1&1 Cloud Server API Documentation

Release 1.0

1&1 Internet AG

January 23, 2012

CONTENTS

1 Cloud Server API Reference Documentation **2**
- 1.1 Getting Started . . . 2
- 1.2 Resource CloudServer (/v1/cloudserver) . . . 3
- 1.3 Resource Server . . . 3
- 1.4 Resource Image . . . 4
- 1.5 Resource HardwareConfiguration . . . 5
- 1.6 Resource EstimatedTime . . . 5
- 1.7 Resource PriceInfo . . . 5
- 1.8 Resource Terms . . . 5
- 1.9 Enum ServerState . . . 6
- 1.10 Enum ServerAction . . . 6
- 1.11 Enum ImageOsType . . . 6

2 Cloud API Sandbox Reference Documentation **7**
- 2.1 Getting Started . . . 7
- 2.2 Resource Sandbox (/v1) . . . 8
- 2.3 Resource SandboxUser . . . 9
- 2.4 Resource CloudServer . . . 9

3 Reference **10**
- 3.1 Cloud Server API Reference . . . 10
- 3.2 Cloud API Sandbox Reference Documentation . . . 13

4 Tools **16**
- 4.1 Clients . . . 16

This is the 1&1 Cloud Server REST API documentation. It provides you with an overview of the available resources and supported requests, as well as with a detailed reference.

CHAPTER

ONE

CLOUD SERVER API REFERENCE DOCUMENTATION

The `Cloud Server API` allows for programmatically managing your Dynamic Cloud Server.

Please note that we provide a *sandbox environment* where all API calls are simulated. It is meant for developers to test and debug their applications before using them on a real Dynamic Cloud Server, i.e. detect that your application accidentially reinstalls your server before it is too late.

There are also *client libraries* for all major programming languages available. They allow for immediately start getting productive, as all the boilerplate marshalling and unmarshalling of requests, responses and exceptions is already done for you!

1.1 Getting Started

1.1.1 Communicating with the API: Data Exchange Format

Arguments, either in the query part of an URI (typical for GET requests) or in the request body section (typical for POST and PUT requests) are of the MIME-type `application/x-www-form-urlencoded`.

All responses are of MIME-type `application/json`, and this type must be present in the HTTP-Accept header in order to communicate with the service.

Successful responses return a HTTP status code of **2xx**, whereas clientside errors (such as invalid paths or arguments) return **4xx** and serverside errors return **5xx** respectively.

When an error occurs, the returned error object is always of the form

```
{
  type = ERROR_IDENTIFIER_STRING
  message = SPECIFIC_ERROR_MESSAGE
}
```

1.1.2 Authentication

Each request must be authenticated by means of basic auth. Basic Auth requires the `Authentication` request header to include `Basic <Credentials>` where credentials is a Base64 encoded string, consisting of `YOUR_USERNAME:YOUR_PASSWORD`.

If the credentials of a request are invalid or missing, the service will respond with the HTTP status `401 ("Unauthorized")`.

1.1.3 First Steps

Quickly check if you can access the Cloudserver API endpoint by setting up a "ping" call which should return "OK":

```
$ curl "https://beta.cloud-api.1und1.de/ping"
```

Gather together the username (customer number) and password of your customer or test contract and you are ready for the request examples below:

1.1.4 Request Examples

Example GET call: retrieve all servers:

```
$ curl -H "Accept: application/json" \
--basic --user YOUR_USERNAME:YOUR_PASSWORD \
"https://beta.cloud-api.1und1.de/v1/cloudserver/servers"
```

Example GET call with arguments: retrieve estimated time for a hardware change:

```
$ curl -H "Accept: application/json" \
--basic --user YOUR_USERNAME:YOUR_PASSWORD \
"https://beta.cloud-api.1und1.de/v1/cloudserver/servers/53517
/hardware-configuration/estimated-time-for-new-configuration
?cpus=2&ramSizeInGb=2"
```

Example POST call with arguments: change the server's hardware configuration:

```
$ curl -X POST -d"cpus=2&ramSizeInGb=2" -H "Accept: application/json" \
--basic --user YOUR_USERNAME:YOUR_PASSWORD \
"https://beta.cloud-api.1und1.de/v1/cloudserver/servers/53517
/hardware-configuration"
```

1.1.5 Known bugs/issues

- The server sometimes responds with an "internal server error" when providing wrong credentials.

