "c4c2ddc6-9366-41c9-a813-9abd88b924b7",
          "kp_providerResourceId": "standard",
          "zone": "nova"
        },
        "metadata": {
          "name": "KP1",
          "description": "my first key pair"
        },
        "results": {
          "providerInstanceId": "c4c2ddc6-9366-41c9-a813-9abd88b924b7",
          "keyFingerprint": "a6:0d:bd:20:56:0d:a7:47:b4:01:a1:65:10:dd:58:5f",
          "privateKey": "<HIDDEN>",
```

```
          "publicKey": "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAAAgQDV4ijOycM6VmQZoCQCu74MkonLYKWCr0976ZlU8
QsDo+e5R0M+6LBTTxjEn/fQUZejDIaIkWJXhRR4W80mzevifrQ== \n",
          "instanceStatus": "created"
        }
  }
]
```

The requestStatus is *completed*.

Note that most of the time a request will complete quickly with either *completed* or *failed* status. The *failed* status will include an error message and error code in the response. If you see a request that seems to be stuck for some time with the *in-progress* status then it is possible that the response has been lost. This is not likely to occur in normal operation, but could possibly occur under some error conditions. If this occurs then try the original request again.

Note that when the request is returned secure properties (the privateKey) are hidden.

Get the instance:

```
>node runcws.js geti
geti (retrieve instance)
options: {
  "url": "http://cws.computenext.com/api/instance/c4c2ddc6-9366-41c9-a813-9abd88b924b7",
  "method": "get",
  "auth": {
        "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
        "pass": "<hidden>"
  }
}
----- RESULT -----
{
  "instanceId": "c4c2ddc6-9366-41c9-a813-9abd88b924b7",
  "created": "2013-12-20T01:16:15.938Z",
  "updated": "2013-12-20T01:16:15.938Z",
  "ownerId": "d00f16e6-7503-4838-a658-8b60594ff285",
  "resourceUri": "kp/hpcloud/nova/standard",
  "resourceType": "kp",
  "provider": "hpcloud",
  "region": "nova",
  "attributes": {
        "providerInstanceId": "c4c2ddc6-9366-41c9-a813-9abd88b924b7",
        "publicKey": "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAAAgQDV4ijOycM6VmQZoCQCu74MkonLYKWCr0976ZlU8we
Do+e5R0M+6LBTTxjEn/fQUZejDIaIkWJXhRR4W80mzevifrQ== \n",
        "privateKey": "-----BEGIN RSA PRIVATE KEY-----\nMIICXQIBAAKBgQDV4ijOycM6VmQZoCQCu74MkonLYKWCr
ws\nfvJB/TiIQsDo+e5R0M+6LBTTxjEn/fQUZejDIaIkWJXhRR4W80mzevifrQIDAQAB\nAoGAVqq51nEzRqRTE38smF7y9605YMv
3H2v/RndLu2Sj/6ar6Yp\ncYR8jyOIwQIATBn7XWoayeY6EeAu/FU9KFqtTWYQUtXG+YECQQD9Pdi1IQstqzbf\nE+TjQ1WBiBM3
0+FDja3KMeaGvjWCV591T48EVugUzvDiGY1dsk\ncv2dUeBqILmgcl9yZN1TNUWl0k6r+Ulq5QJBAJwfC8GmzHBsNFjUt/BPq6A+
k/k3xqzm07jbz7T5CxsUq+Vn0x7XPjlKfxqwM5N30z+NDQWG9XSzOzTd\n4Huw6NWPa2GabIHkSnMFAkANjrBz7Qp9l7FJVGOn29
SA PRIVATE KEY-----\n",
        "keyFingerprint": "a6:0d:bd:20:56:0d:a7:47:b4:01:a1:65:10:dd:58:5f",
        "instanceStatus": "created",
        "transientStatus": false
  },
  "attributeTimestamps": {
        "instanceStatus": "2013-12-20T01:16:15.936Z"
  },
  "metadata": {
        "name": "KP1",
        "description": "my first key pair"
  },
```

```
  "parameters": {
        "kp_providerResourceId": "standard",
        "zone": "nova"
  }
}
```

The instanceStatus is *created*.

List the requests:

```
>node runcws.js listr
listr (retrieve multiple requests)
options: {
  "url": "http://cws.computenext.com/api/request?cleanup=true",
  "method": "get",
  "auth": {
        "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
        "pass": "<hidden>"
  }
}
----- RESULT -----
[]
```

No requests - the request has been cleaned-up because it was *completed*.

List instances:

```
>node runcws.js listi
listi (retrieve multiple instances)
options: {
  "url": "http://cws.computenext.com/api/instance?cleanup=true",
  "method": "get",
  "auth": {
        "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
        "pass": "<hidden>"
  }
}
----- RESULT -----
[
  {
        "instanceId": "c4c2ddc6-9366-41c9-a813-9abd88b924b7",
        "created": "2013-12-20T01:16:15.938Z",
        "updated": "2013-12-20T01:16:15.938Z",
        "ownerId": "d00f16e6-7503-4838-a658-8b60594ff285",
        "resourceUri": "kp/hpcloud/nova/standard",
        "resourceType": "kp",
        "provider": "hpcloud",
        "region": "nova",
        "attributes": {
          "providerInstanceId": "c4c2ddc6-9366-41c9-a813-9abd88b924b7",
          "publicKey": "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAAAgQDV4ijOycM6VmQZoCQCu74MkonLYKWCr0976ZlU8
QsDo+e5R0M+6LBTTxjEn/fQUZejDIaIkWJXhRR4W80mzevifrQ== \n",
          "privateKey": "-----BEGIN RSA PRIVATE KEY-----\nMIICXQIBAAKBgQDV4ijOycM6VmQZoCQCu74MkonLYKW
0iws\nfvJB/TiIQsDo+e5R0M+6LBTTxjEn/fQUZejDIaIkWJXhRR4W80mzevifrQIDAQAB\nAoGAVqq51nEzRqRTE38smF7y9605Y
XW3H2v/RndLu2Sj/6ar6Yp\ncYR8jyOIwQIATBn7XWoayeY6EeAu/FU9KFqtTWYQUtXG+YECQQD9Pdi1IQstqzbf\nE+TjQ1WBiBN
x70+FDja3KMeaGvjWCV591T48EVugUzvDiGY1dsk\ncv2dUeBqILmgcl9yZN1TNUWl0k6r+Ulq5QJBAJwfC8GmzHBsNFjUt/BPq6A
6uk/k3xqzm07jbz7T5CxsUq+Vn0x7XPjlKfxqwM5N30z+NDQWG9XSzOzTd\n4Huw6NWPa2GabIHkSnMFAkANjrBz7Qp9l7FJVGOn2
 RSA PRIVATE KEY-----\n",
          "keyFingerprint": "a6:0d:bd:20:56:0d:a7:47:b4:01:a1:65:10:dd:58:5f",
          "instanceStatus": "created",
```

```
          "transientStatus": false
        },
        "attributeTimestamps": {
          "instanceStatus": "2013-12-20T01:16:15.936Z"
        },
        "metadata": {
          "name": "KP1",
          "description": "my first key pair"
        },
        "parameters": {
          "kp_providerResourceId": "standard",
          "zone": "nova"
        }
  }
]
```

Delete the instance:

```
>node runcws.js deletei
deletei (delete instance)
options: {
  "url": "http://cws.computenext.com/api/instance/c4c2ddc6-9366-41c9-a813-9abd88b924b7",
  "method": "delete",
  "auth": {
        "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
        "pass": "<hidden>"
  }
}
##### setting current requestId: efa6371a-634b-4073-83ff-f694e3d6fc41
----- RESULT -----
{
  "requestId": "efa6371a-634b-4073-83ff-f694e3d6fc41",
  "instanceId": "c4c2ddc6-9366-41c9-a813-9abd88b924b7",
  "created": "2013-12-20T01:23:26.909Z",
  "updated": "2013-12-20T01:23:26.909Z",
  "ownerId": "d00f16e6-7503-4838-a658-8b60594ff285",
  "requestStatus": "in-progress",
  "resourceUri": "kp/hpcloud/nova/standard",
  "resourceType": "kp",
  "provider": "hpcloud",
  "region": "nova",
  "connector": "openStack.compute",
  "parameters": {
        "providerInstanceId": "c4c2ddc6-9366-41c9-a813-9abd88b924b7",
        "action": "kp.delete",
        "instanceId": "c4c2ddc6-9366-41c9-a813-9abd88b924b7",
        "kp_providerResourceId": "standard",
        "zone": "nova"
  }
}
```

Get the request:

```
>node runcws.js getr
getr (retrieve request)
options: {
  "url": "http://cws.computenext.com/api/request/efa6371a-634b-4073-83ff-f694e3d6fc41",
  "method": "get",
  "auth": {
```

```
        "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
        "pass": "<hidden>"
  }
}
----- RESULT -----
[
  {
        "requestId": "efa6371a-634b-4073-83ff-f694e3d6fc41",
        "instanceId": "c4c2ddc6-9366-41c9-a813-9abd88b924b7",
        "created": "2013-12-20T01:23:26.909Z",
        "updated": "2013-12-20T01:23:27.409Z",
        "ownerId": "d00f16e6-7503-4838-a658-8b60594ff285",
        "requestStatus": "completed",
        "resourceUri": "kp/hpcloud/nova/standard",
        "resourceType": "kp",
        "provider": "hpcloud",
        "region": "nova",
        "connector": "openStack.compute",
        "parameters": {
          "providerInstanceId": "c4c2ddc6-9366-41c9-a813-9abd88b924b7",
          "action": "kp.delete",
          "instanceId": "c4c2ddc6-9366-41c9-a813-9abd88b924b7",
          "kp_providerResourceId": "standard",
          "zone": "nova"
        },
        "results": {
          "instanceStatus": "deleted"
        }
  }
]
```

The requestStatus is *completed*.

Get the instance:

```
>node runcws.js geti
geti (retrieve instance)
options: {
  "url": "http://cws.computenext.com/api/instance/c4c2ddc6-9366-41c9-a813-9abd88b924b7",
  "method": "get",
  "auth": {
        "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
        "pass": "<hidden>"
  }
}
----- RESULT -----
{
  "instanceId": "c4c2ddc6-9366-41c9-a813-9abd88b924b7",
  "created": "2013-12-20T01:16:15.938Z",
  "updated": "2013-12-20T01:23:27.403Z",
  "ownerId": "d00f16e6-7503-4838-a658-8b60594ff285",
  "resourceUri": "kp/hpcloud/nova/standard",
  "resourceType": "kp",
  "provider": "hpcloud",
  "region": "nova",
  "providerResourceId": "standard",
  "attributes": {
        "providerInstanceId": "c4c2ddc6-9366-41c9-a813-9abd88b924b7",
        "publicKey": "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAAAgQDV4ijOycM6VmQZoCQCu74MkonLYKWCr0976ZlU8we
```

```
Do+e5R0M+6LBTTxjEn/fQUZejDIaIkWJXhRR4W80mzevifrQ== \n",
        "privateKey": "-----BEGIN RSA PRIVATE KEY-----\nMIICXQIBAAKBgQDV4ijOycM6VmQZoCQCu74MkonLYKWCr
ws\nfvJB/TiIQsDo+e5R0M+6LBTTxjEn/fQUZejDIaIkWJXhRR4W80mzevifrQIDAQAB\nAoGAVqq51nEzRqRTE38smF7y9605YMv
3H2v/RndLu2Sj/6ar6Yp\ncYR8jyOIwQIATBn7XWoayeY6EeAu/FU9KFqtTWYQUtXG+YECQQD9Pdi1IQstqzbf\nE+TjQ1WBiBM3r
0+FDja3KMeaGvjWCV591T48EVugUzvDiGY1dsk\ncv2dUeBqILmgcl9yZN1TNUWl0k6r+Ulq5QJBAJwfC8GmzHBsNFjUt/BPq6A+l
k/k3xqzm07jbz7T5CxsUq+Vn0x7XPjlKfxqwM5N30z+NDQWG9XSzOzTd\n4Huw6NWPa2GabIHkSnMFAkANjrBz7Qp9l7FJVGOn29j
SA PRIVATE KEY-----\n",
        "keyFingerprint": "a6:0d:bd:20:56:0d:a7:47:b4:01:a1:65:10:dd:58:5f",
        "instanceStatus": "deleted",
        "transientStatus": false
  },
  "attributeTimestamps": {
        "instanceStatus": "2013-12-20T01:23:27.381Z"
  },
  "metadata": {
        "name": "KP1",
        "description": "my first key pair"
  },
  "parameters": {
        "kp_providerResourceId": "standard",
        "zone": "nova"
  }
}
```

The instanceStatus is *deleted*.

Note that at some point *deleted* instances will be cleaned-up from the database and will return a "404 Not Found".

9.2 Create A Virtual Machine

This example will show how to create (and delete) a virtual machine (vm) using runcws.

Create the instance:

