Node.js API Documentation

Overview

All responses are in JSON

If an error occurs, the server responds with status(500) and returns:
`{"error": error_code, "message": "error message..." }`

A list of potential error codes is provided in the **Error Codes** section.

If the operation is successful, the server responds with status(200) and returns:
`{"message": "success_msg here", ... }`

The “message” is followed by any operation-specific content.

Usage

Most API calls require a valid session token as part of the URI. Tokens are 32-character alphanumeric strings. Tokens are always passed in the URI query, for example:

```
GET ip_addr:port/api_call?token=<your token here>
```

Session tokens are invalidated after **60 days**. We assume that within 60 days our users will be able to restore their failed hardware. Users or course may renew their session.

If a user stops an instance, that instance is held for a period of **2 weeks**. If an instance remains stopped for more than 2 weeks, it is permanently terminated to free space.

User Authentication API Calls

register()

Creates a new user given and allocates the appropriate S3 resources.

Format of Call:

```
POST  /register
```

Request Parameters:

1.	username	passed as part of the request body
2.	email	passed as part of the request body
3.	password	passed as part of the request body

Example Request:

```
52.11.1.237:3000/register
```

Response Content:

1. Only responds with an error if failed or a message if successful.

Example Response:

```
{
    "message": "New user created. S3 bucket provisioned"
}
```

login()

Login an existing user and generate a new session token. If the user is already logged in, a new session token is generated and the old one invalidated.

Format of Call:

```
POST  /login
```

Request Parameters:

1. email	passed as part of the request body
2. password	passed as part of the request body

Example Request:

```
52.11.1.237:3000/login
```

Response Content:

1. user_id	a unique 36-character alphanumeric string for each user
2. token	new, valid session token

Example Response:

```
{
    "user_id": "c2cf281d-061c-40e8-8732-7bedb9e763ec",
    "token": "1d6c35119ac844259380dc89126badfe",
    "message": "login successful"
}
```

logout()

Stop a running instance when it is no longer needed.

Format of Call:

```
POST  /logout
```

Request Parameters:

1. token	this is the one exception where the token is passed as part of the request body. This is for security reasons.

Example Request:

```
52.11.1.237:3000/logout
```

Response Content:

1. Only responds with an error if failed or a message if successful.

Example Response:

```
{
    "message": "Logout successful"
}
```

Backups API Calls

getBackupList()

Get a list of all backups current stored for a given user.

Format of call:

```
GET   /backups/:user_id?token=
```

Request Parameters:

1. user_id	passed as part of the URL
2. token	passed as part of the URL query

Example Request:

```
52.11.1.237:3000/backups/c2cf281d-061c-40e8-8732-7bedb9e763ec?
token=413ed6c7938c47889b43d32f2c525aae
```

Response Content:

backups	array of backup objects containing:
1. backup_id	unique identifier of each backup, similar to token
2. file_size	file size in MB
3. file_name	file name
4. date_created	UTC timestamp as a string

Example Response:

```
{ "message": "Backups obtained successfully",
    "backups": [
        {
            "backup_id": "073b691c7f014fa88fa952983ea138f5",
            "file_size": 28.3,
            "file_name": "backup01.vhdx",
            "date_created": "Thu, 19 Mar 2015 04:42:27 GMT"
        }, ...
    ]
}
```

getBackup()

Get information on a specific backup for a given user.

Format of Call:

```
GET   /backups/:user_id/:backup_id?token=
```

Request Parameters:

1.	user_id	passed as part of the URL
2.	backup_id	passed as part of the URL
3.	token	passed as part of the URL query

Example Request:

```
52.11.1.237:3000/backups/1b177a4c-e727-47c6-9ac5-9a9cc22d54b7/218f88386abf4
64da939e2eb30b7addc?token=b31ae96a983e4c49ace01e4fb49cde08
```

Response Content:

Same as getBackupList()

Example Response:

Same as getBackupList(). backups array contains only one backup object.

startUpload()

Obtains temporary (12 hour) AWS S3 credentials to begin the upload process.

Format of Call:

```
POST  /backups/uploads/:user_id?token=
```

Request Parameters:

1.	user_id	passed as part of the URL
2.	token	passed as part of the URL query
3.	file_name	passed as part of the request body
4.	file_size	passed as part of the request body

Example Request:

```
52.11.1.237:3000/backups/uploads/c2cf281d-061c-40e8-8732-7bedb9e763ec?token
=413ed6c7938c47889b43d32f2c525aae
```

Response Content:

1.	upload_id	unique identifier for the current upload
2.	<u>credentials</u>	<u>object containing:</u>
a.	AccessKeyId	public AWS access key
b.	SecretAccessKey	secret AWS access key
c.	SessionToken	temporary session token needed to make each AWS request
d.	Expiration	localized expiration timestamp

