OpenStack Load Balancing API

OpenStack Specification Proposal

Draft Proposal by Citrix

Based on Rackspace Cloud Load Balancers API v1.0 (Beta)

Table of Contents

1. Overview

This draft API is based on the Rackspace Cloud Load Balancers API (Beta). It is a strict subset of the Rackspace API that can accommodate several Load Balancing vendors, including Citrix NetScaler and Open Source solutions like HA Proxy.

The API allows for extensions by OpenStack Cloud Providers and Load Balancer vendors to expose extra functionality through an API that is a superset of this API.

1.1. Intended Audience

This guide is intended for software developers who want to create applications using the OpenStack Load Balancing API. It assumes the reader has a general understanding of load balancing concepts and is familiar with:

- RESTful web services
- HTTP/1.1 conventions
- JSON and/or XML serialization formats

1.2. Document Change History

This version of the Developer Guide replaces and supersedes all previous versions. The most recent changes are described in the table below:

Revision Date Summary of Changes

Revision Date	Summary of Changes
1st Draft	4th April

2. Concepts

To use the OpenStack Load Balancing API effectively, you should understand several key concepts:

2.1. Load Balancer

A load balancer is a logical device which belongs to a cloud account. It is used to distribute workloads between multiple back-end systems or services, based on the criteria defined as part of its configuration.

2.2. Virtual IP

A virtual IP is the IP address configured on the load balancer for use by clients connecting to a service that is load balanced. Incoming connections are distributed to back-end nodes based on the configuration of the load balancer.

2.3. Node

A node is a back-end device providing a service on a specified IP and port.

2.4. Health Monitor

A health monitor is a feature of the load balancer that is used to determine whether or not a back-end node is usable for processing a request. The service supports two types of health monitors: passive and active.

2.4.1. Passive Health Monitor

By default, all load balancing configurations utilize a passive health monitor, which is the default monitoring and does not require configuration from the user. If the passive health monitoring determines that a node is down, unreachable or malfunctioning, it puts the node in an OFFLINE state and stops sending traffic to it.

2.4.2. Active Health Monitor

Active health monitoring is a technique that uses synthetic transactions that are executed at periodic intervals to determine the condition of a node. When active monitoring is enabled, it takes over the monitoring of the node, and passive monitoring is disabled. Conversely, when active monitoring configuration is removed by the user, passive monitoring is re-enabled for the nodes of the load balancer.

The active health monitor can use one of three types of probes:

- connect
- HTTP
- HTTPS

These probes are executed at configured intervals; in the event of a failure, the node status changes to OFFLINE and the node will not receive traffic. If, after running a subsequent test, the probe detects that the node has recovered, then the node's status is changed to ONLINE and it is capable of servicing requests.

2.5. Session Persistence

Session persistence is a feature of the load balancing service that attempts to force subsequent connections to a service to be redirected to the same node as long as it is ONLINE.

2.6. Connection Logging

The connection logging feature allows for retrieving access logs (for HTTP-based protocol traffic) or connection and transfer logs (for all other traffic).

3. API Operations

3.1. List LoadBalancers

Verb	URI	Description	Representation
GET	/loadbalancers	List all loadbalancers configured for an account	XML, JSON

Normal Response Code(s): 200

Error Response Code(s): loadBalancerFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest(400)

This request returns a list of load balancers currently configured for the account.

This operation does not require a request body.

Example: List Load Balancers Response: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<loadBalancers xmlns="http://docs.openstack.org/loadbalancers/api/v1.0">
  <loadBalancer id="71" name="lb-site1" status="ACTIVE"
    protocol="HTTP"  port="80" algorithm="ROUND_ROBIN">
    <virtualIp address="206.55.130.2" ipVersion="IPV4" type="PUBLIC" />
    <cluster name="dc-eastcoast-4" />
    <created time="2010-11-29T17:31:41Z" />
    <updated time="2010-11-30T08:35:15Z" />
  </loadBalancer>
  <loadBalancer id="166"  name="lb-site2" status="ACTIVE"
    protocol="HTTP" port="80" algorithm="LEAST_CONNECTIONS">
    <virtualIp address="206.55.130.15" ipVersion="IPV4" type="PUBLIC" />
    <cluster name="dc-eastcoast-4" />
    <created time="2010-11-30T03:23:42Z" />
    <updated time="2010-11-30T03:23:44Z" />
  </loadBalancer>
</loadBalancers>
```

Example: List Load Balancers Response: JSON


```
{
    "loadBalancers" :  [
            {
                  "id" : "71",
                  "name":"lb-site1",
                  "status":"ACTIVE",
                  "protocol":"HTTP",
                  "port":"80",
                  "algorithm":"ROUND_ROBIN",
                  "virtualIp": {
                                  "address":"206.55.130.2",
                                  "ipVersion":"IPV4",
                                  "type":"PUBLIC"
                                },
                  "nodes": [
                          {
                               "id":"1041",
                               "address":"10.1.1.1",
                               "port":"80",
                               "condition":"ENABLED",
                               "status":"ONLINE"
                          },
                          {
                               "id":"1411",
                               "address":"10.1.1.2",
                               "port":"80",
                               "condition":"ENABLED",
                               "status":"ONLINE"
                          }
                      ],
                  "sessionPersistence": {
                              "persistenceType":"HTTP_COOKIE"
                                      },
                  "connectionLogging":{
                             "enabled":"true"
                                     },
                  "cluster":{
                              "name":"dc-eastcoast-4"
                            },
                  "created":{
                              "time":"2010-11-30T03:23:42.000+0000"
                            },
                  "updated":{
                              "time":"2010-11-30T03:23:44.000+0000"
                            }
            }
        ]
}
```

3.2. List LoadBalancer Details

Verb	URI	Description	Representation
GET	/loadbalancers/*loadBalancerId*	List details of the specified load balancer	XML, JSON

Normal Response Code(s): 200

Error Response Code(s): loadBalancerFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest(400), overLimit (413)

This operation provides detailed output for a specific load balancer configured and associated with your account.

This operation does not require a request body.

Example: List Load Balancer Details Response: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<loadBalancer xmlns="http://docs.openstack.org/loadbalancers/api/v1.0" id="71" name="lb-site1"
status="ACTIVE" protocol="HTTP"  port="80" algorithm="ROUND_ROBIN">
   <virtualIp address="206.55.130.2" ipVersion="IPV4" type="PUBLIC" />
   <nodes>
      <node   id="1041" address="10.1.1.1" port="80" condition="ENABLED"  status="ONLINE" />
      <node   id="1411" address="10.1.1.2" port="80" condition="ENABLED"  status="ONLINE" />
   </nodes>
   <sessionPersistence     persistenceType="HTTP_COOKIE"  />
   <connectionLogging enabled="true" />
   <cluster name="c1.dfw1" />
   <created time="2010-11-30T03:23:42Z" />
   <updated time="2010-11-30T03:23:44Z" />
</loadBalancer>
```