```
>node runcws.js createi demo\vm.create.json
createi (create instance from resource)
options: {
  "url": "http://cws.computenext.com/api/resource/vm/hpcloud/nova/standard.small",
  "method": "post",
  "json": {
        "imageUri": "image/hpcloud/nova/ami-00000075",
        "metadata": {
          "name": "VM-1",
          "description": "my first virtual machine"
        }
  },
  "auth": {
        "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
        "pass": "<hidden>"
  }
}
##### setting current requestId: 6b7fd23c-05d9-4a88-b8fd-efa859c1a04a
----- RESULT -----
{
  "requestId": "6b7fd23c-05d9-4a88-b8fd-efa859c1a04a",
  "created": "2014-01-09T22:51:39.039Z",
  "updated": "2014-01-09T22:51:39.039Z",
```

```
  "ownerId": "d00f16e6-7503-4838-a658-8b60594ff285",
  "requestStatus": "in-progress",
  "resourceUri": "vm/hpcloud/nova/standard.small",
  "resourceType": "vm",
  "provider": "hpcloud",
  "region": "nova",
  "connector": "openStack.compute",
  "parameters": {
        "imageUri": "image/hpcloud/nova/ami-00000075",
        "action": "vm.create",
        "instanceId": "615d07cf-d0e5-4325-ac02-1148d6e0d50b",
        "vm_providerResourceId": "Standard.small",
        "zone": "nova",
        "cpuCount": "2",
        "cpuSpeed": "1.2",
        "localStorage": "60",
        "ram": "2",
        "username": "ubuntu",
        "image_providerResourceId": "ami-00000075"
  },
  "metadata": {
        "name": "VM-1",
        "description": "my first virtual machine"
  }
}
```

Get the request:

```
>node runcws.js getr
getr (retrieve request)
options: {
  "url": "http://cws.computenext.com/api/request/6b7fd23c-05d9-4a88-b8fd-efa859c1a04a",
  "method": "get",
  "auth": {
        "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
        "pass": "<hidden>"
  }
}
##### setting current instanceId: 615d07cf-d0e5-4325-ac02-1148d6e0d50b
----- RESULT -----
[
  {
        "requestId": "6b7fd23c-05d9-4a88-b8fd-efa859c1a04a",
        "instanceId": "615d07cf-d0e5-4325-ac02-1148d6e0d50b",
        "created": "2014-01-09T22:51:39.039Z",
        "updated": "2014-01-09T22:51:42.394Z",
        "ownerId": "d00f16e6-7503-4838-a658-8b60594ff285",
        "requestStatus": "completed",
        "resourceUri": "vm/hpcloud/nova/standard.small",
        "resourceType": "vm",
        "provider": "hpcloud",
        "region": "nova",
        "connector": "openStack.compute",
        "parameters": {
          "imageUri": "image/hpcloud/nova/ami-00000075",
          "action": "vm.create",
          "instanceId": "615d07cf-d0e5-4325-ac02-1148d6e0d50b",
          "vm_providerResourceId": "Standard.small",
          "zone": "nova",
```

```
          "cpuCount": "2",
          "cpuSpeed": "1.2",
          "localStorage": "60",
          "ram": "2",
          "username": "ubuntu",
          "image_providerResourceId": "ami-00000075"
        },
        "metadata": {
          "name": "VM-1",
          "description": "my first virtual machine"
        },
        "results": {
          "providerInstanceId": 2818157,
          "instanceStatus": "creating",
          "password": "<HIDDEN>"
        }
  }
]
```

The requestStatus is *completed*.

Note that the password property is hidden in the request.

Get the instance:

```
>node runcws.js geti
geti (retrieve instance)
options: {
  "url": "http://cws.computenext.com/api/instance/615d07cf-d0e5-4325-ac02-1148d6e0d50b",
  "method": "get",
  "auth": {
        "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
        "pass": "<hidden>"
  }
}
----- RESULT -----
{
  "instanceId": "615d07cf-d0e5-4325-ac02-1148d6e0d50b",
  "created": "2014-01-09T22:51:42.389Z",
  "updated": "2014-01-09T22:51:42.389Z",
  "ownerId": "d00f16e6-7503-4838-a658-8b60594ff285",
  "resourceUri": "vm/hpcloud/nova/standard.small",
  "resourceType": "vm",
  "provider": "hpcloud",
  "region": "nova",
  "attributes": {
        "providerInstanceId": 2818157,
        "password": "bAZRf2qQ5VDdE76t",
        "instanceStatus": "creating",
        "transientStatus": true
  },
  "attributeTimestamps": {
        "password": "2014-01-09T22:51:42.384Z",
        "instanceStatus": "2014-01-09T22:51:42.387Z"
  },
  "metadata": {
        "name": "VM-1",
        "description": "my first virtual machine"
  },
```

```
  "parameters": {
        "imageUri": "image/hpcloud/nova/ami-00000075",
        "vm_providerResourceId": "Standard.small",
        "zone": "nova",
        "cpuCount": "2",
        "cpuSpeed": "1.2",
        "localStorage": "60",
        "ram": "2",
        "username": "ubuntu",
        "image_providerResourceId": "ami-00000075"
  }
}
```


Note that the instanceStatus is *creating* which means that the vm instance itself is not completely created - it has not reached a stable state.

The *transientStatus* property is a hint that we need to keep polling for the instanceStatus to change.

So, we need to "retrieve instance" again - this time with *refresh* so that the instance status will be fetched from the provider side.

Get instance (with refresh):


```
>node runcws.js getir
getir (retrieve instance (with refresh))
options: {
  "url": "http://cws.computenext.com/api/instance/615d07cf-d0e5-4325-ac02-1148d6e0d50b?refresh=true",
  "method": "get",
  "auth": {
        "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
        "pass": "<hidden>"
  }
}
##### setting current requestId: fa948a88-ed48-4c93-a69b-1f2becd11ae9
----- RESULT -----
{
  "requestId": "fa948a88-ed48-4c93-a69b-1f2becd11ae9",
  "instanceId": "615d07cf-d0e5-4325-ac02-1148d6e0d50b",
  "created": "2014-01-09T22:58:08.462Z",
  "updated": "2014-01-09T22:58:08.462Z",
  "ownerId": "d00f16e6-7503-4838-a658-8b60594ff285",
  "requestStatus": "in-progress",
  "resourceUri": "vm/hpcloud/nova/standard.small",
  "resourceType": "vm",
  "provider": "hpcloud",
  "region": "nova",
  "connector": "openStack.compute",
  "parameters": {
        "providerInstanceId": 2818157,
        "action": "vm.retrieve",
        "instanceId": "615d07cf-d0e5-4325-ac02-1148d6e0d50b",
        "vm_providerResourceId": "Standard.small",
        "zone": "nova"
  }
}
```


Note that "?refresh=true" was specified on the query parameters.

Note that a *request* is returned, not an instance.

Get the request:

```
>node runcws.js getr
getr (retrieve request)
options: {
  "url": "http://cws.computenext.com/api/request/fa948a88-ed48-4c93-a69b-1f2becd11ae9",
  "method": "get",
  "auth": {
        "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
        "pass": "<hidden>"
  }
}
----- RESULT -----
[
  {
        "requestId": "fa948a88-ed48-4c93-a69b-1f2becd11ae9",
        "instanceId": "615d07cf-d0e5-4325-ac02-1148d6e0d50b",
        "created": "2014-01-09T22:58:08.462Z",
        "updated": "2014-01-09T22:58:08.899Z",
        "ownerId": "d00f16e6-7503-4838-a658-8b60594ff285",
        "requestStatus": "completed",
        "resourceUri": "vm/hpcloud/nova/standard.small",
        "resourceType": "vm",
        "provider": "hpcloud",
        "region": "nova",
        "connector": "openStack.compute",
        "parameters": {
          "providerInstanceId": 2818157,
          "action": "vm.retrieve",
          "instanceId": "615d07cf-d0e5-4325-ac02-1148d6e0d50b",
          "vm_providerResourceId": "Standard.small",
          "zone": "nova"
        },
        "results": {
          "instanceStatus": "running",
          "privateIpAddress": "10.3.118.211",
          "publicIpAddress": "15.185.250.68"
        }
  }
]
```

Now we can see that the request is *completed* and the instanceStatus is *running*.

Get the instance (no refresh):

```
>node runcws.js geti
geti (retrieve instance)
options: {
  "url": "http://cws.computenext.com/api/instance/615d07cf-d0e5-4325-ac02-1148d6e0d50b",
  "method": "get",
  "auth": {
        "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
        "pass": "<hidden>"
  }
}
----- RESULT -----
{
  "instanceId": "615d07cf-d0e5-4325-ac02-1148d6e0d50b",
  "created": "2014-01-09T22:51:42.389Z",
  "updated": "2014-01-09T22:58:08.891Z",
```

```
  "ownerId": "d00f16e6-7503-4838-a658-8b60594ff285",
  "resourceUri": "vm/hpcloud/nova/standard.small",
  "resourceType": "vm",
  "provider": "hpcloud",
  "region": "nova",
  "providerResourceId": "Standard.small",
  "attributes": {
        "providerInstanceId": 2818157,
        "password": "bAZRf2qQ5VDdE76t",
        "instanceStatus": "running",
        "transientStatus": false,
        "privateIpAddress": "10.3.118.211",
        "publicIpAddress": "15.185.250.68"
  },
  "attributeTimestamps": {
        "password": "2014-01-09T22:58:08.858Z",
        "instanceStatus": "2014-01-09T22:58:08.861Z",
        "privateIpAddress": "2014-01-09T22:58:08.859Z",
        "publicIpAddress": "2014-01-09T22:58:08.860Z"
  },
  "metadata": {
        "name": "VM-1",
        "description": "my first virtual machine"
  },
  "parameters": {
        "imageUri": "image/hpcloud/nova/ami-00000075",
        "vm_providerResourceId": "Standard.small",
        "zone": "nova",
        "cpuCount": "2",
        "cpuSpeed": "1.2",
        "localStorage": "60",
        "ram": "2",
        "username": "ubuntu",
        "image_providerResourceId": "ami-00000075"
  }
}
```


The instanceStatus is now *running* and the password and IP addresses are available on the *attributes*.

Delete the instance:


```
>node runcws.js deletei
deletei (delete instance)
options: {
  "url": "http://cws.computenext.com/api/instance/615d07cf-d0e5-4325-ac02-1148d6e0d50b",
  "method": "delete",
  "auth": {
        "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
        "pass": "<hidden>"
  }
}
##### setting current requestId: 296b6f68-b003-4fd1-aa9e-5fb979d5d42a
----- RESULT -----
{
  "requestId": "296b6f68-b003-4fd1-aa9e-5fb979d5d42a",
  "instanceId": "615d07cf-d0e5-4325-ac02-1148d6e0d50b",
  "created": "2014-01-09T23:05:09.072Z",
  "updated": "2014-01-09T23:05:09.072Z",
  "ownerId": "d00f16e6-7503-4838-a658-8b60594ff285",
```

```
  "requestStatus": "in-progress",
  "resourceUri": "vm/hpcloud/nova/standard.small",
  "resourceType": "vm",
  "provider": "hpcloud",
  "region": "nova",
  "connector": "openStack.compute",
  "parameters": {
        "providerInstanceId": 2818157,
        "action": "vm.delete",
        "instanceId": "615d07cf-d0e5-4325-ac02-1148d6e0d50b",
        "vm_providerResourceId": "Standard.small",
        "zone": "nova"
  }
}
```

Get the request:

```
>node runcws.js getr
getr (retrieve request)
options: {
  "url": "http://cws.computenext.com/api/request/296b6f68-b003-4fd1-aa9e-5fb979d5d42a",
  "method": "get",
  "auth": {
        "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
        "pass": "<hidden>"
  }
}
----- RESULT -----
[
  {
        "requestId": "296b6f68-b003-4fd1-aa9e-5fb979d5d42a",
        "instanceId": "615d07cf-d0e5-4325-ac02-1148d6e0d50b",
        "created": "2014-01-09T23:05:09.072Z",
        "updated": "2014-01-09T23:05:09.588Z",
        "ownerId": "d00f16e6-7503-4838-a658-8b60594ff285",
        "requestStatus": "completed",
        "resourceUri": "vm/hpcloud/nova/standard.small",
        "resourceType": "vm",
        "provider": "hpcloud",
        "region": "nova",
        "connector": "openStack.compute",
        "parameters": {
          "providerInstanceId": 2818157,
          "action": "vm.delete",
          "instanceId": "615d07cf-d0e5-4325-ac02-1148d6e0d50b",
          "vm_providerResourceId": "Standard.small",
          "zone": "nova"
        },
        "results": {
          "instanceStatus": "deleted"
        }
  }
]
```

The requestStatus is *completed*.

Get the instance:

```
>node runcws.js geti
geti (retrieve instance)
options: {
  "url": "http://cws.computenext.com/api/instance/615d07cf-d0e5-4325-ac02-1148d6e0d50b",
  "method": "get",
  "auth": {
        "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
        "pass": "<hidden>"
  }
}
----- RESULT -----
{
  "instanceId": "615d07cf-d0e5-4325-ac02-1148d6e0d50b",
  "created": "2014-01-09T22:51:42.389Z",
  "updated": "2014-01-09T23:05:09.573Z",
  "ownerId": "d00f16e6-7503-4838-a658-8b60594ff285",
  "resourceUri": "vm/hpcloud/nova/standard.small",
  "resourceType": "vm",
  "provider": "hpcloud",
  "region": "nova",
  "providerResourceId": "Standard.small",
  "attributes": {
        "providerInstanceId": 2818157,
        "password": "bAZRf2qQ5VDdE76t",
        "instanceStatus": "deleted",
        "transientStatus": false,
        "privateIpAddress": "10.3.118.211",
        "publicIpAddress": "15.185.250.68"
  },
  "attributeTimestamps": {
        "password": "2014-01-09T22:58:08.858Z",
        "instanceStatus": "2014-01-09T23:05:09.515Z",
        "privateIpAddress": "2014-01-09T22:58:08.859Z",
        "publicIpAddress": "2014-01-09T22:58:08.860Z"
  },
  "metadata": {
        "name": "VM-1",
        "description": "my first virtual machine"
  },
  "parameters": {
        "imageUri": "image/hpcloud/nova/ami-00000075",
        "vm_providerResourceId": "Standard.small",
        "zone": "nova",
        "cpuCount": "2",
        "cpuSpeed": "1.2",
        "localStorage": "60",
        "ram": "2",
        "username": "ubuntu",
        "image_providerResourceId": "ami-00000075"
  }
}
```

The instanceStatus is *deleted*.

Note that sometimes you will see the vm in the *deleting* state (which is transient) before it moves to *deleted* state.

Note that at some point *deleted* instances will be cleaned-up from the database and will return a "404 Not Found".

9.3 Create A Private Image from a Virtual Machine

This example will show how to create a private image (image) from a virtual machine (vm).

We start with an existing vm instance from which we want to create a private image.

Get the existing *vm* instance:

```
>node runcws.js geti
geti (retrieve instance)
options: {
  "url": "http://cws.computenext.com/api/instance/3ebb5243-2a11-4da0-844c-60f6f087bcfd",
  "method": "get",
  "auth": {
        "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
        "pass": "<hidden>"
  }
}
----- RESULT -----
{
  "instanceId": "3ebb5243-2a11-4da0-844c-60f6f087bcfd",
  "created": "2014-01-09T23:27:16.793Z",
  "updated": "2014-01-09T23:28:35.070Z",
  "ownerId": "d00f16e6-7503-4838-a658-8b60594ff285",
  "resourceUri": "vm/hpcloud/nova/standard.small",
  "resourceType": "vm",
  "provider": "hpcloud",
  "region": "nova",
  "providerResourceId": "Standard.small",
  "attributes": {
        "providerInstanceId": 2818403,
        "password": "S3m7cbqC78YhGQ7i",
        "instanceStatus": "running",
        "transientStatus": false,
        "privateIpAddress": "10.3.216.96",
        "publicIpAddress": "15.185.244.42"
  },
  "attributeTimestamps": {
        "password": "2014-01-09T23:28:35.043Z",
        "instanceStatus": "2014-01-09T23:28:35.047Z",
        "privateIpAddress": "2014-01-09T23:28:35.043Z",
        "publicIpAddress": "2014-01-09T23:28:35.045Z"
  },
  "metadata": {
        "name": "VM-1",
        "description": "my first virtual machine"
  },
  "parameters": {
        "imageUri": "image/hpcloud/nova/ami-00000075",
        "vm_providerResourceId": "Standard.small",
        "zone": "nova",
        "cpuCount": "2",
        "cpuSpeed": "1.2",
        "localStorage": "60",
        "ram": "2",
        "username": "ubuntu",
        "image_providerResourceId": "ami-00000075"
  }
}
```

The vm instanceStatus is *running*.

Create the private *image* instance from the *vm* instance:

```
>node runcws.js createimage demo\vm.create.image.json
createimage (create an image instance from a VM instance)
options: {
  "url": "http://cws.computenext.com/api/instance/3ebb5243-2a11-4da0-844c-60f6f087bcfd?action=image",
  "method": "post",
  "json": {
        "metadata": {
          "name": "IMAGE-1",
          "description": "my first image"
        }
  },
  "auth": {
        "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
        "pass": "<hidden>"
  }
}
##### setting current requestId: 79ca5b31-f42a-4489-b354-c0e3a7dfb473
##### setting current instanceId: e5cff882-8d95-4f8e-9087-531075a0b0e6
----- RESULT -----
{
  "requestId": "79ca5b31-f42a-4489-b354-c0e3a7dfb473",
  "instanceId": "e5cff882-8d95-4f8e-9087-531075a0b0e6",
  "created": "2014-01-10T00:27:32.368Z",
  "updated": "2014-01-10T00:27:32.368Z",
  "ownerId": "d00f16e6-7503-4838-a658-8b60594ff285",
  "requestStatus": "in-progress",
  "resourceUri": "image/hpcloud/nova/",
  "resourceType": "image",
  "provider": "hpcloud",
  "region": "nova",
  "connector": "openStack.compute",
  "parameters": {
        "providerInstanceId": 2818403,
        "action": "vm.create.image",
        "instanceId": "e5cff882-8d95-4f8e-9087-531075a0b0e6",
        "vm_providerResourceId": "Standard.small",
        "zone": "nova"
  },
  "metadata": {
        "name": "IMAGE-1",
        "description": "my first image"
  }
}
```

Note that this is an action on the *vm* instance. The JSON file supplied to runcws (vm.create.image.json) is basically to supply metadata for the private image instance only.

Get the request:

```
>node runcws.js getr
getr (retrieve request)
options: {
  "url": "http://cws.computenext.com/api/request/79ca5b31-f42a-4489-b354-c0e3a7dfb473",
  "method": "get",
  "auth": {
        "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
```

```
        "pass": "<hidden>"
  }
}
----- RESULT -----
[
  {
        "requestId": "79ca5b31-f42a-4489-b354-c0e3a7dfb473",
        "instanceId": "e5cff882-8d95-4f8e-9087-531075a0b0e6",
        "created": "2014-01-10T00:27:32.368Z",
        "updated": "2014-01-10T00:27:33.302Z",
        "ownerId": "d00f16e6-7503-4838-a658-8b60594ff285",
        "requestStatus": "completed",
        "resourceUri": "image/hpcloud/nova/",
        "resourceType": "image",
        "provider": "hpcloud",
        "region": "nova",
        "connector": "openStack.compute",
        "parameters": {
          "providerInstanceId": 2818403,
          "action": "vm.create.image",
          "instanceId": "e5cff882-8d95-4f8e-9087-531075a0b0e6",
          "vm_providerResourceId": "Standard.small",
          "zone": "nova"
        },
        "metadata": {
          "name": "IMAGE-1",
          "description": "my first image"
        },
        "results": {
          "providerInstanceId": "ami-00058180",
          "instanceStatus": "creating"
        }
  }
]
```

The requestStatus is *completed*. The image instanceStatus is *creating* so we should retrieve the image instance again, with refresh.

Get the *image* instance (with refresh):

```
>node runcws.js getir
getir (retrieve instance (with refresh))
options: {
  "url": "http://cws.computenext.com/api/instance/e5cff882-8d95-4f8e-9087-531075a0b0e6?refresh=true",
  "method": "get",
  "auth": {
        "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
        "pass": "<hidden>"
  }
}
##### setting current requestId: 26b5c11d-dbd7-498c-9ba9-3683acb620b3
----- RESULT -----
{
  "requestId": "26b5c11d-dbd7-498c-9ba9-3683acb620b3",
  "instanceId": "e5cff882-8d95-4f8e-9087-531075a0b0e6",
  "created": "2014-01-10T00:29:36.854Z",
  "updated": "2014-01-10T00:29:36.854Z",
  "ownerId": "d00f16e6-7503-4838-a658-8b60594ff285",
  "requestStatus": "in-progress",
```

```
  "resourceUri": "image/hpcloud/nova/ami-00058180",
  "resourceType": "image",
  "provider": "hpcloud",
  "region": "nova",
  "connector": "openStack.compute",
  "parameters": {
        "providerInstanceId": "ami-00058180",
        "action": "image.retrieve",
        "instanceId": "e5cff882-8d95-4f8e-9087-531075a0b0e6",
        "image_providerResourceId": "ami-00058180",
        "zone": "nova"
  }
}
```

Get the request:

```
>node runcws.js getr
getr (retrieve request)
options: {
  "url": "http://cws.computenext.com/api/request/26b5c11d-dbd7-498c-9ba9-3683acb620b3",
  "method": "get",
  "auth": {
        "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
        "pass": "<hidden>"
  }
}
----- RESULT -----
[
  {
        "requestId": "26b5c11d-dbd7-498c-9ba9-3683acb620b3",
        "instanceId": "e5cff882-8d95-4f8e-9087-531075a0b0e6",
        "created": "2014-01-10T00:29:36.854Z",
        "updated": "2014-01-10T00:29:37.315Z",
        "ownerId": "d00f16e6-7503-4838-a658-8b60594ff285",
        "requestStatus": "completed",
        "resourceUri": "image/hpcloud/nova/ami-00058180",
        "resourceType": "image",
        "provider": "hpcloud",
        "region": "nova",
        "connector": "openStack.compute",
        "parameters": {
          "providerInstanceId": "ami-00058180",
          "action": "image.retrieve",
          "instanceId": "e5cff882-8d95-4f8e-9087-531075a0b0e6",
          "image_providerResourceId": "ami-00058180",
          "zone": "nova"
        },
        "results": {
          "providerInstanceId": "360832",
          "progress": 100,
          "instanceStatus": "created"
        }
  }
]
```

The requestStatus is *completed* and the instanceStatus (of the image) is now *created*.

Get the *image* instance:

```
>node runcws.js geti
geti (retrieve instance)
options: {
  "url": "http://cws.computenext.com/api/instance/e5cff882-8d95-4f8e-9087-531075a0b0e6",
  "method": "get",
  "auth": {
        "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
        "pass": "<hidden>"
  }
}
----- RESULT -----
{
  "instanceId": "e5cff882-8d95-4f8e-9087-531075a0b0e6",
  "created": "2014-01-10T00:27:33.296Z",
  "updated": "2014-01-10T00:29:37.306Z",
  "ownerId": "d00f16e6-7503-4838-a658-8b60594ff285",
  "resourceUri": "image/hpcloud/nova/ami-00058180",
  "resourceType": "image",
  "provider": "hpcloud",
  "region": "nova",
  "providerResourceId": "ami-00058180",
  "attributes": {
        "providerInstanceId": "ami-00058180",
        "instanceStatus": "created",
        "transientStatus": false
  },
  "attributeTimestamps": {
        "instanceStatus": "2014-01-10T00:29:37.282Z"
  },
  "metadata": {
        "name": "IMAGE-1",
        "description": "my first image"
  },
  "parameters": {
        "providerInstanceId": 2818403,
        "vm_providerResourceId": "Standard.small",
        "zone": "nova"
  }
}
```

The image instanceStatus is now *created*.

You can now use this image resourceUri to create new vm instances.

Deletion of a private image instance is similar to any other instance type.

CHAPTER 10

Getting Started with Workloads

This section will explain how to get started with the ComputeNext workload API.

For this tutorial we will use a simple command line interface tool from ComputeNext called *runcws*.

runcws is a JavaScript program for Node.js.

(CWS = ComputeNext Web Services)

It is basically a simple layer over the REST API, and it keeps track of the various ID's for you to make things easier.

It makes one REST call to the ComputeNext REST API and returns the results.

You could also use any other HTTP or REST test tool such as *curl*

NOTE: This tutorial assumes you are familiar with the basic concepts of the ComputeNext API, including instances and workloads.

The workloadAPI consists of a bunch of standard CRUD methods (Create/Retrieve/Update/Delete) plus some other methods used to control the planning, activation and deactivation of resources in the workload.

Tutorial Notes

The examples in this tutorial were done with an early version of the runcws tool and used HTTP (not HTTPS).

It is **strongly recommended** that you use **HTTPS** for all interactions with the ComputeNext API because Basic authentication is used.

If HTTPS is not used, your credentials will be in clear text "on the wire".

Some examples in this tutorial were done at different times, so sometimes the timestamps etc. may not exactly match between examples.

To execute the workload plan you must first enter your payment information into the ComputeNext website. If you do not have payment information you will receive a "403 Forbidden" error.

The examples were run on a Windows platform, things might be slightly different on Linux.

10.1 Download and Install runcws

runcws is a Node.js script so first you will need to install Node.js from here: http://nodejs.org/download/

Any current stable build should work.

runcws is installed using NPM:

```
C:\>npm install runcws
npm http GET https://registry.npmjs.org/runcws
npm http 304 https://registry.npmjs.org/runcws
runcws@1.0.3 runcws

C:\>cd runcws

C:\runcws>dir
 Volume in drive C has no label.
 Volume Serial Number is A000-0000