1.2 Resource CloudServer (/v1/cloudserver)

Entry-point for accessing all Cloudserver-related API calls.

Properties

Methods

GET /servers

Returns all servers of the logged in customer or an empty list if the customer has no server.

GET /servers/{id}

Returns the logged in customer's server of the given id.

1.3 Resource Server

Represents a single dynamic cloud server.

Properties

String uri	The resource's absolute URI.
long id	The resource's identifier.
long contractId	The owning customer's contract ID.
String hostName	The servers hostname.
String ipAddress	The servers IP address.
boolean isConfigurable	Determines whether the server can be configured by the API.

Methods

GET /state

Returns the current server state.

GET /possible-actions

Returns the possible actions according to the current state. I.e. when the server is stopped, only starting is possible, but when the server is running, one may perform a shutdown, a restart etc.

POST /action

Initiates the given action on the server.

GET /possible-images

Returns a list of available images for reinstalling the server.

GET /possible-images/{imageId}

Returns information on the image denoted by the given id.

GET /image

Returns the server's currently installed image.

POST /image

Reinstalls the server with the image denoted by the given id.

GET /hardware-configuration

Returns the server's current number of cpus, ram size in GB and hd size in GB.

POST /hardware-configuration

Reconfigures the server's cpus, ram size and hd size accordingly.

1.4 Resource Image

Represents a dynamic cloud server image.

Properties

String uri	The resource's absolute URI.
long id	The resource's identifier.
String name	The image's name.
ImageOsType osType	The image's OS family (windows, linux or unknown).

Methods

GET /estimated-time-for-new-installation

Returns the minimum and maximum estimated time for reinstalling the server with the given image.

1.5 Resource HardwareConfiguration

Represents a set of cpus, ram and hd resources.

Properties

int cpus	The amount of cpus of the given server.
int ramSizeInGb	The ram size in GB of the given server.
int hdSizeInGb	The hd size in GB of the given server.

Methods

GET /estimated-time-for-new-configuration

Returns the estimated time for reconfiguring the server with the given amount of cpus, ram size and hd size. All arguments are optional. If one argument is missing, no change for the missing agrument's configuration is assumed.

GET /price-info

Returns pricing information for hardware configuration changes.

GET /terms-of-use

Returns the legal terms of use text.

1.6 Resource EstimatedTime

Represents a time estimation with a lower and an upper bound.

Properties

long minimumTimeInSeconds	Minimum estimated duration for the given action in seconds.
long maximumTimeInSeconds	Maximum estimated duration for the given action in seconds.

1.7 Resource PriceInfo

Represents pricing information for configuring the Dynamic Cloud Server.

Properties

String currencyCode	What currency will be used ("EUR" | "USD" | "GBP").
String countryCode	For which country the pricing information is displayed ("de" | "en" | "en_US" | "fr" | "es").
float cpuUnitPrice	The cost for adding one more CPU.
float ramUnitPrice	The cost for adding one more GB of RAM.
float hddUnitPrice	The cost for adding 100 more GB of disk space.
float baseServerPrice	The cost of a Dynamic Cloud server with minimal configuration (1 CPU, 1 GB RAM, 100 GB HDD).

1.8 Resource Terms

The terms and conditions contract text.

Properties

String countryCode	For which country the terms and conditions are displayed (“de” | “en” | “en_US” | “fr” | “es”).
String terms	The terms and conditions text.

1.9 Enum ServerState

Specifies a server state.

RUNNING	The server is up and running.
LOCKED	The server is stopped and locked, rejecting any action request.
REBOOTING	The server is currently rebooting.
RECONFIGURING	The server is reconfiguring (i.e. changing hardware settings or installing a new image).
RESUMING	The server is about to be resumed.
STARTING	The server is about to be started.
STOPPED	The server is currently stopped.
STOPPING	The server is about to be stopped.
SUSPENDED	The server is currently suspended.
SUSPENDING	The server is about to be suspended.
UNKNOWN	The server state is currently unknown.
MAINTENANCE	The server is flagged as “currently under maintenance”.
BUSY	The exact server state is currently unknown, but it is in a transition.

