VDI-Portal Management-Server API-Documentation

11. September 2011

Bachelor-Praktikum
Sommersemester 2011
TU Darmstadt

Inhaltsverzeichnis

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

Request	HTTP method	`GET`
	URL	`/images`
	Consumes	nothing
Response	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

Request	HTTP method	`POST`
	URL	`/node`
	Consumes	application/json
Response	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

Request	HTTP method	`DELETE`
	URL	`/node/{id}`
	Parameter	`id: [a-zA-Z0-9-]+`
	Consumes	nothing
Response	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

Request	HTTP method	`POST`
	URL	`/vm`
	Parameter	Header: User: `userId`
	Consumes	application/json
Response	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:

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

Response Example:

```
{
        "id": 4
}
```

5.2 Get a VM

Request	HTTP method	`GET`
	URL	`/vm/{id}`
	Parameter	Header: User: `userId` `id: [a-zA-Z0-9-]+`
	Consumes	nothing
Response	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

Request	HTTP method	`DELETE`
	URL	`/vm/{id}`
	Parameter	Header: User: `userId` `id: [a-zA-Z0-9-]+`
	Consumes	nothing
Response	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 Search for VMs by tag

Request	HTTP method	`GET`
	URL	`/vm?tag={tag}`
	Parameter	Header: User: `userId` `tag: [a-zA-Z0-9-]+`
	Consumes	nothing
Response	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": ""
        }
]
```

5.5 Update a VM

Request	HTTP method	`PUT`
	URL	`/vm/{id}`
	Parameter	Header: User: `userId` `id: [a-zA-Z0-9-]+`
	Consumes	application/json
Response	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
}
```

5.6 Search for VMs by tag

Request	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
Response	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)

5.7 Get all VM types

Request	HTTP method	`GET`
	URL	`/types`
	Consumes	nothing
Response	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)"
        }
}
```