 Directory of C:\runcws

01/06/2014  04:31 PM    <DIR>          .
01/06/2014  04:31 PM    <DIR>          ..
01/06/2014  04:31 PM               333 current.json
01/06/2014  04:31 PM    <DIR>          demo
01/06/2014  04:31 PM            18,365 LICENSE
01/06/2014  04:31 PM             1,184 package.json
01/06/2014  04:31 PM               168 README.md
01/06/2014  04:31 PM            10,991 runcws.js
01/06/2014  04:31 PM             6,726 runcws.json
01/06/2014  04:31 PM    <DIR>          workloads
               6 File(s)         37,767 bytes
               4 Dir(s)  583,675,293,696 bytes free
```

You will need to run *node runcws.js* from this directory.

The *current.json* file contains current settings for the runcws script and needs to be writable.

The *demo* directory contains sample JSON files for the instance API tutorial.

The *workloads* directory contains sample JSON workload files for the workload API tutorial.

Get a list of the possible commands/methods that runcws provides by entering at the command line:

```
>node runcws.js
SYNTAX: node runcws <command>
EXAMPLE: node runcws show
----- commands (resource) -----
metadata (retrieve resource metadata)
query (query resources)
region (retrieve region details)
restrictions (retrieve image restrictions)
capabilities (retrieve resource capabilities)
resource (retrieve resource details)
action (retrieve resource actions)
validate (validate resource)
----- commands (instance) -----
createi (create instance from resource)
getr (retrieve request)
listr (retrieve multiple requests)
createimage (create an image instance from a VM instance)
geti (retrieve instance)
getir (retrieve instance (with refresh))
listi (retrieve multiple instances)
listiwl (retrieve all instances in the workload)
listitx (retrieve all instances in the transaction)
updatei (update instance)
deletei (delete instance)
```

```
----- commands (workload) -----
createwl (create workload)
clonewl (clone workload)
getwl (retrieve workload)
listwl (retrieve multiple workloads)
updatewl (update workload)
deletewl (delete workload)
updateel (create or update a workload element)
deleteel (delete workload element)
activate (plan workload activation)
deactivate (plan workload deactivation)
execute (execute workload plan)
steps (retrieve transaction steps)
errors (retrieve transaction errors)
status (retrieve transaction status)
cancel (cancel transaction)
----- commands (misc) -----
setr (set current requestId)
seti (set current instanceId)
setwl (set current workloadId)
settx (set current transactionId)
show (show current settings)
```

Note the first set of commands are used to query resources. The query command itself has its own set of options shown as follows when one enters the command:

```
>node runcws.js query
query (query resources)
ERROR: missing parameter: resource type (image | instanceType | virtualMachine | volumeStorage | soft
```

Entering the following command will query the virtual machines producing the same output as using the http GET method:

```
>node runcws.js query virtualMachine
```

10.2 Set Up runcws

Before you start with runcws, you will need to obtain your API keys for your ComputeNext user account.

Instructions for doing this are here: *Obtaining API Keys*

As part of the runcws installation there is a JSON file named *current.json*

Open this with a text editor and update the *apikey* and *apisec* properties.

10.3 List (Retrieve Multiple) Workloads

First, we will try and list your workloads. If this succeeds, you will know that you are communicating OK with the ComputeNext REST API endpoint.

All input and output to/from the API is in JSON format.

If you have no workloads this method will return an empty JSON array:

```
>node runcws.js listwl
listwl (retrieve multiple workloads)
options: {
  "url": "http://cws.computenext.com/api/workload",
  "method": "get",
  "auth": {
    "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
    "pass": "<hidden>"
  }
}
----- RESULT -----
[]
```

If you already have one or more workloads defined, your output may look something like this:

```
>node runcws.js listwl
listwl (retrieve multiple workloads)
options: {
  "url": "http://cws.computenext.com/api/workload",
  "method": "get",
  "auth": {
    "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
    "pass": "<hidden>"
  }
}
----- RESULT -----
[
  {
    "workloadId": "34452cb1-7523-4157-b70b-09a326721aa6",
    "ownerId": "d00f16e6-7503-4838-a658-8b60594ff285",
    "uniqueName": "wl-26820482",
    "name": "WL-26820482",
    "description": "WL-26820482",
    "created": "2013-12-17T18:07:06.549Z",
    "updated": "2013-12-17T18:07:21.523Z",
    "workloadStatus": "none",
    "hasPlan": false,
    "hasExecute": false
  },
  {
    "workloadId": "235e4fdf-b39c-40a9-aa4c-177965414e51",
    "ownerId": "d00f16e6-7503-4838-a658-8b60594ff285",
    "uniqueName": "wl-d2940014",
    "name": "WL-D2940014",
    "description": "WL-D2940014",
    "created": "2013-12-17T18:07:28.667Z",
    "workloadStatus": "none",
    "hasPlan": false,
    "hasExecute": false
  }
]
```

In this case there are two workloads. In this call, only the "header" information for the workloads are returned, not the full workload JSON.

The *workloadStatus* property can have the values "none" or "in-progress". "in-progress" means that some activity is in progress, such as planning, activation or deactivation (see later).

"hasPlan" means that this workload has a workload *plan*. See later.

"hasExecute" means that this workload has an *execute* section. See later.

10.4 Create Workload

Once we have established that we are communicating OK with the ComputeNext REST API endpoint, we can try to create our first workload.

There are some sample workloads in the runcws installation directory under "workloads".

For a description of the workload schema see here: *Workload Schema* and here: *Workload Element Schema*

For a description of the parameters required for the various resource types see here: *Resource Types, Actions and Parameters*

To create our first workload:

```
>node runcws.js createwl workloads\hello_vm.json
createwl (create workload)
options: {
  "url": "http://cws.computenext.com/api/workload",
  "method": "post",
  "json": {
    "name": "Hello VM",
    "description": "Workload 'hello world' for one VM",
    "metadata": {
      "test": "this is metadata for the entire workload - can be anything",
      "test1": "another line of metadata"
    },
    "elements": [
      {
        "name": "VM 1",
        "uri": "vm/hpcloud/nova/standard.small",
        "parameters": {
          "imageUri": "image/hpcloud/nova/ami-00000075",
          "keyPair": "KP 1",
          "securityGroups": [
            "SG 1",
            "SG 2"
          ]
        },
        "metadata": {
          "description": "hello world - my first virtual machine"
        }
      }
    ]
  },
  "auth": {
    "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
    "pass": "<hidden>"
  }
}
### setting current workloadId: b4ee62da-8dff-4a6b-b39f-54acf26a3a6d
----- RESULT -----
{
  "workloadId": "b4ee62da-8dff-4a6b-b39f-54acf26a3a6d",
  "ownerId": "d00f16e6-7503-4838-a658-8b60594ff285",
  "uniqueName": "hellovm",
  "name": "Hello VM",
```

```
  "description": "Workload 'hello world' for one VM",
  "created": "2013-12-19T01:23:27.354Z",
  "metadata": {
    "test": "this is metadata for the entire workload - can be anything",
    "test1": "another line of metadata"
  },
  "elements": [
    {
      "name": "VM 1",
      "uri": "vm/hpcloud/nova/standard.small",
      "parameters": {
        "imageUri": "image/hpcloud/nova/ami-00000075",
        "keyPair": "KP 1",
        "securityGroups": [
          "SG 1",
          "SG 2"
        ]
      },
      "metadata": {
        "description": "hello world - my first virtual machine"
      }
    }
  ]
}
```


The workload JSON is echoed back in the result from the API call so we can check that everything is correct.

10.5 Retrieve Workload

We can now retrieve the workload we just created:


```
>node runcws.js getwl
getwl (retrieve workload)
options: {
  "url": "http://cws.computenext.com/api/workload/b4ee62da-8dff-4a6b-b39f-54acf26a3a6d",
  "method": "get",
  "auth": {
    "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
    "pass": "<hidden>"
  }
}
----- RESULT -----
{
  "workloadId": "b4ee62da-8dff-4a6b-b39f-54acf26a3a6d",
  "ownerId": "d00f16e6-7503-4838-a658-8b60594ff285",
  "uniqueName": "hellovm",
  "name": "Hello VM",
  "description": "Workload 'hello world' for one VM",
  "created": "2013-12-19T01:23:27.354Z",
  "metadata": {
    "test": "this is metadata for the entire workload - can be anything",
    "test1": "another line of metadata"
  },
  "elements": [
    {
      "name": "VM 1",
```

```
      "uri": "vm/hpcloud/nova/standard.small",
      "parameters": {
        "imageUri": "image/hpcloud/nova/ami-00000075",
        "keyPair": "KP 1",
        "securityGroups": [
          "SG 1",
          "SG 2"
        ]
      },
      "metadata": {
        "description": "hello world - my first virtual machine"
      }
    }
  ],
  "workloadStatus": "none"
}
```

10.6 Clone Workload

Cloning a workload allows us to create another workload that is exactly the same as the original workload except for its name - because the workload name must be unique.

Clone the workload:

```
>node runcws.js clonewl MyFirstClone
clonewl (clone workload)
options: {
  "url": "http://cws.computenext.com/api/workload/clone/b4ee62da-8dff-4a6b-b39f-54acf26a3a6d",
  "method": "post",
  "json": {
    "name": "MyFirstClone"
  },
  "auth": {
    "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
    "pass": "<hidden>"
  }
}
### setting current workloadId: 9c3bfffc-0d12-4ebb-b897-83b8ee567065
----- RESULT -----
{
  "workloadId": "9c3bfffc-0d12-4ebb-b897-83b8ee567065",
  "ownerId": "d00f16e6-7503-4838-a658-8b60594ff285",
  "uniqueName": "myfirstclone",
  "name": "MyFirstClone",
  "description": "Workload 'hello world' for one VM",
  "created": "2013-12-19T01:38:31.713Z",
  "metadata": {
    "test": "this is metadata for the entire workload - can be anything",
    "test1": "another line of metadata"
  },
  "elements": [
    {
      "name": "VM 1",
      "uri": "vm/hpcloud/nova/standard.small",
      "parameters": {
        "imageUri": "image/hpcloud/nova/ami-00000075",
```

```
        "keyPair": "KP 1",
        "securityGroups": [
          "SG 1",
          "SG 2"
        ]
      },
      "metadata": {
        "description": "hello world - my first virtual machine"
      }
    }
  ]
}
```

If we now list the workloads, we will see both the original workload (name = Hello VM) and the new "cloned" workload (name = MyFirstClone)

10.7 Update Workload

To demonstrate updating a workload, we will overwrite our cloned workload (MyFirstClone) with a completely different workload JSON (Hello VS):

```
>node runcws.js updatewl workloads\hello_vs.json
updatewl (update workload)
options: {
  "url": "http://cws.computenext.com/api/workload/9c3bfffc-0d12-4ebb-b897-83b8ee567065",
  "method": "put",
  "json": {
    "name": "Hello VS",
    "description": "Workload 'hello world' for one VS",
    "metadata": {
      "test": "this is metadata for the entire workload - can be anything",
      "test1": "another line of metadata"
    },
    "elements": [
      {
        "name": "VS 1",
        "uri": "vs/hpcloud/nova/standard.10",
        "parameters": {
          "sizeInGBytes": 1
        },
        "metadata": {
          "description": "hello world - my first volume storage"
        }
      }
    ]
  },
  "auth": {
    "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
    "pass": "<hidden>"
  }
}
----- RESULT -----
{
  "ownerId": "d00f16e6-7503-4838-a658-8b60594ff285",
  "uniqueName": "hellovs",
  "name": "Hello VS",
```

```
  "description": "Workload 'hello world' for one VS",
  "updated": "2013-12-19T01:42:26.517Z",
  "metadata": {
    "test": "this is metadata for the entire workload - can be anything",
    "test1": "another line of metadata"
  },
  "elements": [
    {
      "name": "VS 1",
      "uri": "vs/hpcloud/nova/standard.10",
      "parameters": {
        "sizeInGBytes": 1
      },
      "metadata": {
        "description": "hello world - my first volume storage"
      }
    }
  ]
}
```

Get the workload:

```
>node runcws.js getwl
getwl (retrieve workload)
options: {
  "url": "http://cws.computenext.com/api/workload/9c3bfffc-0d12-4ebb-b897-83b8ee567065",
  "method": "get",
  "auth": {
    "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
    "pass": "<hidden>"
  }
}
----- RESULT -----
{
  "workloadId": "9c3bfffc-0d12-4ebb-b897-83b8ee567065",
  "ownerId": "d00f16e6-7503-4838-a658-8b60594ff285",
  "uniqueName": "hellovs",
  "name": "Hello VS",
  "description": "Workload 'hello world' for one VS",
  "created": "2013-12-19T01:38:31.713Z",
  "updated": "2013-12-19T01:42:26.517Z",
  "metadata": {
    "test": "this is metadata for the entire workload - can be anything",
    "test1": "another line of metadata"
  },
  "elements": [
    {
      "name": "VS 1",
      "uri": "vs/hpcloud/nova/standard.10",
      "parameters": {
        "sizeInGBytes": 1
      },
      "metadata": {
        "description": "hello world - my first volume storage"
      }
    }
  ],
  "workloadStatus": "none"
}
```

Note that the workloadId has not changed, so we updated the name, description, metadata and elements properties of the original cloned workload.

10.8 Update Workload Element

Now, let's add one new workload element (VM 2) to this workload:

```
>node runcws.js updateel workloads\vm_element.json
updateel (create or update a workload element)
options: {
  "url": "http://cws.computenext.com/api/workload/9c3bfffc-0d12-4ebb-b897-83b8ee567065/element",
  "method": "put",
  "json": {
    "name": "VM 2",
    "uri": "vm/hpcloud/nova/standard.small",
    "parameters": {
      "imageUri": "image/hpcloud/nova/ami-00000075",
      "keyPair": "KP 1",
      "securityGroups": [
        "SG 1",
        "SG 2"
      ]
    },
    "metadata": {
      "description": "hello world - my SECOND virtual machine"
    }
  },
  "auth": {
    "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
    "pass": "<hidden>"
  }
}
----- RESULT -----
{
  "name": "VM 2",
  "uri": "vm/hpcloud/nova/standard.small",
  "parameters": {
    "imageUri": "image/hpcloud/nova/ami-00000075",
    "keyPair": "KP 1",
    "securityGroups": [
      "SG 1",
      "SG 2"
    ]
  },
  "metadata": {
    "description": "hello world - my SECOND virtual machine"
  }
}
```

Get the workload:

```
>node runcws.js getwl
getwl (retrieve workload)
options: {
  "url": "http://cws.computenext.com/api/workload/9c3bfffc-0d12-4ebb-b897-83b8ee567065",
```

```
  "method": "get",
  "auth": {
    "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
    "pass": "<hidden>"
  }
}
----- RESULT -----
{
  "workloadId": "9c3bfffc-0d12-4ebb-b897-83b8ee567065",
  "ownerId": "d00f16e6-7503-4838-a658-8b60594ff285",
  "uniqueName": "hellovs",
  "name": "Hello VS",
  "description": "Workload 'hello world' for one VS",
  "created": "2013-12-19T01:38:31.713Z",
  "updated": "2013-12-19T01:47:06.516Z",
  "metadata": {
    "test": "this is metadata for the entire workload - can be anything",
    "test1": "another line of metadata"
  },
  "elements": [
    {
      "name": "VM 2",
      "uri": "vm/hpcloud/nova/standard.small",
      "parameters": {
        "imageUri": "image/hpcloud/nova/ami-00000075",
        "keyPair": "KP 1",
        "securityGroups": [
          "SG 1",
          "SG 2"
        ]
      },
      "metadata": {
        "description": "hello world - my SECOND virtual machine"
      }
    },
    {
      "name": "VS 1",
      "uri": "vs/hpcloud/nova/standard.10",
      "parameters": {
        "sizeInGBytes": 1
      },
      "metadata": {
        "description": "hello world - my first volume storage"
      }
    }
  ],
  "workloadStatus": "none"
}
```


You can see that there was originally only the “VS 1” workload element, now the “VM 2” workload element has been added.

10.9 Delete Workload Element

To delete a workload element, the JSON that is sent must contain (at least) the workload element name.

Workload elements are identified by name only, and the workload element name must be unique in the workload.

Delete one element (VM 2):

```
>node runcws.js deleteel workloads\vm_element.json
deleteel (delete workload element)
options: {
  "url": "http://cws.computenext.com/api/workload/9c3bfffc-0d12-4ebb-b897-83b8ee567065/element",
  "method": "delete",
  "json": {
    "name": "VM 2",
    "uri": "vm/hpcloud/nova/standard.small",
    "parameters": {
      "imageUri": "image/hpcloud/nova/ami-00000075",
      "keyPair": "KP 1",
      "securityGroups": [
        "SG 1",
        "SG 2"
      ]
    },
    "metadata": {
      "description": "hello world - my SECOND virtual machine"
    }
  },
  "auth": {
    "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
    "pass": "<hidden>"
  }
}
----- RESULT -----
{
  "name": "VM 2",
  "uri": "vm/hpcloud/nova/standard.small",
  "parameters": {
    "imageUri": "image/hpcloud/nova/ami-00000075",
    "keyPair": "KP 1",
    "securityGroups": [
      "SG 1",
      "SG 2"
    ]
  },
  "metadata": {
    "description": "hello world - my SECOND virtual machine"
  }
}
```

Get the workload:

```
>node runcws.js getwl
getwl (retrieve workload)
options: {
  "url": "http://cws.computenext.com/api/workload/9c3bfffc-0d12-4ebb-b897-83b8ee567065",
  "method": "get",
  "auth": {
    "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
    "pass": "<hidden>"
  }
}
----- RESULT -----
{
  "workloadId": "9c3bfffc-0d12-4ebb-b897-83b8ee567065",
```

```
  "ownerId": "d00f16e6-7503-4838-a658-8b60594ff285",
  "uniqueName": "hellovs",
  "name": "Hello VS",
  "description": "Workload 'hello world' for one VS",
  "created": "2013-12-19T01:38:31.713Z",
  "updated": "2013-12-19T01:57:58.944Z",
  "metadata": {
    "test": "this is metadata for the entire workload - can be anything",
    "test1": "another line of metadata"
  },
  "elements": [
    {
      "name": "VS 1",
      "uri": "vs/hpcloud/nova/standard.10",
      "parameters": {
        "sizeInGBytes": 1
      },
      "metadata": {
        "description": "hello world - my first volume storage"
      }
    }
  ],
  "workloadStatus": "none"
}
```


You can see the "VM 2" workload element has now been removed from the workload.

10.10 Delete Workload

To delete the workload:

```
>node runcws.js deletewl
deletewl (delete workload)
options: {
  "url": "http://cws.computenext.com/api/workload/9c3bfffc-0d12-4ebb-b897-83b8ee567065",
  "method": "delete",
  "auth": {
    "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
    "pass": "<hidden>"
  }
}
----- RESULT -----
{
  "ownerId": "d00f16e6-7503-4838-a658-8b60594ff285",
  "workloadId": "9c3bfffc-0d12-4ebb-b897-83b8ee567065",
  "deleted": true
}
```

The "deleted" property tells us whether the workload was actually deleted - or whether it was just not actually there:

```
>node runcws.js deletewl
deletewl (delete workload)
options: {
  "url": "http://cws.computenext.com/api/workload/9c3bfffc-0d12-4ebb-b897-83b8ee567065",
  "method": "delete",
  "auth": {
    "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
```

```
    "pass": "<hidden>"
  }
}
----- RESULT -----
{
  "ownerId": "d00f16e6-7503-4838-a658-8b60594ff285",
  "workloadId": "9c3bfffc-0d12-4ebb-b897-83b8ee567065",
  "deleted": false
}
```

10.11 Plan Workload Activation

We have now deleted the workload that we cloned, so we need to set our workloadId used by runcws back to the original workload (Hello VM):

```
>node runcws.js setwl b4ee62da-8dff-4a6b-b39f-54acf26a3a6d
setwl (set current workloadId)
----- current settings -----
requestId: 190fe3a8-4d97-4de3-b6a9-c00ae7a4e1ec
instanceId: 3e9b4f1a-3e1e-4784-941a-feab27974b45
workloadId: b4ee62da-8dff-4a6b-b39f-54acf26a3a6d
transactionId: 8b3e4597-ab54-4b37-bcf0-3ebbbf1238f3
```

BTW, we can see the current settings used by runcws using the *show* command:

```
>node runcws.js show
show (show current settings)
----- current settings -----
requestId: 190fe3a8-4d97-4de3-b6a9-c00ae7a4e1ec
instanceId: 3e9b4f1a-3e1e-4784-941a-feab27974b45
workloadId: b4ee62da-8dff-4a6b-b39f-54acf26a3a6d
transactionId: 8b3e4597-ab54-4b37-bcf0-3ebbbf1238f3
```

We are now going to plan the activation of the workload.

In the workload API, activation (or deactivation) of a workload is a two phase process. First, the workload is planned, and then the plan is executed. This two phase process allows you to check what will be done before it is actually done, so you can verify that you are getting what you expected. If something fails, it also makes it easier to understand exactly what failed and why.

To plan activation:

```
>node runcws.js activate
activate (plan workload activation)
options: {
  "url": "http://cws.computenext.com/api/workload/b4ee62da-8dff-4a6b-b39f-54acf26a3a6d/plan?action=ac
  "method": "put",
  "auth": {
    "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
    "pass": "<hidden>"
  }
}
----- RESULT -----
{
  "workloadStatus": "in-progress",
  "action": "plan-activate"
}
```

The workload is now in the process of planning. For a simple workload such as this it should be quite quick. For larger and more complex workloads this can take some time.

Get the workload:

```
>node runcws.js getwl
getwl (retrieve workload)
options: {
  "url": "http://cws.computenext.com/api/workload/b4ee62da-8dff-4a6b-b39f-54acf26a3a6d",
  "method": "get",
  "auth": {
    "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
    "pass": "<hidden>"
  }
}
----- RESULT -----
{
  "workloadId": "b4ee62da-8dff-4a6b-b39f-54acf26a3a6d",
  "ownerId": "d00f16e6-7503-4838-a658-8b60594ff285",
  "uniqueName": "hellovm",
  "name": "Hello VM",
  "description": "Workload 'hello world' for one VM",
  "created": "2013-12-19T01:23:27.354Z",
  "metadata": {
    "test": "this is metadata for the entire workload - can be anything",
    "test1": "another line of metadata"
  },
  "elements": [
    {
      "name": "VM 1",
      "uri": "vm/hpcloud/nova/standard.small",
      "parameters": {
        "imageUri": "image/hpcloud/nova/ami-00000075",
        "keyPair": "KP 1",
        "securityGroups": [
          "SG 1",
          "SG 2"
        ]
      },
      "metadata": {
        "description": "hello world - my first virtual machine"
      }
    }
  ],
  "workloadStatus": "none",
  "plan": {
    "action": "activate",
    "ownerId": "d00f16e6-7503-4838-a658-8b60594ff285",
    "created": "2013-12-19T02:07:54.180Z",
    "expires": "2013-12-19T02:12:54.180Z",
    "elements": [
      {
        "name": "VM 1",
        "uri": "vm/hpcloud/nova/standard.small",
        "parameters": {
          "imageUri": "image/hpcloud/nova/ami-00000075",
          "keyPairId": "*0000_kp_create_kp1",
          "securityGroupIds": [
            "*0001_sg_create_sg1",
```

```
            "*0003_sg_create_sg2"
          ]
        },
        "metadata": {
          "description": "hello world - my first virtual machine",
          "name": "VM 1"
        },
        "resource": {
          "id": "vm_hpcloud_nova_standard-small",
          "uri": "vm/hpcloud/nova/standard.small",
          "resourceType": "vm",
          "provider": "hpcloud",
          "region": "nova",
          "providerResourceId": "Standard.small",
          "cpuSpeed": "1.2",
          "cpuCount": "2",
          "localStorage": "60",
          "ram": "2",
          "operatingSystemVersion": "64 Bit",
          "zone": "nova",
          "connectorType": "openStack.compute"
        }
      }
    ],
    "serial": [
      {
        "parallel": [
          {
            "step": {
              "id": "0000_kp_create_kp1",
              "action": "kp.create",
              "uri": "kp/hpcloud/nova/standard",
              "metadata": {
                "name": "KP 1"
              },
              "sourceElement": "VM 1",
              "resource": {
                "provider": "hpcloud",
                "region": "nova",
                "resourceType": "kp"
              },
              "timing": {
                "min": 1.09,
                "avg": 1.82,
                "max": 2.11
              }
            }
          }
        ]
      },
      {
        "parallel": [
          {
            "step": {
              "id": "0001_sg_create_sg1",
              "action": "sg.create",
              "uri": "sg/hpcloud/nova/standard",
              "metadata": {
```

```
            "name": "SG 1"
          },
          "sourceElement": "VM 1",
          "resource": {
            "provider": "hpcloud",
            "region": "nova",
            "resourceType": "sg"
          },
          "timing": {
            "min": 1.03,
            "avg": 1.04,
            "max": 1.08
          }
        }
      },
      {
        "step": {
          "id": "0003_sg_create_sg2",
          "action": "sg.create",
          "uri": "sg/hpcloud/nova/standard",
          "metadata": {
            "name": "SG 2"
          },
          "sourceElement": "VM 1",
          "resource": {
            "provider": "hpcloud",
            "region": "nova",
            "resourceType": "sg"
          },
          "timing": {
            "min": 1.03,
            "avg": 1.04,
            "max": 1.08
          }
        }
      }
    ]
  },
  {
    "parallel": [
      {
        "step": {
          "id": "0002_sg_update_add_access_sg1",
          "action": "sg.update.add-access",
          "instanceId": "*0001_sg_create_sg1",
          "parameters": {
            "rules": [
              {
                "protocol": "tcp",
                "from-port": 22,
                "to-port": 22
              },
              {
                "protocol": "tcp",
                "from-port": 3389,
                "to-port": 3389
              },
              {
```