1.10 Enum ServerAction

Specifies an action that can be conducted by a server.

FORCED_RESTART	Forcefully restarts a server by switching the power of a server off and on again.
POWER_OFF	Turns off the power for a server.
RESTART	Shuts down a server and starts it again.
RESUME	Resumes a suspended server.
START	Starts a server that is powered down.
STOP	Shuts down a server.
SUSPEND	Suspends a server.

1.11 Enum ImageOsType

The operating system family of an image.

LINUX	
WINDOWS	
OTHER	

CHAPTER

TWO

CLOUD API SANDBOX REFERENCE DOCUMENTATION

The sandbox environment is meant for our customers and 3rd-party developers to set up test environments for testing their own software that uses our Cloud API.

NOTE: Only Sandbox-specific calls are listed here. For the mirrored calls, please refer to their original documentation, as the API is the same there (it is just that the actions are not triggered, but simulated instead).

2.1 Getting Started

2.1.1 Why use a sandbox

The sandbox environment is meant for our customers and 3rd-party developers to set up test environments for testing and debugging their own software that uses the Cloud API.

The features at a glance:

- The sandbox mirrors all calls provided by the Cloud API.
- All mirrored calls are simulated: no real hardware will be changed and no production services are contacted.
- The sandbox additionally provides means for setting up, customize and delete test customer users (so-called sandbox users). These sandbox users simulate 1&1 customers, so the 3rd-party developer does not need to have several 1&1 contracts for testing his software.
- The sandbox user management calls (all calls under `/sandbox-users`) need authentification (basic auth) with the username (customer number) and password of the "real" 1&1 customer.
- All mirrored calls (i.e. all calls under `/cloudserver`) may only be accessed with the credentials of a previously created simulated 1&1 customer (sandbox user).

2.1.2 First Steps

Quickly check if you can access the Cloud API Sandbox endpoint by setting up a "ping" call which should return "OK":

```
$ curl "https://beta.cloud-api-sandbox.1und1.de/ping"
```

Gather together the username (customer number) and password of your customer or test contract and you are ready for setting up your own test environment on the sandbox:

Set up a sandbox user to be able to use the mirrored calls of the Cloud Server API:

```
$ curl -X POST -d "customerId=123&password=1337" \
-H "Accept: application/json" --basic --user YOUR_USERNAME:YOUR_PASSWORD \
"https://beta.cloud-api-sandbox.1und1.de/v1/sandbox-users"
```

Give the newly created sandbox user a simulated Dynamic Cloud Server. This call is not available in the "real" Cloud Server API, but the sandbox provides it, because 3rd-party developers need to set up their test customers. Note that you need to provide your previously created sandbox user credentials now:

```
$ curl -X POST -d "cpus=3&ramSizeInGb=7&hdSizeInGb=100" \
-H "Accept: application/json" --basic --user 123:1337 \
"https://beta.cloud-api-sandbox.1und1.de/v1/cloudserver/servers"
```

2.1.3 Request Examples

Example GET call: retrieve all servers:

```
$ curl -H "Accept: application/json" --basic --user 123:1337 \
"https://beta.cloud-api-sandbox.1und1.de/v1/cloudserver/servers"
```

Example GET call with arguments: retrieve estimated time for a hardware change:

```
$ curl -H "Accept: application/json" --basic --user 123:1337 \
"https://beta.cloud-api-sandbox.1und1.de/v1/cloudserver/servers
/1/hardware-configuration/estimated-time-for-new-configuration
?cpus=5&ramSizeInGb=8"
```

Example POST call with arguments: change the server's hardware configuration:

```
$ curl -X POST -d"cpus=3&ramSizeInGb=10" \
-H "Accept: application/json" --basic --user 123:1337 \
"https://beta.cloud-api-sandbox.1und1.de/v1/cloudserver/servers
/1/hardware-configuration"
```

2.1.4 Known bugs/issues

- The server sometimes responds with an "internal server error" when providing wrong credentials.