Example: Get Load Balancer Details Response: JSON


```
{
       "id" : "71",
       "name":"lb-site1",
       "status":"ACTIVE",
       "protocol":"HTTP",
       "port":"80",
       "algorithm":"ROUND_ROBIN",
       "virtualIp": {
                        "address":"206.55.130.2",
                        "ipVersion":"IPV4",
                        "type":"PUBLIC"
                     },
       "nodes": [
                {
                     "id":"1041",
                     "address":"10.1.1.1",
                     "port":"80",
                     "condition":"ENABLED",
                     "status":"ONLINE"
                },
                {
                     "id":"1411",
                     "address":"10.1.1.2",
                     "port":"80",
                     "condition":"ENABLED",
                     "status":"ONLINE"
                }
           ],
       "sessionPersistence": {
                    "persistenceType":"HTTP_COOKIE"
                              },
       "connectionLogging":{
                   "enabled":"true"
                            },
       "cluster":{
                   "name":"c1.dfw1"
                  },
       "created":{
                   "time":"2010-11-30T03:23:42.000+0000"
                  },
       "updated":{
                   "time":"2010-11-30T03:23:44.000+0000"
                  }
}
```

3.3. Create LoadBalancer

Verb	URI	Description	Representation
POST	/loadbalancers	Create a new load balancer with the configuration defined by the request.	XML, JSON

Normal Response Code(s): 202

Error Response Code(s): loadBalancerFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest(400), overLimit (413)

This operation asynchronously provisions a new load balancer based on the configuration defined in the request object. Once the request is validated and progress has started on the provisioning process, a response object will be returned. The object will contain a unique identifier and status of the request. Using the identifier, the caller can check on the progress of the operation by performing a GET on `loadbalancers/id`. If the corresponding request cannot be fulfilled due to insufficient or invalid data, an HTTP 400 (Bad Request) error response will be returned with information regarding the nature of the failure in the body of the response. Failures in the validation process are non-recoverable and require the caller to correct the cause of the failure and POST the request again.

> Note
>
> Users may configure all documented features of the load balancer at creation time by simply providing the additional elements / attributes in the request. Refer to the subsequent sections of this specification for an overview of all features the load balancing service supports.

Example: Create Load Balancer (Required Attributes) Request: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<loadBalancer xmlns="http://docs.openstack.org/loadbalancers/api/v1.0" id="71" name="a-new-loadbalancer" protocol="HTTP"  port="80" algorithm="ROUND_ROBIN">
   <virtualIp  type="PUBLIC" />
   <nodes>
      <node   address="10.1.1.1" port="80"  />
      <node   address="10.1.1.2" port="8080"   />
   </nodes>
</loadBalancer>
```

Example: Create Load Balancer (Required Attributes) Response: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<loadBalancer xmlns="http://docs.openstack.org/loadbalancers/api/v1.0" id="71" name="a-new-loadbalancer" status="ACTIVE" protocol="HTTP"  port="80" algorithm="ROUND_ROBIN">
   <virtualIp address="206.55.130.2" ipVersion="IPV4" type="PUBLIC" />
   <nodes>
      <node   id="1041" address="10.1.1.1" port="80" condition="ENABLED"  status="ONLINE" />
      <node   id="1411" address="10.1.1.2" port="8080" condition="ENABLED"  status="ONLINE" />
   </nodes>
   <cluster name="dc-eastcoast-4" />
   <created time="2011-03-14T11:41:37Z" />
   <updated time="2011-03-15T08:05:21Z" />
</loadBalancer>
```

Example: Create Load Balancer (Required Attributes) Request: JSON

```
{
    "loadBalancer" :  {
              "name":"a-new-loadbalancer ",
              "protocol":"HTTP",
              "port":"80",
              "algorithm":"ROUND_ROBIN",
              "virtualIp": {"type":"PUBLIC"},
              "nodes": [
                       {
                           "address":"10.1.1.1",
                           "port":"80",
                       },
                       {
                           "address":"10.1.1.2",
                           "port":"8080",
                       }
                  ]
        }
}
```

Example: Create Load Balancer (Required Attributes) Response: JSON


```
{
    "loadBalancer" :
       {
            "id" : "71",
             "name":"a-new-loadbalancer",
              "status":"ACTIVE",
              "protocol":"HTTP",
              "port":"80",
              "algorithm":"ROUND_ROBIN",
              "virtualIp": {
                            "address":"206.55.130.2",
                            "ipVersion":"IPV4",
                            "type":"PUBLIC"
                            },
              "nodes": [
                       {
                          "id":"1041",
                           "address":"10.1.1.1",
                           "port":"80",
                           "condition":"ENABLED",
                           "status":"ONLINE"
                       },
                       {
                          "id":"1411",
                           "address":"10.1.1.2",
                           "port":"80",
                           "condition":"ENABLED",
                           "status":"ONLINE"
                       }
                   ],
              "sessionPersistence": {
                           "persistenceType":"HTTP_COOKIE"
               },
              "connectionLogging":{
                           "enabled":"true"
              },
              "cluster":{
                           "name":"c1.dfw1"
                           },
              "created":{
                           "time":"2010-11-30T03:23:42.000+0000"
                         },
              "updated":{
                           "time":"2010-11-30T03:23:44.000+0000"
                }
        }
}
```

If you have at least one load balancer, you may create subsequent load balancers that share a single virtual IP by issuing a POST and supplying a virtual IP address instead of a type. Additionally, this feature is highly desirable if you wish to load balance both an unsecured and secure protocol using one IP / DNS name (for example, HTTP and HTTPS).