Example Response:

```
{ "message": "Obtained temporary credentials",
    "upload_id": "9a670f35793c477ebf05a5b76d44fc42",
    "credentials": {
        "AccessKeyId": "ASIAINJ3SQ6F7PY2WHBA",
        "SecretAccessKey":"0/hcsjMBTt0aNBCiYxWDeZXxR8mztAqW0juH6tdU",
        "SessionToken":
"AQoDYXdzEID//////////wEagAN3aGUwJIgW7k3+DkMA74zHhQkSLGwdEU/CcDrrVYkn0rPo+6
Ya7ywCRLgPhMWzxFcZGTvNcwZXMmZiDS3jcxmYEXUT1ADkqhc4uMUXwCoqgrSlDjTK9Msj61zHF
SncrhcBoWCrG+913vHWEhtWfY5rf+5aYCqMzRJwoK7CylLx1y5bU3iyb4S+yo2kDomADJp9ynwm
IHGWIcjHrF17F/5sf6DCxuQtIKSinWJ4Gd3Z+hbr/DpXS0VCwIc6cKZ1rwBDvfwANN5GrYN2/di
azbX5ldk8qxKAjxTxmwOTZqo9uGLVGFHy/qJhHXrA8arYxZLRixJVVcexF+m+c2eJJlqUy0e1UO
nls1m/LL3b+y5ll68eTvy4MUNOKfljHBKH6tYHUtMHlZIrVQDiDIHI2Nn0hbmLxajdJmMWkJUxD
i1BOh2vJTdIav9mDRIPAp/9N5XbU+HRkPSlDk3xSdIb5uOZyE1iVMCem59bl5BPzZPxqgb0G9yD
Y+XURNNlj9ckhm8gp5u9qAU=",
        "Expiration": "2015-03-23T11:17:59.000Z"
    }}
```

completeUpload()

Indicate to server that upload attempt is complete (either successful or failed).

Format of Call:

```
PUT   /backups/uploads/:user_id/:upload_id?token=
```

Request Parameters:

1.	user_id	passed as part of the URL
2.	upload_id	passed as part of the URL
3.	token	passed as part of the URL query
4.	upload_status	passed as part of the request body either **‘S’** for successful or **‘F’** for failed

Response Content:

1. only responds with a message indicating error or success

Example Response:

```
{
    "message": "Backup created"
}
```

deleteBackup()

Removes a backup from S3. Generally used to free space as a user's space limits are reached. While the physical backup is destroyed, a record of it is retained by our service.

Format of Call:

```
DELETE  /backups/:user_id/:backup_id?token=
```

Request Parameters:

1. user_id	passed as part of the URL
2. backup_id	passed as part of the URL

Response Content:

1. Only responds with an error if failed or a message if successful

Example Response:

```
{
    "message": "Backup deleted successfully"
}
```

Recovery API Calls

getInstances()

Gets a list of active EC2 instances that are currently running backups for a given user. Typically there is only 1 instance running per user, but it is possible to run more than one if needed.

Format of Call:

```
GET   /recovery/:user_id?token=
```

Request Parameters:

1. user_id   passed as part of the URL

Response Content:

2. only responds with a message indicating error or success

Example Response:

```
{
    "message": "Backup created"
}
```

startInstance()

Starts a new EC2 instance from a .vhd backup stored in S3.

Format of Call:

```
POST  /recovery/:user_id/:backup_id?token=
```

Request Parameters:

1.	user_id	passed as part of the URL
2.	backup_id	passed as part of the URL

Response Content:

1.	instance_id	unique AWS assigned ID of the new EC2 instance
2.	instance_name	generic name generated for the instance
3.	ip_address	public IP that the instance may be reached at
4.	availability_zone	AWS resource zone where the instance was started, typically ‘us-west-2’

Example Response:

```
{
      tbd
}
```

stopInstance()

Stop a running instance when it is no longer needed.

Format of Call:

```
DELETE  /recovery/:instance_id?token=
```

Request Parameters:

1. instance_id   passed as part of the URL

Response Content:

1. only responds with an error if failed or a message if successful

Example Response:

```
{
    "message": "Instance stopped"
}
```

Response Error Codes

List of possible error codes that the API will return. These error codes are grouped by service (user_auth, backups, recovery). All error codes beginning with ‘1’ are system-wide.

system-wide:

101	cannot connect to database
102	invalid token

/register:

201	user with that email already exists
202	invalid username format (must be > 4 characters and < 32)
203	invalid email format
204	invalid password format (at least 6 chars, 1 uppercase, 1 lowercase, 1 number)
205	failed to create S3 bucket when new user was registered (this is a big problem…)

/login:

206	user not found
207	password does not match
208	invalid email format
209	invalid password format

/logout:

211	failed to deactivate token

/backups/getBackupList():

301	failed to get list of backups
302	no backups exist for user

/backups/startUpload():

303	unable to get temporary S3 credentials

/backups/completeUpload():

304	failed to complete upload
305	no upload found
306	failed to get backup information
307	could not verify backup

/backups/deleteBackup():

308	failed to obtain file_name to delete
309	no backup to delete
310	failed to delete backup

/recovery/getInstances():

401	failed to obtain instance information

/recovery/stopInstance():

402	failed to terminate instance
403	failed to update instance information
404	failed to obtain instance status
405	instance not found
406	instance no longer active (permanently stopped)
407	instance already stopped

/recovery/startInstance():

408
409
410
411
412
413
414

Periodic Cron Jobs

ManageSessions
Scans the session tokens table once daily at 4:01am UTC and invalidates any tokens that are more than **60 days** old. We assume that within 60 days our users will be able to restore their failed hardware. Users or course may renew their session.

FreeInstances
Scans the instance table once daily at 4:02am UTC and permanently terminates any instances that have been stopped for more than **2 weeks**. We give a 2 week grace period to our users where they may restart and access a stopped instance before it is permanently destroyed.