2.2 Resource Sandbox (/v1)

Entry-point for setting up and managing one's own sandbox environment.

Properties

Methods

GET /sandbox-users

Returns all sandbox users created by the logged in customer or an empty list if the customer has no sandbox users.

GET /sandbox-users/{customerId}

Returns the logged in customer's sandbox user of the given id.

POST /sandbox-users

Creates a new sandbox user with given customerId and password.

PUT /sandbox-users/{customerId}

Configures an existing sandbox user.

DELETE /sandbox-users/{customerId}

Deletes the sandbox user with the given id.

DELETE /sandbox-users

Deletes all sandbox users of the logged in customer.

2.3 Resource SandboxUser

Represents a simulated 1&1 customer.

Properties

String uri	The resource's absolute URI.
int customerId	The resource's identifier and simulated 1&1 customer number.
String password;	The simulated customer's login password.

Methods

2.4 Resource CloudServer

Represents a simulated Dynamic Cloud Server owned by a SandboxUser.

Properties

Methods

POST /servers

Adds a new server to the logged in sandbox user.

DELETE /servers/{id}

Deletes the server with the given id.

DELETE /servers

Deletes all servers of the logged in sandbox user.

CHAPTER

THREE

REFERENCE

3.1 Cloud Server API Reference

3.1.1 Resource CloudServer (/v1/cloudserver)

Server[] GET /cloudServer/mobile/servers

Returns

A list with all servers of the logged in customer or an empty list if the customer has no server.

Server GET /servers/{id}

Returns

the logged in customer's server of the given id.

Throws

IllegalArgumentErrorException	if the given id is not numeric.
NotFoundErrorException	if the given id does not exist.

3.1.2 Resource Server

ServerState GET /state

Returns

the current server state.

Throws

IllegalArgumentErrorException	if the given id is not numeric.
NotFoundErrorException	if the given id does not exist.

ServerAction[] GET /possible-actions

Returns

the possible actions according to the current state. I.e. when the server is stopped, only starting is possible, but when the server is running, one may perform a shutdown, a restart etc.

Throws

IllegalArgumentErrorException	if the given id is not numeric.
NotFoundErrorException	if the given id does not exist.

POST /action (*ServerAction* action)

Description

Initiates the given action (i.e. starting, stopping, rebooting...) on the server.

Parameters

ServerAction action	The action to perform.

Throws

IllegalArgumentErrorException	if the given id is not numeric or the given action is not a valid ServerAction? string.
NotFoundErrorException	if the given id does not exist.
IllegalStateErrorException	if the current server state does not allow given action (i.e. stopping an already stopped server).

Image[] GET possible-images

Returns

a list of available images for reinstalling the server.

Throws

IllegalArgumentErrorException	if the given id is not numeric.
NotFoundErrorException	if the given id does not exist.

Image GET /possible-images/{id}

Returns

information on the image denoted by the given id.

Throws

IllegalArgumentErrorException	if the given id is not numeric.
NotFoundErrorException	if the given id does not exist.

Image GET /image

Returns

the server's currently installed image.

Throws

IllegalArgumentErrorException	if the given id is not numeric.
NotFoundErrorException	if the given id does not exist.

Image POST /image (long id)

Description

Reinstalls the server with the image denoted by the given id.

Parameters

id	the ID of the image to install

Throws

IllegalArgumentErrorException	if the given id is not numeric.
NotFoundErrorException	if the given id does not exist.

HardwareConfiguration GET /hardware-configuration

Returns

the server's current number of cpus, ram size in GB and hd size in GB.

Throws

IllegalArgumentErrorException	if the given id is not numeric.
NotFoundErrorException	if the given id does not exist.

POST /hardware-configuration (int cpus default null, int ramSizeInGb default null, int hdSizeInGb default null)

Description

Reconfigures the server's cpus, ram size and hd size accordingly. All arguments are optional: if an argument is missing, the server's current configuration for that argument won't get changed.

Parameters

cpus	the new CPU count. Valid values are from 1 to including 6.
ramSizeInGb	the new RAM size in GB. Valid values are from 1 to including 24.
hdSizeInGb	the new disk size in GB. Valid values are from 100 to including 800.