Example: Create Load Balancer (Required Attributes with Shared IP)
Request: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<loadBalancer xmlns="http://docs.openstack.org/loadbalancers/api/v1.0" name="another-loadbalancer"
protocol="HTTPS"  port="80" algorithm="ROUND_ROBIN">
   <virtualIp  address="206.10.10.210" />
   <nodes>
      <node   address="10.1.1.3" port="80"  />
      <node   address="10.1.1.4" port="80"   />
   </nodes>
</loadBalancer>
```

Example: Create Load Balancer (Required Attributes with Shared IP)
Response: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<loadBalancer xmlns="http://docs.openstack.org/loadbalancers/api/v1.0" id="83" name="another-
loadbalancer" status="ACTIVE" protocol="HTTP"  port="80" algorithm="ROUND_ROBIN">
   <virtualIp address="206.55.130.2" ipVersion="IPV4" type="PUBLIC" />
   <nodes>
      <node   id="1421" address="10.1.1.3" port="80" condition="ENABLED"  status="ONLINE" />
      <node   id="1422" address="10.1.1.4" port="80" condition="ENABLED"  status="ONLINE" />
   </nodes>
</loadBalancer>
```

Example: Create Load Balancer (Required Attributes with Shared IP)
Request: JSON

```
{
    "loadBalancer" :
       {
             "name":"another-loadbalancer ",
              "protocol":"HTTP",
              "port":"80",
              "algorithm":"ROUND_ROBIN",
              "virtualIp": {
                             "address":"206.10.10.210"
                             },
              "nodes": [
                       {
                           "address":"10.1.1.3",
                           "port":"80",
                       },
                       {
                           "address":"10.1.1.4",
                           "port":"80",
                       }
                  ]
        }
}
```

Example: Create Load Balancer (Required Attributes with Shared IP)
Response: JSON


```
{
    "loadBalancer" :
       {
             "id":"83 ",
             "name":"another-loadbalancer ",
              "protocol":"HTTP",
              "port":"80",
              "algorithm":"ROUND_ROBIN",
              "virtualIp": {
                            "address":"206.10.10.210",
                            "ipVersion":"IPV4",
                            "type":"PUBLIC"
                            },
              "nodes": [
                      {
                          "id":"1421 ",
                           "address":"10.1.1.3",
                           "port":"80",
                           "condition":"ENABLED",
                           "status":"ONLINE"
                      },
                      {
                          "id":"1422 ",
                           "address":"10.1.1.4",
                           "port":"80",
                           "condition":"ENABLED",
                           "status":"ONLINE"
                      }
                      ],
              "cluster":{
                           "name":"c1.dfw1"
                         },
              "created":{
                           "time":"2010-11-30T03:23:42.000+0000"
                         },
              "updated":{
                           "time":"2010-11-30T03:23:44.000+0000"
                 }
        }
}
```

3.4. Remove Load Balancer

Verb	URI	Description
DELETE	/loadbalancers/*loadBalancerId*	Removes a loadbalancer from an account

Normal Response Code(s): 202

Error Response Code(s): loadBalancerFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest(400)

The remove load balancer function removes the specified load balancer and its associated configuration from the account. Any and all configuration data is immediately purged and is not recoverable.

This operation does not require a request body.

This operation does not return a response body.

3.5. List, Add, Modify, and Remove Nodes

Verb	URI	Description	Representation
GET	/loadbalancers/*loadBalancerId*/nodes	List nodes configured for the load balancer.	XML, JSON
POST	/loadbalancers/*loadBalancerId*/nodes/*nodeId*	Add a new node to the load balancer	XML, JSON
PUT	/loadbalancers/*loadBalancerId*/nodes/*nodeId*	Modifies the configuration of a node on the load balancer	XML, JSON
DELETE	/loadbalancers/*loadBalancerId*/nodes/*nodeId*	Removes a node from the load balancer	
GET	/loadbalancers/*loadBalancerId*/nodes/*nodeId*	List details for a specific node	XML, JSON

The nodes defined by the load balancer are responsible for servicing the requests received through the load balancer's virtual IP. By default, the load balancer employs a basic health check that ensures the node is listening on its defined port. The node is checked at the time of addition and at regular intervals as defined by the load balancer health check configuration. If a back-end node is not listening on its port or does not meet the conditions of the defined active health check for the load balancer, then the load balancer will not forward connections and its status will be listed as OFFLINE. Only nodes that are in an ONLINE status will receive and be able to service traffic from the load balancer.

All nodes have an associated status that indicates whether the node is online or offline. Only nodes that are in an ONLINE status will receive and be able to service traffic from the load balancer. The OFFLINE status represents a node that cannot accept or service traffic. The status is determined by the passive or active health monitors.

The caller can assign weights to the nodes as part of the weight attribute of the node element. When the nodes do not already have an assigned weight, the service will automatically set the weight to "1" for all nodes.

When a node is added, it is assigned a unique identifier that can be used for mutating operations such as changing the condition or removing it.

3.5.1. List Nodes

Verb	URI	Description	Representation
GET	/loadbalancers/*loadBalancerId*/nodes	List nodes configured for the load balancer.	XML, JSON

Normal Response Code(s): 200

Error Response Code(s): loadBalancerFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest(400)

This operation does not require a request body.

Example: List Node Response: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<nodes xmlns="http://docs.openstack.org/loadbalancers/api/v1.0" >
   <node   id="1421" address="10.1.1.3" port="80" condition="ENABLED"  status="ONLINE" />
   <node   id="1422" address="10.1.1.4" port="80" condition="ENABLED"  status="ONLINE" />
</nodes>
```

Example: List Node Response: JSON

```
{
        "nodes": [
                {
                    "id":"1421",
                    "address":"10.1.1.3",
                    "port":"80",
                    "condition":"ENABLED",
                    "status":"ONLINE"
                },
                {
                     "id":"1422",
                     "address":"10.1.1.4",
                     "port":"80",
                     "condition":"ENABLED",
                     "status":"ONLINE"
                },
            ]
}
```

3.5.2. Add Nodes

Verb	URI	Description	Representation
POST	/loadbalancers/*loadBalancerId*/nodes	Add nodes to the load balancer.	XML, JSON

Normal Response Code(s): 202

Error Response Code(s): loadBalancerFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest(400), overLimit (413)

Example: Add Nodes Request: XML

URI: /loadbalancers/4532/nodes

Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<nodes xmlns="http://docs.openstack.org/loadbalancers/api/v1.0" >
   <node   address="10.1.1.5" port="80" />
   <node   address="10.1.1.10" port="80" weight="2" />
</nodes>
```

