

# VDI-Portal Management-Server API-Documentation

30. September 2011

**Bachelor-Praktikum**  
Sommersemester 2011  
TU Darmstadt

## Inhaltsverzeichnis

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Common response status codes</b>	<b>3</b>
<b>3</b>	<b>Images</b>	<b>3</b>
3.1	Get all available ISO-images . . . . .	3
<b>4</b>	<b>Nodes</b>	<b>4</b>
4.1	Register a new node . . . . .	4
4.2	Delete a node . . . . .	4
<b>5</b>	<b>Virtual Machines</b>	<b>5</b>
5.1	Create a new VM . . . . .	5
5.2	Get a VM . . . . .	6
5.3	Delete a VM . . . . .	6
5.4	Delete a Virtual Harddisk . . . . .	7
5.5	Search for VMs by tag . . . . .	7
5.6	Update a VM . . . . .	8
5.7	Search for VMs by tag . . . . .	8
5.8	Get all VM types . . . . .	9

## 1 Introduction

This document proposes an API based on the [Representational State Transfer \(REST\)](#) model for accessing the VDI-Portal's resources through a fixed set of operations.

## 2 Common response status codes

These status codes can be returned by any resource.

---

400	BAD REQUEST	Returned if the request was syntactically incorrect
404	NOT FOUND	Returned if the requested resource could not be found
405	METHOD NOT ALLOWED	Returned if it is not allowed to request this resource
406	NOT ACCEPTABLE	Returned if none of the accept types is supported by this resource
415	UNSUPPORTED MEDIA TYPE	Returned if the request content type is not accepted by this resource
422	UNPROCESSABLE ENTITY	Returned if the request body was valid, but cannot be mapped to the requested resource

---

The response body can contain additional information regarding the error and how to resolve it.

## 3 Images

### 3.1 Get all available ISO-images

---

<b>Request</b>	HTTP method	GET
	URL	/images
	Consumes	nothing
<b>Response</b>	Success status code	200 OK (Request was processed without errors)
	Produces	application/json

---

#### Response Example:

```
[
    "Ubuntu_11-04.iso",
    "Fedora_15.iso"
]
```

## 4 Nodes

### 4.1 Register a new node

---

<b>Request</b>	HTTP method	POST
	URL	/node
	Consumes	application/json
<b>Response</b>	Success status code	201 CREATED (Resource was created successfully)
	Produces	application/json
	Failure status code	422 UNPROCESSABLE ENTITY (JSON object was semantically not correct)

---

#### Request Example:

```
{
  "address": "192.168.1.4",
  "resources": {
    "ramSize": 4096,
    "cpuLoad": 0.45,
    "freeDiskSpace": 100395746742
  }
}
```

#### Response Example:

```
{
  "nodeId": 5
}
```

### 4.2 Delete a node

---

<b>Request</b>	HTTP method	DELETE
	URL	/node/{id}
	Parameter	id: [a-zA-Z0-9-]+
	Consumes	nothing
<b>Response</b>	Success status code	204 NO CONTENT (Request was successfully processed, but the server does not return any content)
	Produces	nothing
	Failure status code	404 NOT FOUND (No resource with the requested id found)

---

## 5 Virtual Machines

### 5.1 Create a new VM

---

<b>Request</b>	HTTP method URL Parameter Consumes	POST /vm Header: User: userId application/json
<b>Response</b>	Success status code Produces Failure status code	201 CREATED (Resource was created successfully) application/json 422 UNPROCESSABLE ENTITY (JSON object was semantically not correct)

---

#### Request Example:

```
{
  "name": "My Linux VM",
  "osTypeId": "Linux24",
  "description": "Fedora 16 alpha",
  "memorySize": 512,
  "hddSize": 2,
  "vramSize": 32,
  "accelerate2d": true,
  "accelerate3d": true
}
```

or:

```
{
  "name": "My Linux VM",
  "osTypeId": "Linux24",
  "description": "Fedora 16 alpha",
  "memorySize": 512,
  "hddFile": "/path/to/hdd.vdi",
  "vramSize": 32,
  "accelerate2d": true,
  "accelerate3d": true
}
```

#### Response Example:

```
{
  "id": 4,
  "hddFile": "/path/to/hdd.vdi"
}
```

## 5.2 Get a VM

---

<b>Request</b>	HTTP method	GET
	URL	/vm/{id}
	Parameter	Header: User: userId id: [a-zA-Z0-9-]+
	Consumes	nothing
<b>Response</b>	Success status code	200 OK (Request was processed without errors)
	Produces	application/json
	Failure status code	404 NOT FOUND (No resource with the requested id found)