Throws

IllegalArgumentErrorException	if the given id is not numeric or if one of the given arguments is not numeric or exceeds its validity bounds.
NotFoundErrorException	if the given id does not exist.

3.1.3 Resource Image

EstimatedTime GET /estimated-time-for-new-installation

Returns

the minimum and maximum estimated time for reinstalling the server with the given image.

Throws

IllegalArgumentErrorException	if the given id is not numeric.
NotFoundErrorException	if the given id does not exist.

3.1.4 Resource HardwareConfiguration

EstimatedTime GET /estimated-time-for-new-configuration (int cpus default null, int ramSizeInGb default null, int hdSizeInGb default null)

Description

All arguments are optional. If one argument is missing, no change for the missing agrument's configuration is assumed.

Parameters

cpus	the CPU count. Valid values are from 1 to including 6.
ramSizeInGb	the RAM size in GB. Valid values are from 1 to including 24.
hdSizeInGb	the disk size in GB. Valid values are from 100 to including 800.

Returns

the estimated time for reconfiguring the server with the given amount of cpus, ram size and hd size.

Throws

IllegalArgumentErrorException	if the given id is not numeric.
NotFoundErrorException	if the given id does not exist.

PriceInfo GET /price-info

Returns

pricing information for hardware configuration changes.

Throws

IllegalArgumentErrorException	if the given id is not numeric.
NotFoundErrorException	if the given id does not exist.

Terms GET /terms-of-use

Returns

the legal terms of use text.

Throws

IllegalArgumentErrorException	if the given id is not numeric.
NotFoundErrorException	if the given id does not exist.

3.2 Cloud API Sandbox Reference Documentation

3.2.1 Resource Sandbox (/v1)

SandboxUser[] GET /sandbox-users

Returns

all sandbox users created by the logged in customer or an empty list if the customer has no sandbox users.

SandboxUser GET /sandbox-users/{customerId}

Returns

the logged in customer's sandbox user of the given id.

Throws

IllegalArgumentErrorException	if the given id is not numeric.
NotFoundErrorException	if the given id does not exist.

SandboxUser POST /sandbox-users (int customerId, String password)

Description

Creates a new sandbox user with given customerId and password. Both arguments are mandatory.

Parameters

customerId	the simulated 1&1 customer identifying customer ID.
password	the password the new sandbox user should use for login.

Returns

the newly created sandbox user.

Throws

IllegalArgumentErrorException	if the given customerId is not numeric or refers to an already existing sandbox user.
MissingArgumentErrorException	if either customerId or password is missing.

PUT /sandbox-users/{customerId} (String password)

Description

Configures an existing sandbox user. The password argument is mandatory.

Parameters

password	the new sandbox user's password.

Throws

IllegalArgumentErrorException	if the given customerId is not numeric or refers to an already existing sandbox user.
MissingArgumentErrorException	if the password is missing.
NotFoundErrorException	if the given id does not exist.

DELETE /sandbox-users/{customerId}

Description

Deletes the sandbox user with the given id.

Throws

IllegalArgumentErrorException	if the given id is not numeric.
NotFoundErrorException	if the given id does not exist.

DELETE /sandbox-users

Description

Deletes all sandbox users of the logged in customer.

3.2.2 Resource CloudServer

Server POST /servers (int cpus, int ramSizeInGb, int hdSizeInGb)

Description

Adds a new server to the logged in sandbox user. All arguments are mandatory.

Parameters

cpus	the CPU count. Valid values are from 1 to including 6.
ramSizeInGb	the RAM size in GB. Valid values are from 1 to including 24.
hdSizeInGb	the disk size in GB. Valid values are from 100 to including 800.

Returns

the newly created server.

Throws

IllegalArgumentErrorException	if one of the given arguments is not numeric or exceed its validity bounds.
MissingArgumentErrorException	if at least one argument is missing.

DELETE /servers/{id}

Description

Deletes the server with the given id.

Throws

IllegalArgumentErrorException	if the given id is not numeric.
NotFoundErrorException	if the given id does not exist.