Response:

```
<?xml version="1.0" encoding="UTF-8"?>
<nodes xmlns="http://docs.openstack.org/loadbalancers/api/v1.0" >
   <node   id="271" address="10.1.1.5" port="80" condition="ENABLED"  status="ONLINE" />
   <node   id="1231" address="10.1.1.10" port="80" weight="2" condition="ENABLED"
status="ONLINE" />
</nodes>
```

Example: Add Nodes Request: JSON

URI: /loadbalancers/4532/nodes

Request:

```
{
        "nodes": [
                {
                    "address":"10.1.1.5",
                    "port":"80",
                },
                {
                     "address":"10.1.1.10",
                     "port":"80",
                     "weight":"2"
                },
            ]
}
```

Response:

```
{
        "nodes": [
                {
                    "id":"271",
                    "address":"10.1.1.5",
                    "port":"80",
                    "condition":"ENABLED",
                    "status":"ONLINE"
                },
                {
                    "id":"1231",
                     "address":"10.1.1.10",
                     "port":"80",
                     "weight":"2",
                    "condition":"ENABLED",
                    "status":"ONLINE"
                },
            ]
}
```

3.5.3. Modify Node

Verb	URI	Description	Representation
PUT	/loadbalancers/*loadBalancerId*/nodes/*nodeId*	Modifies attributes of a balancer's node.	XML, JSON

Normal Response Code(s): 202

Error Response Code(s): loadBalancerFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest(400)

Note

The node's IP and Port are immutable attributes and cannot be modified by the caller for a PUT request. Supplying an unsupported attribute will result in a 400 (badRequest) fault.

This operation does not return a response.

Example: Modify Node's Condition Request: XML

URI: /loadbalancers/4532/nodes/271

Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<node xmlns="http://docs.openstack.org/loadbalancers/api/v1.0"   condition=" DISABLED"  />
```

Example: Modify Node's Condition and Weight Request: JSON

URI: /loadbalancers/4532/nodes/271

Request:

```
{
      "condition":"DISABLED" ,
      "weight":"4"
}
```

3.5.4. Remove Node

Verb	URI	Description
DELETE	/loadbalancers/*loadBalancerId*/nodes/*nodeId*	Removes a node from the load balancer

Normal Response Code(s): 202

Error Response Code(s): loadBalancerFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest(400)

This operation does not require a request body.

This operation does not return a response body.

3.6. Update Load Balancer Attributes

Verb	URI	Description	Representation
PUT	/loadbalancers/*loadBalancerId*	Update the properties of the load balancer	XML, JSON

Normal Response Code(s): 202

Error Response Code(s): loadBalancerFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest(400), overLimit (413)

This operation asynchronously updates the attributes of the specified load balancer. Upon successful validation of the request, the service will return a 202 (Accepted) response code.

A caller can poll the load balancer with its ID to wait for the changes to be applied and the load balancer to return to an ACTIVE status.

This operation allows the caller to change one or more of the following attributes:

- name
- algorithm

This operation does not return a response body.

> Note
>
> The load balancer's ID, port, protocol and status are immutable attributes and cannot be modified by the caller. Supplying an unsupported attribute will result in a 400 (badRequest) fault.

Example: Modify LoadBalancer's Algorithm Request: XML

URI: /loadbalancers/4532

Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<loadBalancer  xmlns="http://docs.openstack.org/loadbalancers/api/v1.0"
algorithm="LEAST_CONNECTIONS"  />
```

Example: Modify LoadBalancer's name Request: JSON

URI: /loadbalancers/738

Request:

```
{
        "name":"loadbalancer-v2"
}
```

3.7. Usage Reporting

Verb	URI	Description	Representation
GET	/loadbalancers/*loadBalancerId*/usage	List current usage	XML, JSON

Normal Response Code(s): 200

Error Response Code(s): loadBalancerFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest(400)

The load balancer usage reports provide a view of all data transfer associated with the load balancer. Values for both incomingTransfer and outgoingTransfer are expressed in bytes transferred.

This operation does not require a request body.

Example: Get Usage Response: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<loadBalancerRecords xmlns="http://docs.openstack.org/loadbalancers/api/v1.0" >
        <loadBalancerUsageRecord  incomingTransfer="97301"  outgoingTransfer="241050"
startTime="2010-12-21T12:32:07-06:00"  endTime="2010-12-21T12:36:30-06:00"  />
</loadBalancerRecords>
```

Example: Get Usage Response: JSON

```
{
        "loadBalancerRecords": [
                {
                    "incomingTransfer":"1421",
                    "outgoingTransfer":"10.1.1.3",
                    "startTime":"2010-12-21T12:32:07-06:00",
                    "endTime":"2010-12-21T12:36:30-06:00"
                }
            ]
}
```

3.8. Active Health Monitoring

Verb	URI	Description	Representation
GET	/loadbalancers/*loadBalancerId*/healthmonitor	Retrieves the Health monitor configuration, if one exists	XML, JSON
PUT	/loadbalancers/*loadBalancerId*/healthmonitor	Updates the settings for a health monitor	XML, JSON
DELETE	/loadbalancers/*loadBalancerId*/healthmonitor	Removes the health monitor	

The load balancing service includes a health monitoring operation which periodically checks your back-end nodes to ensure they are responding correctly. If a node is not responding, it is removed from rotation until the health monitor determines that the node is functional. In addition to being performed periodically, the health check also is performed against every node that is added to ensure that the node is operating properly before allowing it to service traffic.

Every health monitor has a type attribute to signify what kind of monitor it is.

Name	Description
CONNECT	Health monitor is a connect monitor
HTTP	Health monitor is an HTTP monitor
HTTPS	Health monitor is an HTTPS monitor

3.8.1. Connection Monitor

The monitor connects to each node on its defined port to ensure that the service is listening properly. The connect monitor is the most basic type of health check and does no post-processing or protocol specific health checks. It includes several configurable properties:

- **delay**: The minimum number of seconds to wait before checking the health of a node after it is put into OFFLINE status.
- **timeout**: Maximum number of seconds to wait for a connection to be established before timing out.

- **attemptsBeforeDeactivation**: Number of permissible monitor failures before removing a node from rotation.

3.8.2. HTTP/HTTPS Monitor

The HTTP & HTTPS monitors is a more intelligent monitor that is capable of processing a HTTP or HTTPS response to determine the condition of a node. It supports the same basic properties as the connect monitor and includes three additional attributes that are used to evaluate the HTTP response.

- **method:** The HTTP method (GET or HEAD) that will be used in the sample request
- **path**: The HTTP path that will be used in the sample request

A monitoring probe is considered to have failed if the response received has a status code that is not in the 2xx or 3xx range.

3.8.3. Retrieve the Health Monitor

Verb	URI	Description	Representation
GET	/loadbalancers/*loadBalancerId*/healthmonitor	Retrieves the Health monitor configuration, if one exists	XML, JSON

Normal Response Code(s): 200

Error Response Code(s): loadBalancerFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest(400)

This operation does not require a request body.

Example: Retrieve Health Monitor (type CONNECT): XML

URI: /loadbalancers/4532/healthmonitor

Response:

```
<?xml version="1.0" encoding="UTF-8"?>
<healthMonitor  xmlns="http://docs.openstack.org/loadbalancers/api/v1.0"  type="CONNECT"
delay="10" timeout="10"  attemptsBeforeDeactivation="3" />
```

Example: Retrieve Health Monitor (type CONNECT): JSON

URI: /loadbalancers/4532/healthmonitor

Response:

```
{
      "type":"CONNECT" ,
      "delay":"10",
      "timeout":"10",
      "attemptsBeforeDeactivation":"3"
}
```

Example: Retrieve Health Monitor (type HTTP): XML

URI: /loadbalancers/4532/healthmonitor

Response:

```
<?xml version="1.0" encoding="UTF-8"?>
<healthMonitor  xmlns="http://docs.openstack.org/loadbalancers/api/v1.0"  type="HTTP"
delay="10" timeout="10"  attemptsBeforeDeactivation="3" path="/"  method="HEAD"  />
```

Example: Retrieve Health Monitor (type HTTP): JSON

URI: /loadbalancers/4532/healthmonitor

Response:

```
{
      "type":"HTTP" ,
      "delay":"10",
      "timeout":"10",
      "attemptsBeforeDeactivation":"3",
      "path":"/",
      "method":"HEAD"
}
```

3.8.4. Update a Health Monitor

Verb	URI	Description	Representation
PUT	/loadbalancers/*loadBalancerId*/healthmonitor	Updates the settings for a health monitor	XML, JSON

Normal Response Code(s): 202

Error Response Code(s): loadBalancerFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest(400)

This operation does not return a response.

Example: Update Health Monitor (type HTTPS): XML

URI: /loadbalancers/4532/healthmonitor

Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<healthMonitor  xmlns="http://docs.openstack.org/loadbalancers/api/v1.0"  type="HTTPS"
delay="20" timeout="3"  attemptsBeforeDeactivation="5" path="/check"  method="GET"  />
```

Example: Update Health Monitor (type HTTPS): JSON

URI: /loadbalancers/4532/healthmonitor

Request:

```
{
      "type":"HTTPS" ,
      "delay":"20",
      "timeout":"3",
      "attemptsBeforeDeactivation":"5",
      "path":"/check",
      "method":"GET",
}
```

3.8.5. Delete a Health Monitor

Verb	URI	Description	Representation
DELETE	/loadbalancers/*loadBalancerId*/healthmonitor	Deletes the health monitor associated with this load balancer.	

Normal Response Code(s): 202

Error Response Code(s): loadBalancerFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest(400)

This operation does not require a request.

This operation does not return a response.

After deleting the health monitor, the load balancer’s nodes health will be monitored using passive health monitoring.

3.9. Session Persistence

Verb	URI	Description	Representation
GET	/loadbalancers/*loadBalancerId*/sessionpersistence	List session persistence configuration.	XML, JSON
PUT	/loadbalancers/*loadBalancerId*/sessionpersistence	Enable session persistence.	XML, JSON
DELETE	/loadbalancers/*loadBalancerId*/sessionpersistence	Disable session persistence.	

Session persistence is a feature of the load balancing service which forces subsequent requests from clients to be directed to the same node. This is common with many web applications that do not inherently share application state between back-end servers.

Table: Session Persistence Modes

Name	Description
HTTP_COOKIE	A session persistence mechanism that inserts an HTTP cookie and is used to determine the destination back-end node. This is supported for HTTP load-balancing only.

3.9.1. Retrieve Session Persistence configuration

Verb	URI	Description	Representation
GET	/loadbalancers/*loadBalancerId*/sessionpersistence	List session persistence configuration.	XML, JSON

Normal Response Code(s): 200

Error Response Code(s): loadBalancerFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest(400)

This operation does not require a request.

Example: Retrieve session persistence: XML

URI: /loadbalancers/4532/sessionpersistence

Response:

```
<?xml version="1.0" encoding="UTF-8"?>
<sessionpersistence   xmlns="http://docs.openstack.org/loadbalancers/api/v1.0"
persistenceType="HTTP_COOKIE" />
```

Example: Retrieve session persistence: JSON

URI: /loadbalancers/4532/sessionpersistence

Response:

```
{
      "persistenceType":"HTTP_COOKIE"
}
```

3.9.2. Enable Session Persistence on a Load Balancer

Verb	URI	Description	Representation
PUT	/loadbalancers/*loadBalancerId*/sessionpersistence	Enable session persistence	XML, JSON

Normal Response Code(s): 202

Error Response Code(s): loadBalancerFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest(400)

This operation does not return a response.