---

### Response Example:

```
{
  "id": 4,
  "name": "My Linux VM",
  "osTypeId": "Linux24",
  "description": "Fedora 16 alpha",
  "memorySize": 512,
  "hddSize": 2,
  "tags": [
    {
      "identifier": "linux",
      "name": "Linux"
    }
  ],
  "status": "STARTED",
  "rdpUrl": "192.168.1.100:5000",
  "lastActive": "2011-09-11 16:56:33",
  "image": ""
}
```

## 5.3 Delete a VM

---

<b>Request</b>	HTTP method	DELETE
	URL	/vm/{id}
	Parameter	Header: User: userId id: [a-zA-Z0-9-]+
	Consumes	nothing
<b>Response</b>	Success status code	204 NO CONTENT (Request was successfully processed, but the server does not return any content)
	Produces	nothing
	Failure status code	404 NOT FOUND (No resource with the requested id found)

---

## 5.4 Delete a Virtual Harddisk

<b>Request</b>	HTTP method	DELETE
	URL	/vm/disk/{file}
	Parameter	Header: User: userId file: [a-zA-Z0-9-_.]+
	Consumes	nothing
<b>Response</b>	Success status code	204 NO CONTENT (Request was successfully processed, but the server does not return any content)
	Produces	nothing
	Failure status code	404 NOT FOUND (No resource with the requested id found)
		409 CONFLICT (The request could not be completed due to the current state of the resource.) 500 INTERNAL SERVER ERROR (The server encountered an unexpected condition which prevented it from fulfilling the request.)

## 5.5 Search for VMs by tag

<b>Request</b>	HTTP method	GET
	URL	/vm?tag={tag}
	Parameter	Header: User: userId tag: [a-zA-Z0-9-]+
	Consumes	nothing
<b>Response</b>	Success status code	200 OK (Request was processed without errors)
	Produces	application/json
	Failure status code	404 NOT FOUND (No resource with the requested id found)

### Response Example:

```
[
  {
    "id": 4,
    "name": "My Linux VM",
    "osTypeId": "Linux24",
    "description": "Fedora 16 alpha",
    "memorySize": 512,
    "hddSize": 2,
    "tags": [
      {
        "identifier": "linux",
        "name": "Linux"
      }
    ]
  },
]
```

```
        "status": "STARTED",  
        "rdpUrl": "192.168.1.100:5000",  
        "lastActive": "2011-09-11 16:56:33",  
        "image": ""  
    }  
]
```

## 5.6 Update a VM

---

<b>Request</b>	HTTP method	PUT
	URL	/vm/{id}
	Parameter	Header: User: userId id: [a-zA-Z0-9-]+
	Consumes	application/json
<b>Response</b>	Success status code	201 CREATED (Resource was created successfully)
	Produces	nothing
	Failure status code	422 UNPROCESSABLE ENTITY (JSON object was semantically not correct)

---

### Request Example:

```
{  
    "status": "STOPPED",  
    "image": "",  
    "machineName": "Linux2.4",  
    "description": "Fedora Alpha Test image",  
    "memorySize": 1024,  
    "vramSize": 32,  
    "accelerate2d": false,  
    "accelerate3d": true  
}
```

## 5.7 Search for VMs by tag

---

<b>Request</b>	HTTP method	GET
	URL	/vm/{id}?width={width}&height={height}
	Parameter	Header: User: userId id: [a-zA-Z0-9-]+ width: [0-9-]+ height: [0-9-]+
	Consumes	nothing
<b>Response</b>	Success status code	200 OK (Request was processed without errors)
	Produces	image/png
	Failure status code	404 NOT FOUND (No resource with the requested id found)

---



## 5.8 Get all VM types

<b>Request</b>	HTTP method	GET
	URL	/types
	Consumes	nothing
<b>Response</b>	Success status code	200 OK (Request was processed without errors)
	Produces	application/json

### Response Example:

```
{
  "Linux": {
    "linux24": "Linux 2.4",
    "linux24_64": "Linux 2.4 (64 bit)"
  },
  "BSD": {
    "free_bsd": "Free BSD",
    "open_bsd_64": "OpenBSD (64 bit)"
  }
}
```