DELETE /servers

Description

Deletes all servers of the logged in sandbox user.

CHAPTER

FOUR

TOOLS

4.1 Clients

4.1.1 Java

- cloudapi-java-client-1.0.3.zip

4.1.2 dotNet (C#)

- cloudapi-dotnet-client-1.0.1.0.zip

4.1.3 Objective-C (iPhone development)

- cloudapi-ios-client-1.0.0.zip

4.1.4 PHP

coming soon!

4.1.5 Python

coming soon!

4.1.6 Sandbox Usage Examples

dotNet Client Sandbox Example Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using CloudapiClient;
using CloudapiClient.Cloudserver;

namespace CloudapiClientDemo
{
  class Demo
  {
    /**
     * Example usage of the Cloud API Java client to work against the sandbox.
     *
```

```
     * NOTE: This example only works if you have already set up
     * your own exclusive test environment within the sandbox.
     *
     * A getting-started guide for the sandbox is available at
     * http://developer.1and1.com/api/cloud_server/api_doc/build/html/index.html
     */
    static void Main()
    {
      // Use the username and password of the sandbox user you created
      // while configuring your sandbox environment here.
      string sandboxUserName = "MY_OWN_CREATED_SANDBOX_USER";
      string sandboxUserPassword = "MY_OWN_CREATED_SANDBOX_USER_PASSWORD";

      CloudserverApi cloudserverApi = new CloudserverApi();
      cloudserverApi.Connect(
        "https://beta.cloud-api-sandbox.1und1.de",
        sandboxUserName,
        sandboxUserPassword
      );

      foreach (Server server in cloudserverApi.GetServers())
      {
        Console.WriteLine("Retrieved server id[{0}] contractId[{1}] " +
          "hostName[{2}] ipAddress[{3}] isConfigurable[{4}]\n",
          server.Id, server.ContractId, server.HostName,
          server.IpAddress, server.IsConfigurable);

        Console.WriteLine("| Retrieved possible images ids[{0}]\n",
          server.GetPossibleImages().Aggregate("", (c, i) => c + " " + i.Id));

        Console.WriteLine("| Retrieved active image[{0}]\n", server.GetImage().Name);

        List<ServerAction> actions = server.GetPossibleActions();
        Console.WriteLine("| Retrieved possibleActions:{0}\n",
          actions.Aggregate("", (c, action) => c + " " + action));

        HardwareConfiguration conf = server.GetHardwareConfiguration();
        Console.WriteLine("| Retrieved configuration cpus[{0}] ram[{1}] hd[{2}]\n",
          conf.Cpus, conf.RamSizeInGb, conf.HdSizeInGb);

        Console.WriteLine();
      }
      Console.ReadKey(true);
    }
  }
}
```

Java Client Sandbox Example Code

```
package cloudserversandboxexample;

import ui.hosting.cloudapi.client.CloudApi;
import ui.hosting.cloudapi.client.cloudserver.Server;

/**
 * Example usage of the Cloud API Java client to work against the sandbox.
 *
 * NOTE: This example only works if you have already set up your own
 * exclusive test environment within the sandbox (a getting-started guide
 * for the sandbox is available at
 * http://developer.1and1.com/api/cloud_server/api_doc/build/html/index.html
```

```
 *
 * @author Nico Siomos
 */
public class CloudServerSandboxExample {

    public static void main(String[] args) {
        // Use the username and password of the sandbox user you created
        // while configuring your sandbox environment here.
        String sandboxUserName = "MY_OWN_CREATED_SANDBOX_USER";
        String sandboxUserPassword = "MY_OWN_CREATED_SANDBOX_USER_PASSWORD";

        CloudApi cloudApiSandbox = new CloudApi(
            "https://beta.cloud-api-sandbox.1und1.de",
            sandboxUserName,
            sandboxUserPassword
        );

        // Displays the servers you added to your sandbox user
        // while configuring your sandbox environment.
        for (Server server : cloudApiSandbox.getCloudServer().getServers()) {
            System.out.println("server name: " + server.getHostName());
            System.out.println("server image: " + server.getImage().getName());
            System.out.println("");
        }
    }
}
```