Example: Enable session persistence: XML

URI: /loadbalancers/4532/sessionpersistence

Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<sessionPersistence   xmlns="http://docs.openstack.org/loadbalancers/api/v1.0"
persistenceType="HTTP_COOKIE" />
```

Example: Enable session persistence: JSON

URI: /loadbalancers/4532/sessionpersistence

Request:

```
{
    "persistenceType":"HTTP_COOKIE"
}
```

3.9.3. Disable Session Persistence on a Load Balancer

Verb	URI	Description	Representation
DELETE	/loadbalancers/*loadBalancerId*/sessionpersistence	Disables session persistence.	

Normal Response Code(s): 202

Error Response Code(s): loadBalancerFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest(400)

This operation does not require a request.

This operation does not return a response.

3.10. Connection Logging

Verb	URI	Description	Representation
GET	/loadbalancers/*loadBalancerId*/connectionlogging	View current configuration of connection logging.	XML, JSON
PUT	/loadbalancers/*loadBalancerId*/connectionlogging	Enable or disable connection logging.	XML, JSON

This operation allows the user to enable/disable connection logging on the load balancer.

3.10.1. Retrieve Connection Logging configuration

Verb	URI	Description	Representation
GET	/loadbalancers/*loadBalancerId*/connectionlogging	List connection logging configuration.	XML, JSON

Normal Response Code(s): 200

Error Response Code(s): loadBalancerFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest(400)

This operation does not require a request.

Example: Retrieve connection logging configuration: XML

URI: /loadbalancers/4532/connectionlogging

Response:

```
<?xml version="1.0" encoding="UTF-8"?>
<connectionLogging  xmlns="http://docs.openstack.org/loadbalancers/api/v1.0"  enabled="true" />
```

Example: Retrieve connection logging configuration: JSON

URI: /loadbalancers/4532/connectionlogging

Response:

```
{
      "enabled":"true" ,
}
```

3.10.2. Enable or Disable Connection Logging on a Load Balancer

Verb	URI	Description	Representation
PUT	/loadbalancers/*loadBalancerId*/connectionlogging	Enable or Disable connection logging	XML, JSON

Normal Response Code(s): 202

Error Response Code(s): loadBalancerFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest(400)

This operation does not return a response.

Example: Enable connection logging: XML

URI: /loadbalancers/4532/connectionlogging

Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<connectionLogging  xmlns="http://docs.openstack.org/loadbalancers/api/v1.0"  enabled="true"/>
```

Example: Disable connection logging: JSON

URI: /loadbalancers/4532/connectionlogging

Request:

```
{
    "enabled":"false"
}
```

3.11. Connection Throttling

Verb	URI	Description	Representation
GET	/loadbalancers/*loadBalancerId*/connectionthrottle	List connection throttling configuration.	XML, JSON
PUT	/loadbalancers/*loadBalancerId*/connectionthrottle	Update throttling configuration.	XML, JSON
DELETE	/loadbalancers/*loadBalancerId*/connectionthrottle	Remove connection throttling configurations.	

The connection throttling feature imposes limits on the number of requests from a single source IP address to help mitigate malicious or abusive traffic to your applications. The following properties can be configured based on the traffic patterns for your sites.

- `maxRequestRate`: Maximum number of requests allowed from one client source IP address in the defined rateInterval.
- `rateInterval`: Frequency (in seconds) at which the maxRequestRate is assessed.

For example, a `maxRequestRate` of 30 with a `rateInterval` of 60 would allow a maximum of 30 requests per minute from a single source IP address.

3.11.1. Retrieve Connection Throttling configuration

Verb	URI	Description	Representation
GET	/loadbalancers/*loadBalancerId*/connectionthrottle	Retrieve connection throttling configuration.	XML, JSON

Normal Response Code(s): 200

Error Response Code(s): loadBalancerFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest(400)

This operation does not require a request.

Example: Retrieve connection throttling: XML

URI: /loadbalancers/4532/connectionthrottle

Response:

```
<?xml version="1.0" encoding="UTF-8"?>
<connectionThrottle  xmlns="http://docs.openstack.org/loadbalancers/api/v1.0"
maxRequestRate="50" rateInterval="60" />
```

Example: Retrieve connection throttling: JSON

URI: /loadbalancers/4532/sessionpersistence

Response:

```
{
 "maxRequestRate":"50" ,
 "rateInterval":"60"
}
```

3.11.2. Update Connection Throttling configuration

Verb	URI	Description	Representation
PUT	/loadbalancers/*loadBalancerId*/connectionthrottle	Update configuration of connection throttling	XML, JSON

Normal Response Code(s): 202

Error Response Code(s): loadBalancerFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest(400)

This operation does not return a response.

Example: Update configuration of connection throttling: XML

URI: /loadbalancers/4532/connectionthrottle

Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<connectionThrottle  xmlns="http://docs.openstack.org/loadbalancers/api/v1.0"
maxRequestRate="50" rateInterval="60" />
```

Example: Update configuration of connection throttling: JSON

URI: /loadbalancers/4532/connectionthrottle

Request:

```
{
 "maxRequestRate":"50" ,
 "rateInterval":"60"
}
```

3.11.3. Remove Connection Throttling Configuration

Verb	URI	Description	Representation
DELETE	/loadbalancers/*loadBalancerId*/connectionthrottle	Remove connection throttle settings.	

Normal Response Code(s): 202

Error Response Code(s): loadBalancerFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest(400)

This operation does not require a request.

This operation does not return a response.

3.12. Load Balancing Protocols

Verb	URI	Description	Representation
GET	/loadbalancers/protocols	List all supported load balancing protocols	

Normal Response Code(s): 200

Error Response Code(s): loadBalancerFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest(400)

This operation does not require a request.

All load balancers must define the protocol of the service which is being load balanced. The protocol selection should be based on the protocol of the back-end nodes. When configuring a load balancer, the port selected will be the default port for the given protocol unless otherwise specified.

Example: List Load Balancing Protocols: XML

URI: /loadbalancers/protocols

Response:

```
<?xml version="1.0" encoding="UTF-8"?>
<protocols  xmlns="http://docs.openstack.org/loadbalancers/api/v1.0" >
   <protocol name="HTTP" port="80" />
   <protocol name="TCP" port="*" />
</protocols>
```

Example: List Load Balancing Protocols: JSON

URI: /loadbalancers/protocols

Response:

```
{
      "protocols":
                        [
                          {
                               "name":"HTTP",
                               "port":"80"
                          },
                          {
                                "name":"TCP",
                                "port":"*"
                           },
                        ]
}
```

3.13. Load Balancing Algorithms

Verb	URI	Description	Representation
GET	/loadbalancers/algorithms	List all supported load balancing algorithms	

Normal Response Code(s): 200

Error Response Code(s): loadBalancerFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest(400)

This operation does not require a request.

All load balancers utilize an algorithm that defines how traffic should be directed between back-end nodes.

Name	Description
LEAST_CONNECTIONS	The node with the lowest number of connections will receive requests. Each connection will be assigned to a node based on the number of

	concurrent connections to the node and its weight.
ROUND_ROBIN	Connections are routed to each of the backend servers in turn, with different proportions of traffic being direct to a node depending on its weight.

Example: List Load Balancing Algorithms: XML

URI: /loadbalancers/algorithms

Response:

```
<?xml version="1.0" encoding="UTF-8"?>
<algorithms   xmlns="http://docs.openstack.org/loadbalancers/api/v1.0" >
   <algorithm name="LEAST_CONNECTIONS" />
   <algorithm name="ROUND_ROBIN" />
</algorithms>
```

Example: List Load Balancing Algorithms: JSON

URI: /loadbalancers/algorithms

Response:

```
{
      "algorithms": [
                        {
                              "name":"LEAST_CONNECTIONS",
                        },
                        {
                              "name":"ROUND_ROBIN",
                        }
                  ]
}
```

3.14. Load Balancer Status

All load balancers have a status attribute to signify the current configuration status of the device. This status is immutable by the caller and is updated automatically based on state changes within the service. When a load balancer is first created, it will be placed into a build status where the

configuration is being generated and applied based on the request. Once the configuration is applied and finalized, it will be in an ACTIVE status. In the event of a configuration change or update, the status of the load balancer will change to PENDING_UPDATE to signify configuration changes are in progress but have not yet been finalized. Load balancers in a SUSPENDED status are configured to reject traffic and will not forward requests to back-end nodes

Table: Load Balancer Statuses

Name	Description
ACTIVE	Load balancer is configured properly and ready to serve traffic to incoming requests via the configured virtual IPs
BUILD	Load balancer is being provisioned for the first time and configuration is being applied to bring the service online. The service will not yet be ready to serve incoming requests.
PENDING_UPDATE	Load balancer is online, but configuration changes are being applied to update the service based on a previous request.
PENDING_DELETE	Load balancer is online, but configuration changes are being applied to begin deletion of the service based on a previous request.
SUSPENDED	Load balancer has been taken offline and disabled
ERROR	The system encountered an error when attempting to configure the load balancer

3.15. Node Condition

Every node is the load balancer has an associated condition which determines its role within the load balancer.

Table: Load Balancer Node Conditions

Name	Description
ENABLED	Node is permitted to accept new connections
DISABLED	Node is not permitted to accept any new connections regardless of session persistence configuration.

4. API Faults

API calls that return an error will return one of the following fault objects. All fault objects will extend from the base fault, serviceFault, for easier exception handling for languages that support it.

4.1. serviceFault

The serviceFault and by extension all other faults include message and detail elements which contain strings describing the nature of the fault as well as a code attribute representing the HTTP response code for convenience. The code attribute of the fault is for the convenience of the caller so that they may retrieve the response code from the HTTP response headers or directly from the fault object if they choose. The caller should not expect the serviceFault to be returned directly but should instead expect only one of the child faults to be returned.

4.2. loadBalancerFault

The loadBalancerFault fault shall be returned in the event that an error occurred during a loadbalancer operation.

```
<?xml version="1.0" encoding="UTF-8"?>
<loadBalancerFault  xmlns="http://docs.openstack.org/loadbalancers/api/v1.0"  code="401">
   <message>Account ID in URL not recognized</message>
</loadBalancerFault>
```

4.3. badRequest

This fault indicates that the data in the request object is invalid; for example, a string was used in a parameter that was expecting an integer. The fault will wrap validation errors.

```
<?xml version="1.0" encoding="UTF-8"?>
<badRequest  xmlns="http://docs.openstack.org/loadbalancers/api/v1.0"  code="400">
   <message>Validation fault</message>
   <details>Validation fault</details>
   <validationErrors>
      <message>Node IP is invalid. Please specify a valid ip. </message>
    </validationErrors>
</badRequest>
```

4.4. itemNotFound

```
<?xml version="1.0" encoding="UTF-8"?>
<itemNotFound  xmlns="http://docs.openstack.org/loadbalancers/api/v1.0"  code="404">
   <message>Load Balancer not found </message>
</itemNotFound>
```

4.5. overLimit

```
<?xml version="1.0" encoding="UTF-8"?>
<overLimit  xmlns="http://docs.openstack.org/loadbalancers/api/v1.0"  code="404">
   <message>
      Your account is currently over the limit so your request could not be processed
   </message>
</overLimit>
```

4.6. unauthorized

This fault is returned when the user is not authorized to perform an attempted operation.

```
<?xml version="1.0" encoding="UTF-8"?>
<unauthorized  xmlns="http://docs.openstack.org/loadbalancers/api/v1.0"  code="404">
   <message> Your auth token is invalid. Please renew. </message>
</unauthorized>
```

4.7. outOfVirtualIps

This fault indicates that there are no virtual IPs left to assign to a new loadbalancer. In practice, this fault should not occur, as virtual IPs will be provisioned as capacity is required.

```
<?xml version="1.0" encoding="UTF-8"?>
<outOfVirtualIps  xmlns="http://docs.openstack.org/loadbalancers/api/v1.0"  code="500">
   <message> Out of virtual IPs. </message>
</outOfVirtualIps>
```

4.8. unprocessableEntity

This fault is returned when an operation is requested on an item that does not support the operation, but the request is properly formed.

```
<?xml version="1.0" encoding="UTF-8"?>
<unprocessableEntity  xmlns="http://docs.openstack.org/loadbalancers/api/v1.0"  code="422">
   <message> The object at the specified URI does not support this operation. </message>
</unprocessableEntity>
```

4.9. serviceUnavailable

This fault is returned when the service is unavailable, such as when the service is undergoing maintenance. Note that this does not necessarily mean that the currently configured loadbalancers are unable to process traffic; it simply means that the API is currently unable to service requests.