```
                "protocol": "icmp",
                "from-port": -1,
                "to-port": -1
              }
            ]
          },
          "sourceElement": "VM 1",
          "timing": {
            "min": 1.06,
            "avg": 1.4,
            "max": 3.1
          }
        }
      },
      {
        "step": {
          "id": "0004_sg_update_add_access_sg2",
          "action": "sg.update.add-access",
          "instanceId": "*0003_sg_create_sg2",
          "parameters": {
            "rules": [
              {
                "protocol": "tcp",
                "from-port": 22,
                "to-port": 22
              },
              {
                "protocol": "tcp",
                "from-port": 3389,
                "to-port": 3389
              },
              {
                "protocol": "icmp",
                "from-port": -1,
                "to-port": -1
              }
            ]
          },
          "sourceElement": "VM 1",
          "timing": {
            "min": 1.06,
            "avg": 1.4,
            "max": 3.1
          }
        }
      }
    ]
  },
  {
    "parallel": [
      {
        "step": {
          "id": "0005_vm_create_vm1",
          "action": "vm.create",
          "uri": "vm/hpcloud/nova/standard.small",
          "parameters": {
            "imageUri": "image/hpcloud/nova/ami-00000075",
            "keyPairId": "*0000_kp_create_kp1",
```

```
                "securityGroupIds": [
                  "*0001_sg_create_sg1",
                  "*0003_sg_create_sg2"
                ]
              },
              "metadata": {
                "description": "hello world - my first virtual machine",
                "name": "VM 1"
              },
              "sourceElement": "VM 1",
              "timing": {
                "min": 90.34,
                "avg": 90.34,
                "max": 90.34
              }
            }
          }
        ]
      }
    ],
    "inventory": {
      "hpcloud": {
        "nova": {
          "quota": {
            "kp": {
              "count": "17.31"
            },
            "sg": {
              "count": "32.00"
            },
            "vm": {
              "count": "4.00",
              "cpuCount": "0.60",
              "localStorage": "1.80",
              "ram": "0.06"
            },
            "vs": {
              "count": "0.00",
              "sizeInGBytes": "0.00"
            }
          }
        }
      }
    },
    "summary": {
      "kp": 1,
      "sg": 2,
      "vm": 1
    }
  }
}
```


You can see that the "plan" section has now been added into the workload JSON and that it contains a lot of new properties.

- plan.action
- plan.expires

- plan.elements
- plan.serial
- plan.inventory
- plan.summary

10.11.1 plan.action

This indicates what kind of plan was requested - "activate" or "deactivate".

10.11.2 plan.expires

The workload plan depends on the current state of the instances, so the plan has an expire time of 5 minutes. When the plan expires, it will be removed from the workload JSON.

10.11.3 plan.elements

This is a snapshot of the workload elements at the time they were used for the plan. At this point you can update the workload and add & delete elements to the *elements* section, so the *plan.elements* section preserves what was used for the plan. The *plan.elements* section within the plan is slightly different to the original *elements* section. It has been processed into a form that is useable by the instance API, because each step will be a call to the instance API. You may see references to other steps which start with an asterisk '*'. A *resource* property has been added to each element which contains a summary of the resource properties.

10.11.4 plan.serial

This is the start of the actual workload plan. The actual workload plan is a set of nested sections, some can be serial (executed one at a time) and some can be parallel (executed concurrently). It always starts with *plan.serial*. The serial/parallel sections contain workload steps, which are the steps required to provision the workload elements. See below for the properties of a workload step.

10.11.5 plan.inventory

The plan has a *plan.inventory* property which gives a percentage indication of how much of a resource quota will be used at this region once the workload is provisoned. For example, "inventory.hpcloud.nova.quota.kp.count" is 17.31, which means that 17.31 percent of the quota for key pairs will be reached for the hpcloud.nova region once this workload has been successfully provisioned.

Note that the quotas are not enforced. You may have a quota over 100 percent at some region and you will still be allowed to execute the workload plan - but it is likely that it will fail.

User limits are similar to quotas, but these are the limits on how many resource instances of a given type can be created by the user. These limits are enforced, and if you exceed these limits then there will be an *error* section added to the workload JSON with an explanation of the error.

10.11.6 plan.summary

This property summarizes how many resources of each resource type will be created when this plan is executed.

10.11.7 Workload Steps

A workload step can have similar properties to the workload element from which it was generated. However - one workload element can generate multiple workload steps.

- step.id - this consists of a 4 digit number plus some identifying information about the step - it uniquely identifies the step within the plan.
- step.action - this specifies the instance API action that will be performed for the step.
- step.sourceElement - this references the original workload element that caused this step to be generated.
- step.timing - this has the minimum, average and maximum time (in seconds) expected for this step.

10.11.8 How the Plan is Generated

The workload plan is generated as follows -

1. Key pairs (kp) are created. If you specify a kp by name in the workload element, the workload planner checks to see whether a kp instance with that name (in the instance metadata) already exists in the region. If it does, it uses it. If not, it adds a step to create the kp.
2. Security groups (sg) are created. If you specify an sg by name in the workload element, the workload planner checks to see whether that sg exists. If not, it adds a step to create the sg. It also adds a step to add SSH, RDP and "ping" ports to that sg.
3. Virtual machines (vm) are created.
4. Floating IP addresses (ip) are created - if the vm requires them. A step is added to create the ip, and then another step is added to associate the ip with the vm.
5. Volume storage (vs) is created. A step is added to create the vs, and if required a step is added to attach the vs to the vm.

10.12 Execute Workload

One you have reviewed the workload plan and are happy with it, you can execute the workload plan.

Note: To execute the workload plan you must first enter your payment information into the ComputeNext website. If you do not have payment information you will receive a "403 Forbidden" error.

The process of execution of a workload plan is called a *transaction*.

To execute the workload plan:

```
>node runcws.js execute
execute (execute workload plan)
options: {
  "url": "http://cws.computenext.com/api/workload/b4ee62da-8dff-4a6b-b39f-54acf26a3a6d/execute",
  "method": "put",
  "auth": {
    "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
    "pass": "<hidden>"
  }
}
### setting current transactionId: 06d7ab5f-f420-4fea-b0bf-456fbc14d884
----- RESULT -----
{
  "workloadStatus": "in-progress",
```

```
  "action": "execute",
  "transactionId": "06d7ab5f-f420-4fea-b0bf-456fbc14d884"
}
```

Note that the *transactionId* is returned and that the transaction is "in-progress".

The workload plan is now in the process of executing. This can take some time.

You can monitor the progress of the workload execution with the *transaction* API (part of the *workload* API). See below.

10.12.1 Running Workload

When the workload plan executes, it will call the instance API to create instances of various types - for example, key pairs (kp), security groups (sg), virtual machines (vm), etc.

All instances created by the transaction are tagged with the *transactionId*.

If you want to find out all instances that were created by the transaction you can query the instance API to list instances with this transactionId (runcws listitx).

Virtual machine (vm) and volume storage (vs) instances are tagged with the *workloadId*.

If you want to find out all instances that are part of the "running workload" you can query the instance API to list instances with this workloadId (runcws listiwl).

We only have vm and vs workload elements in the workload, so only vm and vs instances are considered to be part of the running workload.

10.12.2 Partial Failures and Rollback

If something fails during the execution of a transaction, there is no automatic rollback.

For example, assume your workload specifies two virtual machine workload elements, VM1 and VM2.

Let's suppose that during the transaction the creation of one of the virtual machines (VM2) fails for some reason. Because the steps to create virtual machines are (usually) done in parallel, one of the virtual machines is created sucessfully (VM1), but the other (VM2) is not. The transaction will go into "failed" status, and will probably indicate a "reason" of "timeout" or "retries" depending on how the failure manifested itself. The transaction/errors (see below) will contain the details about what step failed and why. But, your "running workload" will still have the virtual machine instance (VM1) that succeeded. (The "running workload" is the collection of instances that are tagged with the workloadId of this workload).

You now have two choices. Either you can attempt the activation again, or you can deactivate. If you know why the create of the virtual machine failed and it is something you can correct, you can update the workload element (for VM2) that failed, and re-activate the workload. This will create a new workload plan. That workload plan will detect that you have already got one virtual machine (VM1) in your "running workload", so it will only generate step(s) to create the one virtual machine (VM2) that failed the first time.

If you want to rollback the creation of the original virtual machine (VM1) then you can deactivate. That will generate a workload plan to delete VM1.

Note that billing is done on the basis of instance activation, and *not* on completion of a successful transaction. If you have a partial activation, you will be charged for those instances that did get created successfully.

10.13 Transaction Status

To get the transaction status:

```
>node runcws.js status
status (retrieve transaction status)
options: {
  "url": "http://cws.computenext.com/api/transaction/06d7ab5f-f420-4fea-b0bf-456fbc14d884/status",
  "method": "get",
  "auth": {
    "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
    "pass": "<hidden>"
  }
}
----- RESULT -----
{
  "transactionId": "06d7ab5f-f420-4fea-b0bf-456fbc14d884",
  "workloadId": "b4ee62da-8dff-4a6b-b39f-54acf26a3a6d",
  "status": "in-progress",
  "started": "2013-12-19T21:51:20.786Z"
}
```

The transaction status can be one of the following values -

- in-progress
- failed

If "failed", there will be a reason code -

- retries
- timeout
- cancelled

The transaction will fail if any one of the workload steps fails.

10.13.1 retries

If a workload step fails, it will attempt one retry. If the step fails again, then the transaction fails with reason code "retries".

10.13.2 timeout

A workload step may timeout waiting for the instance to reach some required state. For example, when creating a virtual machine (vm), the required state is "running". If the instance does not reach the required state within the time allowed the step will fail and hence the transaction will fail with reason code "timeout".

10.13.3 cancelled

If the transaction is cancelled while in-progress (see below) then the transaction will go into failed status with reason code "cancelled".

10.14 Transaction Steps

As the workload plan is executed, a log is kept of the progress. Each log entry has a *Log Sequence Number* (the LSN) which is simply a sequence number of the log record. When you call the workload API to get the transaction steps, you are getting those log entries that show the start and end for each step.

Note that the log for only one transaction (the last one) for each workload is maintained. You cannot go back and look at previous transactions.

To get the transaction steps:

```
>node runcws.js steps
steps (retrieve transaction steps)
options: {
  "url": "http://cws.computenext.com/api/transaction/06d7ab5f-f420-4fea-b0bf-456fbc14d884/steps",
  "method": "get",
  "auth": {
    "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
    "pass": "<hidden>"
  }
}
----- RESULT -----
[
  {
    "lsn": 1,
    "stepId": "0000_kp_create_kp1",
    "timestamp": "2013-12-19T21:51:20.803Z",
    "status": "in-progress"
  },
  {
    "lsn": 8,
    "stepId": "0000_kp_create_kp1",
    "timestamp": "2013-12-19T21:51:22.948Z",
    "status": "completed",
    "elapsedTimeInSeconds": 2.144
  },
  {
    "lsn": 9,
    "stepId": "0001_sg_create_sg1",
    "timestamp": "2013-12-19T21:51:22.961Z",
    "status": "in-progress"
  },
  {
    "lsn": 11,
    "stepId": "0003_sg_create_sg2",
    "timestamp": "2013-12-19T21:51:22.970Z",
    "status": "in-progress"
  },
  {
    "lsn": 18,
    "stepId": "0001_sg_create_sg1",
    "timestamp": "2013-12-19T21:51:24.031Z",
    "status": "completed",
    "elapsedTimeInSeconds": 1.068
  },
  {
    "lsn": 20,
    "stepId": "0003_sg_create_sg2",
```

```
    "timestamp": "2013-12-19T21:51:24.039Z",
    "status": "completed",
    "elapsedTimeInSeconds": 1.067
  },
  {
    "lsn": 21,
    "stepId": "0002_sg_update_add_access_sg1",
    "timestamp": "2013-12-19T21:51:24.060Z",
    "status": "in-progress"
  },
  {
    "lsn": 23,
    "stepId": "0004_sg_update_add_access_sg2",
    "timestamp": "2013-12-19T21:51:24.070Z",
    "status": "in-progress"
  },
  {
    "lsn": 31,
    "stepId": "0002_sg_update_add_access_sg1",
    "timestamp": "2013-12-19T21:51:25.224Z",
    "status": "completed",
    "elapsedTimeInSeconds": 1.163
  },
  {
    "lsn": 34,
    "stepId": "0004_sg_update_add_access_sg2",
    "timestamp": "2013-12-19T21:51:26.201Z",
    "status": "completed",
    "elapsedTimeInSeconds": 2.129
  },
  {
    "lsn": 35,
    "stepId": "0005_vm_create_vm1",
    "timestamp": "2013-12-19T21:51:26.213Z",
    "status": "in-progress"
  }
]
```


The runcws tool gets all the useful log entries for the transaction. For this API method you can also specify a "begin LSN" and an "end LSN" to get just those log records you need.

Each log record can have the following propeties -

- lsn - the Log Sequence Number
- stepId - the workload step id
- timestamp - when this log entry was created
- status - which can be "in-progress", "completed" or "failed"
- reason - if the status is "failed" - can be "retries", "timeout" or "cancelled"
- elapsedTimeInSeconds - if the status is "completed" - this is the time taken for the step

You can see that for stepId "0000_kp_create_kp1" we have two log entries, one with LSN=1 with status "in-progress" and one with LSN=8 with status "completed". The timestamp indicates when the log entry was logged. So this step began at 21:51:20.803, and completed at 21:51:22.948, for an elapsed time of 2.144 seconds.

However - stepId "0005_vm_create_vm1" (LSN=35) has only a log entry for "in-progress" - which means that at the time we did the method call, this step had started but had not yet completed.

10.15 Transaction Errors

To get the transaction errors:

```
>node runcws.js errors
errors (retrieve transaction errors)
options: {
  "url": "http://cws.computenext.com/api/transaction/06d7ab5f-f420-4fea-b0bf-456fbc14d884/errors",
  "method": "get",
  "auth": {
    "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
    "pass": "<hidden>"
  }
}
STATUS: 404
----- RESULT -----
{
  "code": 404,
  "message": "no errors found: transactionId: 06d7ab5f-f420-4fea-b0bf-456fbc14d884",
  "ticket": "f1e8c7dc-13d4-4d42-b999-2574dc6f1c93"
}
```

In this case there are no errors.

If an error is returned, it will include the stepId of the workload step that failed, plus some detailed error information.

Note that it is possible for this call to return an error for a step, but for the overall transaction to succeed. This is because every step attempts a retry on the first error, so the second attempt might succeed.

10.16 Transaction Completes

Get the transaction status:

```
>node runcws.js status
status (retrieve transaction status)
options: {
  "url": "http://cws.computenext.com/api/transaction/06d7ab5f-f420-4fea-b0bf-456fbc14d884/status",
  "method": "get",
  "auth": {
    "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
    "pass": "<hidden>"
  }
}
----- RESULT -----
{
  "transactionId": "06d7ab5f-f420-4fea-b0bf-456fbc14d884",
  "workloadId": "b4ee62da-8dff-4a6b-b39f-54acf26a3a6d",
  "ended": "2013-12-19T21:52:57.516Z",
  "status": "completed",
  "elapsedTimeInSeconds": 96.73
}
```

The transaction status is “completed” and the elapsedTimeInSeconds is given.

Get the steps:

```
>node runcws.js steps
steps (retrieve transaction steps)
options: {
  "url": "http://cws.computenext.com/api/transaction/06d7ab5f-f420-4fea-b0bf-456fbc14d884/steps",
  "method": "get",
  "auth": {
    "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
    "pass": "<hidden>"
  }
}
----- RESULT -----
[
  {
    "lsn": 1,
    "stepId": "0000_kp_create_kp1",
    "timestamp": "2013-12-19T21:51:20.803Z",
    "status": "in-progress"
  },
  {
    "lsn": 8,
    "stepId": "0000_kp_create_kp1",
    "timestamp": "2013-12-19T21:51:22.948Z",
    "status": "completed",
    "elapsedTimeInSeconds": 2.144
  },
  {
    "lsn": 9,
    "stepId": "0001_sg_create_sg1",
    "timestamp": "2013-12-19T21:51:22.961Z",
    "status": "in-progress"
  },
  {
    "lsn": 11,
    "stepId": "0003_sg_create_sg2",
    "timestamp": "2013-12-19T21:51:22.970Z",
    "status": "in-progress"
  },
  {
    "lsn": 18,
    "stepId": "0001_sg_create_sg1",
    "timestamp": "2013-12-19T21:51:24.031Z",
    "status": "completed",
    "elapsedTimeInSeconds": 1.068
  },
  {
    "lsn": 20,
    "stepId": "0003_sg_create_sg2",
    "timestamp": "2013-12-19T21:51:24.039Z",
    "status": "completed",
    "elapsedTimeInSeconds": 1.067
  },
  {
    "lsn": 21,
    "stepId": "0002_sg_update_add_access_sg1",
    "timestamp": "2013-12-19T21:51:24.060Z",
    "status": "in-progress"
  },
  {
```

```
    "lsn": 23,
    "stepId": "0004_sg_update_add_access_sg2",
    "timestamp": "2013-12-19T21:51:24.070Z",
    "status": "in-progress"
  },
  {
    "lsn": 31,
    "stepId": "0002_sg_update_add_access_sg1",
    "timestamp": "2013-12-19T21:51:25.224Z",
    "status": "completed",
    "elapsedTimeInSeconds": 1.163
  },
  {
    "lsn": 34,
    "stepId": "0004_sg_update_add_access_sg2",
    "timestamp": "2013-12-19T21:51:26.201Z",
    "status": "completed",
    "elapsedTimeInSeconds": 2.129
  },
  {
    "lsn": 35,
    "stepId": "0005_vm_create_vm1",
    "timestamp": "2013-12-19T21:51:26.213Z",
    "status": "in-progress"
  },
  {
    "lsn": 48,
    "stepId": "0005_vm_create_vm1",
    "timestamp": "2013-12-19T21:52:56.507Z",
    "status": "completed",
    "elapsedTimeInSeconds": 90.292
  }
]
```


You can see that now all steps have completed, and that VM 1 took about 90 seconds to reach the "running" state.

Get the workload:


```
>node runcws.js getwl
getwl (retrieve workload)
options: {
  "url": "http://cws.computenext.com/api/workload/b4ee62da-8dff-4a6b-b39f-54acf26a3a6d",
  "method": "get",
  "auth": {
    "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
    "pass": "<hidden>"
  }
}
----- RESULT -----
{
  "workloadId": "b4ee62da-8dff-4a6b-b39f-54acf26a3a6d",
  "ownerId": "d00f16e6-7503-4838-a658-8b60594ff285",
  "uniqueName": "hellovm",
  "name": "Hello VM",
  "description": "Workload 'hello world' for one VM",
  "created": "2013-12-19T01:23:27.354Z",
  "updated": "2013-12-19T21:52:57.528Z",
  "metadata": {
    "test": "this is metadata for the entire workload - can be anything",
```

```
    "test1": "another line of metadata"
  },
  "elements": [
    {
      "name": "VM 1",
      "uri": "vm/hpcloud/nova/standard.small",
      "parameters": {
        "imageUri": "image/hpcloud/nova/ami-00000075",
        "keyPair": "KP 1",
        "securityGroups": [
          "SG 1",
          "SG 2"
        ]
      },
      "metadata": {
        "description": "hello world - my first virtual machine"
      }
    }
  ],
  "execute": {
    "created": "2013-12-19T21:52:57.527Z",
    "action": "activate",
    "transactionId": "06d7ab5f-f420-4fea-b0bf-456fbc14d884",
    "elements": [
      {
        "name": "VM 1",
        "uri": "vm/hpcloud/nova/standard.small",
        "parameters": {
          "imageUri": "image/hpcloud/nova/ami-00000075",
          "keyPairId": "*0000_kp_create_kp1",
          "securityGroupIds": [
            "*0001_sg_create_sg1",
            "*0003_sg_create_sg2"
          ]
        },
        "metadata": {
          "description": "hello world - my first virtual machine",
          "name": "VM 1"
        },
        "resource": {
          "id": "vm_hpcloud_nova_standard-small",
          "uri": "vm/hpcloud/nova/standard.small",
          "resourceType": "vm",
          "provider": "hpcloud",
          "region": "nova",
          "providerResourceId": "Standard.small",
          "cpuSpeed": "1.2",
          "cpuCount": "2",
          "localStorage": "60",
          "ram": "2",
          "operatingSystemVersion": "64 Bit",
          "zone": "nova",
          "connectorType": "openStack.compute"
        }
      }
    ]
  },
  "workloadStatus": "none"
```

```
}
```

Note that after the transaction has completed, the plan is no longer in the workload - it is no longer needed. However, if the transaction fails, then the failed plan is included in the *execute* section (see below).

An *execute* section has been added to the workload. This is a snapshot of what the state of the workload was at the last time it was executed.

Get the "running workload" - all the instances that were created for this workload:

```
>node runcws.js listiwl
listiwl (retrieve all instances in the workload)
options: {
  "url": "http://cws.computenext.com/api/instance?workloadId=b4ee62da-8dff-4a6b-b39f-54acf26a3a6d",
  "method": "get",
  "auth": {
    "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
    "pass": "<hidden>"
  }
}
----- RESULT -----
[
  {
    "instanceId": "dd13e743-33be-4b10-b740-99d8cf4f8d71",
    "created": "2013-12-19T21:51:29.681Z",
    "updated": "2013-12-19T21:52:26.831Z",
    "ownerId": "d00f16e6-7503-4838-a658-8b60594ff285",
    "resourceUri": "vm/hpcloud/nova/standard.small",
    "resourceType": "vm",
    "provider": "hpcloud",
    "region": "nova",
    "providerResourceId": "Standard.small",
    "attributes": {
      "providerInstanceId": 2714865,
      "password": "qeh46YnWszk3aRxa",
      "instanceStatus": "running",
      "transientStatus": false,
      "privateIpAddress": "10.2.252.25",
      "publicIpAddress": "15.185.216.105"
    },
    "attributeTimestamps": {
      "password": "2013-12-19T21:52:26.800Z",
      "instanceStatus": "2013-12-19T21:52:26.809Z",
      "privateIpAddress": "2013-12-19T21:52:26.801Z",
      "publicIpAddress": "2013-12-19T21:52:26.808Z"
    },
    "metadata": {
      "description": "hello world - my first virtual machine",
      "name": "VM 1",
      "workloadId": "b4ee62da-8dff-4a6b-b39f-54acf26a3a6d",
      "transactionId": "06d7ab5f-f420-4fea-b0bf-456fbc14d884"
    },
    "parameters": {
      "imageUri": "image/hpcloud/nova/ami-00000075",
      "keyPairId": "60e3e8e9-08aa-4934-90e2-0002de271fdc",
      "securityGroupIds": [
        "9a200d98-31ea-4bbc-a264-b83e5d6219f6",
        "b1092157-703a-4f30-9ed6-895f3277bcf5"
      ],
```

```
      "vm_providerResourceId": "Standard.small",
      "zone": "nova",
      "cpuCount": "2",
      "cpuSpeed": "1.2",
      "localStorage": "60",
      "ram": "2",
      "username": "ubuntu",
      "image_providerResourceId": "ami-00000075",
      "keyPairId_providerInstanceId": "60e3e8e9-08aa-4934-90e2-0002de271fdc",
      "securityGroupIds_providerInstanceId": [
        568643,
        568641
      ]
    }
  }
]
```

10.17 Cancel Transaction

We start with a workload that has two virtual machines defined (VM 1 and VM2):

```
>node runcws.js getwl
getwl (retrieve workload)
options: {
  "url": "http://cws.computenext.com/api/workload/b4ee62da-8dff-4a6b-b39f-54acf26a3a6d",
  "method": "get",
  "auth": {
    "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
    "pass": "<hidden>"
  }
}
----- RESULT -----
{
  "workloadId": "b4ee62da-8dff-4a6b-b39f-54acf26a3a6d",
  "ownerId": "d00f16e6-7503-4838-a658-8b60594ff285",
  "uniqueName": "hellovm",
  "name": "Hello VM",
  "description": "Workload 'hello world' for one VM",
  "created": "2013-12-19T01:23:27.354Z",
  "updated": "2013-12-20T00:04:11.879Z",
  "metadata": {
    "test": "this is metadata for the entire workload - can be anything",
    "test1": "another line of metadata"
  },
  "elements": [
    {
      "name": "VM 1",
      "uri": "vm/hpcloud/nova/standard.small",
      "parameters": {
        "imageUri": "image/hpcloud/nova/ami-00000075",
        "keyPair": "KP 1",
        "securityGroups": [
          "SG 1",
          "SG 2"
        ]
      },
```

```
      "metadata": {
        "description": "hello world - my first virtual machine"
      }
    },
    {
      "name": "VM 2",
      "uri": "vm/hpcloud/nova/standard.small",
      "parameters": {
        "imageUri": "image/hpcloud/nova/ami-00000075",
        "keyPair": "KP 1",
        "securityGroups": [
          "SG 1",
          "SG 2"
        ]
      },
      "metadata": {
        "description": "hello world - my SECOND virtual machine"
      }
    }
  ],
  "execute": {
    "created": "2013-12-20T00:03:12.487Z",
    "action": "deactivate",
    "transactionId": "f1e0ec0c-b871-4b71-be30-03eb05b61191",
    "elements": [
      {
        "name": "VM 1",
        "uri": "vm/hpcloud/nova/standard.small",
        "parameters": {
          "imageUri": "image/hpcloud/nova/ami-00000075",
          "keyPair": "KP 1",
          "securityGroups": [
            "SG 1",
            "SG 2"
          ]
        },
        "metadata": {
          "description": "hello world - my first virtual machine"
        }
      }
    ]
  },
  "workloadStatus": "none"
}
```


Plan activation and then execute:

```
>node runcws.js activate
activate (plan workload activation)
options: {
  "url": "http://cws.computenext.com/api/workload/b4ee62da-8dff-4a6b-b39f-54acf26a3a6d/plan?action=ac
  "method": "put",
  "auth": {
    "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
    "pass": "<hidden>"
  }
}
----- RESULT -----
{
```

```
  "workloadStatus": "in-progress",
  "action": "plan-activate"
}

>node runcws.js execute
execute (execute workload plan)
options: {
  "url": "http://cws.computenext.com/api/workload/b4ee62da-8dff-4a6b-b39f-54acf26a3a6d/execute",
  "method": "put",
  "auth": {
    "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
    "pass": "<hidden>"
  }
}
### setting current transactionId: 654a5e87-04ff-439c-bd51-f08bb033e580
----- RESULT -----
{
  "workloadStatus": "in-progress",
  "action": "execute",
  "transactionId": "654a5e87-04ff-439c-bd51-f08bb033e580"
}
```

Get the workload steps:

```
>node runcws.js steps
steps (retrieve transaction steps)
options: {
  "url": "http://cws.computenext.com/api/transaction/654a5e87-04ff-439c-bd51-f08bb033e580/steps",
  "method": "get",
  "auth": {
    "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
    "pass": "<hidden>"
  }
}
----- RESULT -----
[
  {
    "lsn": 1,
    "stepId": "0000_vm_create_vm1",
    "timestamp": "2013-12-20T00:04:44.651Z",
    "status": "in-progress"
  },
  {
    "lsn": 3,
    "stepId": "0001_vm_create_vm2",
    "timestamp": "2013-12-20T00:04:44.659Z",
    "status": "in-progress"
  }
]
```

The create of both virtual machines is in-progress.

Cancel the transaction:

```
>node runcws.js cancel
cancel (cancel transaction)
options: {
  "url": "http://cws.computenext.com/api/transaction/654a5e87-04ff-439c-bd51-f08bb033e580/cancel",
  "method": "put",
```

```
  "auth": {
    "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
    "pass": "<hidden>"
  }
}
----- RESULT -----
{
  "transactionId": "654a5e87-04ff-439c-bd51-f08bb033e580",
  "workloadId": "b4ee62da-8dff-4a6b-b39f-54acf26a3a6d",
  "action": "cancel",
  "status": "in-progress"
}
```

The cancel operation is in-progress.

If we poll with "steps" eventually we see:

```
>node runcws.js steps
steps (retrieve transaction steps)
options: {
  "url": "http://cws.computenext.com/api/transaction/654a5e87-04ff-439c-bd51-f08bb033e580/steps",
  "method": "get",
  "auth": {
    "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
    "pass": "<hidden>"
  }
}
----- RESULT -----
[
  {
    "lsn": 1,
    "stepId": "0000_vm_create_vm1",
    "timestamp": "2013-12-20T00:04:44.651Z",
    "status": "in-progress"
  },
  {
    "lsn": 3,
    "stepId": "0001_vm_create_vm2",
    "timestamp": "2013-12-20T00:04:44.659Z",
    "status": "in-progress"
  },
  {
    "lsn": 7,
    "stepId": "0001_vm_create_vm2",
    "timestamp": "2013-12-20T00:05:14.802Z",
    "status": "failed",
    "reason": "cancelled",
    "elapsedTimeInSeconds": 30.14
  },
  {
    "lsn": 8,
    "stepId": "0000_vm_create_vm1",
    "timestamp": "2013-12-20T00:05:14.810Z",
    "status": "failed",
    "reason": "cancelled",
    "elapsedTimeInSeconds": 30.157
  }
]
```

Both steps have been cancelled.

If we poll with "status", eventually we see:

```
>node runcws.js status
status (retrieve transaction status)
options: {
  "url": "http://cws.computenext.com/api/transaction/654a5e87-04ff-439c-bd51-f08bb033e580/status",
  "method": "get",
  "auth": {
    "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
    "pass": "<hidden>"
  }
}
----- RESULT -----
{
  "transactionId": "654a5e87-04ff-439c-bd51-f08bb033e580",
  "workloadId": "b4ee62da-8dff-4a6b-b39f-54acf26a3a6d",
  "ended": "2013-12-20T00:05:59.814Z",
  "status": "failed",
  "reason": "cancelled",
  "stepId": "0001_vm_create_vm2",
  "elapsedTimeInSeconds": 75.174
}
```

Note that the transaction status may take some time to reach "cancelled" after the steps have reached "cancelled".

The first step that detected that the workload was cancelled is identified.

However: even though we cancelled the transaction, the requests to create the virtual machines had already been sent to the region. So the virtual machine instances are actually created.

Get the "running workload":

```
>node runcws.js listiwl
listiwl (retrieve all instances in the workload)
options: {
  "url": "http://cws.computenext.com/api/instance?workloadId=b4ee62da-8dff-4a6b-b39f-54acf26a3a6d",
  "method": "get",
  "auth": {
    "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
    "pass": "<hidden>"
  }
}
----- RESULT -----
[
  {
    "instanceId": "0523e9cf-0fe9-4b8d-91e2-21aeca5a1ae3",
    "created": "2013-12-20T00:04:53.328Z",
    "updated": "2013-12-20T00:04:53.328Z",
    "ownerId": "d00f16e6-7503-4838-a658-8b60594ff285",
    "resourceUri": "vm/hpcloud/nova/standard.small",
    "resourceType": "vm",
    "provider": "hpcloud",
    "region": "nova",
    "attributes": {
      "providerInstanceId": 2715437,
      "password": "AFdTwcwmFQc5DyAx",
      "instanceStatus": "creating",
      "transientStatus": true
    },
```

```
    "attributeTimestamps": {
      "password": "2013-12-20T00:04:53.318Z",
      "instanceStatus": "2013-12-20T00:04:53.325Z"
    },
    "metadata": {
      "description": "hello world - my first virtual machine",
      "name": "VM 1",
      "workloadId": "b4ee62da-8dff-4a6b-b39f-54acf26a3a6d",
      "transactionId": "654a5e87-04ff-439c-bd51-f08bb033e580"
    },
    "parameters": {
      "imageUri": "image/hpcloud/nova/ami-00000075",
      "keyPairId": "60e3e8e9-08aa-4934-90e2-0002de271fdc",
      "securityGroupIds": [
        "9a200d98-31ea-4bbc-a264-b83e5d6219f6",
        "b1092157-703a-4f30-9ed6-895f3277bcf5"
      ],
      "vm_providerResourceId": "Standard.small",
      "zone": "nova",
      "cpuCount": "2",
      "cpuSpeed": "1.2",
      "localStorage": "60",
      "ram": "2",
      "username": "ubuntu",
      "image_providerResourceId": "ami-00000075",
      "keyPairId_providerInstanceId": "60e3e8e9-08aa-4934-90e2-0002de271fdc",
      "securityGroupIds_providerInstanceId": [
        568643,
        568641
      ]
    }
  },
  {
    "instanceId": "b6011557-1685-4379-856f-3cec844887fc",
    "created": "2013-12-20T00:04:53.353Z",
    "updated": "2013-12-20T00:04:53.353Z",
    "ownerId": "d00f16e6-7503-4838-a658-8b60594ff285",
    "resourceUri": "vm/hpcloud/nova/standard.small",
    "resourceType": "vm",
    "provider": "hpcloud",
    "region": "nova",
    "attributes": {
      "providerInstanceId": 2715435,
      "password": "ykcLunMnGfH66Coz",
      "instanceStatus": "creating",
      "transientStatus": true
    },
    "attributeTimestamps": {
      "password": "2013-12-20T00:04:53.347Z",
      "instanceStatus": "2013-12-20T00:04:53.351Z"
    },
    "metadata": {
      "description": "hello world - my SECOND virtual machine",
      "name": "VM 2",
      "workloadId": "b4ee62da-8dff-4a6b-b39f-54acf26a3a6d",
      "transactionId": "654a5e87-04ff-439c-bd51-f08bb033e580"
    },
    "parameters": {
```

```
      "imageUri": "image/hpcloud/nova/ami-00000075",
      "keyPairId": "60e3e8e9-08aa-4934-90e2-0002de271fdc",
      "securityGroupIds": [
        "9a200d98-31ea-4bbc-a264-b83e5d6219f6",
        "b1092157-703a-4f30-9ed6-895f3277bcf5"
      ],
      "vm_providerResourceId": "Standard.small",
      "zone": "nova",
      "cpuCount": "2",
      "cpuSpeed": "1.2",
      "localStorage": "60",
      "ram": "2",
      "username": "ubuntu",
      "image_providerResourceId": "ami-00000075",
      "keyPairId_providerInstanceId": "60e3e8e9-08aa-4934-90e2-0002de271fdc",
      "securityGroupIds_providerInstanceId": [
        568643,
        568641
      ]
    }
  }
]
```


Therefore you always need to check what *running instances* you actually have for the workload, because the transaction may have created one or more instances even though the transaction itself has failed or been cancelled. And as noted before, billing considers the instances for charging purposes, not the transactions or workloads.

If you want to perform a "rollback" of all vm and vs instances created by the transaction then you can do a deactivation - see next section.

10.18 Plan Workload Deactivation

To generate a workload plan for deactivation:

```
>node runcws.js deactivate
deactivate (plan workload deactivation)
options: {
  "url": "http://cws.computenext.com/api/workload/b4ee62da-8dff-4a6b-b39f-54acf26a3a6d/plan?action=de
  "method": "put",
  "auth": {
    "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
    "pass": "<hidden>"
  }
}
----- RESULT -----
{
  "workloadStatus": "in-progress",
  "action": "plan-deactivate"
}
```

Get the workload:

```
>node runcws.js getwl
getwl (retrieve workload)
options: {
  "url": "http://cws.computenext.com/api/workload/b4ee62da-8dff-4a6b-b39f-54acf26a3a6d",
  "method": "get",
```

```
  "auth": {
    "user": "63a25f81-15dd-481e-a4d4-c5ed9a80f93a",
    "pass": "<hidden>"
  }
}
----- RESULT -----
{
  "workloadId": "b4ee62da-8dff-4a6b-b39f-54acf26a3a6d",
  "ownerId": "d00f16e6-7503-4838-a658-8b60594ff285",
  "uniqueName": "hellovm",
  "name": "Hello VM",
  "description": "Workload 'hello world' for one VM",
  "created": "2013-12-19T01:23:27.354Z",
  "updated": "2013-12-19T21:52:57.528Z",
  "metadata": {
    "test": "this is metadata for the entire workload - can be anything",
    "test1": "another line of metadata"
  },
  "elements": [
    {
      "name": "VM 1",
      "uri": "vm/hpcloud/nova/standard.small",
      "parameters": {
        "imageUri": "image/hpcloud/nova/ami-00000075",
        "keyPair": "KP 1",
        "securityGroups": [
          "SG 1",
          "SG 2"
        ]
      },
      "metadata": {
        "description": "hello world - my first virtual machine"
      }
    }
  ],
  "execute": {
    "created": "2013-12-19T21:52:57.527Z",
    "action": "activate",
    "transactionId": "06d7ab5f-f420-4fea-b0bf-456fbc14d884",
    "elements": [
      {
        "name": "VM 1",
        "uri": "vm/hpcloud/nova/standard.small",
        "parameters": {
          "imageUri": "image/hpcloud/nova/ami-00000075",
          "keyPairId": "*0000_kp_create_kp1",
          "securityGroupIds": [
            "*0001_sg_create_sg1",
            "*0003_sg_create_sg2"
          ]
        },
        "metadata": {
          "description": "hello world - my first virtual machine",
          "name": "VM 1"
        },
        "resource": {
          "id": "vm_hpcloud_nova_standard-small",
          "uri": "vm/hpcloud/nova/standard.small",
```

```
          "resourceType": "vm",
          "provider": "hpcloud",
          "region": "nova",
          "providerResourceId": "Standard.small",
          "cpuSpeed": "1.2",
          "cpuCount": "2",
          "localStorage": "60",
          "ram": "2",
          "operatingSystemVersion": "64 Bit",
          "zone": "nova",
          "connectorType": "openStack.compute"
        }
      }
    ]
  },
  "workloadStatus": "none",
  "plan": {
    "action": "deactivate",
    "ownerId": "d00f16e6-7503-4838-a658-8b60594ff285",
    "created": "2013-12-19T22:54:48.108Z",
    "expires": "2013-12-19T22:59:48.108Z",
    "elements": [
      {
        "name": "VM 1",
        "uri": "vm/hpcloud/nova/standard.small",
        "parameters": {
          "imageUri": "image/hpcloud/nova/ami-00000075",
          "keyPair": "KP 1",
          "securityGroups": [
            "SG 1",
            "SG 2"
          ]
        },
        "metadata": {
          "description": "hello world - my first virtual machine"
        }
      }
    ],
    "serial": [
      {
        "parallel": [
          {
            "step": {
              "id": "0000_vm_delete_vm1",
              "action": "vm.delete",
              "instanceId": "dd13e743-33be-4b10-b740-99d8cf4f8d71",
              "uri": "vm/hpcloud/nova/standard.small",
              "resource": {
                "provider": "hpcloud",
                "region": "nova",
                "resourceType": "vm"
              },
              "metadata": {
                "description": "hello world - my first virtual machine",
                "name": "VM 1",
                "transactionId": "06d7ab5f-f420-4fea-b0bf-456fbc14d884",
                "originalWorkloadId": "b4ee62da-8dff-4a6b-b39f-54acf26a3a6d"
              },
```

```
              "timing": {
                "min": 15.08,
                "avg": 15.08,
                "max": 15.08
              }
            }
          }
        ]
      }
    ]
  }
}
```


You can see that a step has been added to delete the virtual machine instance.

You can now go ahead and execute this plan (as before, for activate) and it will delete the vm instance.