```
<?xml version="1.0" encoding="UTF-8"?>
<serviceUnavailable  xmlns="http://docs.openstack.org/loadbalancers/api/v1.0"  code="500">
   <message> The load balancing service is currently not available. </message>
</serviceUnavailable>
```

5. API Extensions

Implementations of this API specification are free to augment it with extensions as they see appropriate to extend Load Balancing features (e.g. support for new LB algorithms) in this API or offer new ones.

All client applications written to this core specification (using the base API) should work against extended implementations as specified in this document. Therefore, applications should not receive values not specified in this specification or obtain a different behavior than expected when they are not using (or aware of the availability of) extended APIs.

5.1. Discovering extensions

Implementations should allow users to discover if the LB service supports extensions, together with details about the extensions (document pointers, etc.).

Verb	URI	Description	Representation
GET	/extensions	Returns extensions supported by this implementation of the Load Balancing service	

Normal Response Code(s): 200

Error Response Code(s): loadBalancerFault (400, 500), serviceUnavailable (503), unauthorized (401), badRequest(400)

Example: Retrieving extension information: XML

URI: /extensions

Response:

```
<?xml version="1.0" encoding="UTF-8"?>
<extensions xmlns="http://docs.openstack.org/common/api/v1.0"
   xmlns:atom="http://www.w3.org/2005/Atom">
   <extension   name="SSL Offload"
      namespace="http://docs.citrix.com/loadbalancing/openstack/v1.0"
      alias="CTX-LBSSL"
      updated="2011-07-05T09:45:16-23:00">
      <description>
         Adds the capability of decrypting SSL traffic by the load balancer before forwarding it to nodes
(SSL Offload).
      </description>
      <atom:link rel="describedby"
         type="application/pdf"
         href="http://docs.citrix.com/loadbalancing/openstack/ext/ctx-lbssl/lbssl-11.pdf"/>
   </extension>
</extensions>
```

5.2. Using extended APIs

Clients should operate on resources using extension URIs in order to make use of the extension.

To create a load balancer with extra features, the client uses the extension URI

```
POST /loadbalancers/ext/<ext-name>
```

The same load-balancer can be retrieved using either:

```
GET /loadbalancers/loadBalancerId
```

in which case, only the features and attributes defined in this document are returned.

Or

```
GET /loadbalancers/ext/<ext-name>/loadBalancerId
```

Which returns the extended version of the load balancer (with extra attributes).

Similarly, if updating the load balancer using the standard attributes, the client can update against the URI

```
PUT /loadbalancers/loadBalancerId
```

And when using extended attributes, the client must use the URI for the extension:

```
PUT /loadbalancers/ext/<ext-name>/loadBalancerId
```

Clients can always use the core APIs for resources created or updated with extension APIs. Extension resources are always defined as pure supersets of core resources.

New types of resources can be added in the extension API that have no counterpart in the core API. These resources can be used only from the extension API.

Example: Create Load Balancer with CTX-LBSSL extension Request: XML

Request : `POST /loadbalancers/ext/ctx-lbssl/`

Request Body

```
<?xml version="1.0" encoding="UTF-8"?>
<loadBalancer xmlns="http://docs.openstack.org/loadbalancers/api/v1.0"
xmlns:ctx="http://docs.citrix.org/netscaler/openstack/api/v1.0"   id="71" name="a-new-
loadbalancer" protocol="HTTPS"  port="80" algorithm="ROUND_ROBIN">
   <virtualIp  type="PUBLIC" ctx:sslоffload="true" />
   <nodes>
      <node   address="10.1.1.1" port="80"  />
      <node   address="10.1.1.2" port="8080"   />
   </nodes>
   <ctx:sslcert   keypair="certkey-ssl-212"  />
</loadBalancer>
```

Example: Create Load Balancer with CTX-LBSSL extension Response: XML

```
<?xml version="1.0" encoding="UTF-8"?>
<loadBalancer xmlns="http://docs.openstack.org/loadbalancers/api/v1.0" id="71"
xmlns:ctx="http://docs.citrix.org/netscaler/openstack/api/v1.0"    name="a-new-loadbalancer"
status="ACTIVE" protocol="HTTP"  port="80" algorithm="ROUND_ROBIN" ctx:sslоffload="true">
   <virtualIp address="206.55.130.2" ipVersion="IPV4" type="PUBLIC" />
   <nodes>
      <node   id="1041" address="10.1.1.1" port="80" condition="ENABLED"  status="ONLINE" />
      <node   id="1411" address="10.1.1.2" port="8080" condition="ENABLED"  status="ONLINE" />
   </nodes>
   <ctx:sslcert   keypair="certkey-ssl-212"  />
</loadBalancer>
```

Example: Create Load Balancer with CTX-LBSSL extension Request: JSON

Request : `POST /loadbalancers/ext/ctx-lbssl/`

Request Body

```
{
    "loadBalancer" :
       {
             "name":"a-new-loadbalancer",
              "protocol":"HTTP",
              "port":"80",
              "algorithm":"ROUND_ROBIN",
              "ctx:sslоffload":"true",
              "virtualIp": {"type":"PUBLIC"},
              "nodes": [
                       {
                            "address":"10.1.1.1",
                            "port":"80",
                       },
                       {
                            "address":"10.1.1.2",
                            "port":"8080",
                       }
                  ],
             "ctx:sslcert":{   "keypair":"certkey-ssl-212" }
         }
}
```

Example: Create Load Balancer with CTX-LBSSL extension Response: JSON


```
{
    "loadBalancer" :
       {
              "id" : "71",
              "name":"a-new-loadbalancer",
              "status":"ACTIVE",
              "protocol":"HTTP",
              "port":"80",
              "algorithm":"ROUND_ROBIN",
              "ctx:sslloffload":"true",
              "virtualIp": {
                            "address":"206.55.130.2",
                            "ipVersion":"IPV4",
                            "type":"PUBLIC"
                            },
              "nodes": [
                       {
                          "id":"1041",
                           "address":"10.1.1.1",
                           "port":"80",
                           "condition":"ENABLED",
                           "status":"ONLINE"
                       },
                       {
                          "id":"1411",
                           "address":"10.1.1.2",
                           "port":"80",
                           "condition":"ENABLED",
                           "status":"ONLINE"
                       }
                   ],
              "sessionPersistence": {
                           "persistenceType":"HTTP_COOKIE"
               },
              "connectionLogging":{
                           "enabled":"true"
              },
              "ctx:sslcert":{
                         "keypair":"certkey-ssl-212"
              },
              "cluster":{
                         "name":"dc-eastcoast-4"
                         },
              "created":{
                         "time":"2010-11-30T03:23:42.000+0000"
                         },
              "updated":{
                         "time":"2010-11-30T03:23:44.000+0000"
                         }
        }
}
```