Hashtopolis User API (V1)

s3in!c ⟨s3inlc@hashes.org⟩

June 27, 2019

Introduction

The communication for the User API is always in JSON formatted values. When sending a request to the server, it should be a POST containing the JSON data. Every request has a *section* and a *request* field to state which action should be executed. Every response again then contains the requested section and request and gives a *status* to indicate if the query was successful or not. To increase the readability of this document, requests are always in blue, successful responses in green and error messages in red.

Errors

In case of an error with the query which the user sends to the server, the response will have following format with the corresponding action which was requested and the error message which should help in getting information about this error.

```
{
  "section":"task",
  "request":"create",
  "status":"ERROR",
  "message":"You are not allowed to create tasks!"
}
```

Sections

This part lists all sections available on the API. The value in the brackets denotes the according value to be sent on API queries.

- Access Control (*access*)
- Agents (*agent*)
- Server Config (*config*)
- Crackers (*cracker*)
- Files (*file*)
- Groups (*group*)
- Hashlists (*hashlist*)
- Preconfigured Tasks (*pretask*)
- Superhashlists (*superhaslist*)
- Supertasks (*supertask*)
- Tasks (*task*)
- Test (*test*)
- User Management (*user*)

Test (*test*)

This section is used to do testing queries, e.g. to test connectivity or availability of this API. The test section is the only one which allows to make requests without an access key.

connection

Used to test if the URL is a valid API endpoint.

```
{
  "section":"test",
  "request":"connection"
}
```

```
{
  "section": "test",
  "request": "connection",
  "response": "SUCCESS"
}
```

access

Used to check if a given API key is still valid and can be used.

```
{
  "section": "test",
  "request": "access",
  "accessKey": "mykey"
}
```

```
{
  "section": "test",
  "request": "access",
  "response": "OK"
}
```

```
{
  "section": "test",
  "request": "access",
  "response": "ERROR",
  "message": "API key was not found!"
}
```

Agents (*agent*)

Used to access all functions around agents. Please note that the user must have access to the groups where an agent is member of to retrieve it and to be able to apply changes.

listAgents

List all agents with some basic informations.

```
{
  "section": "agent",
  "request": "listAgents",
  "accessKey": "mykey"
}
```

```
{
  "section": "agent",
  "request": "listAgents",
  "response": "OK",
  "agents": [
    {
      "agentId": "2",
      "name": "cracker1",
      "devices": [
        "Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz",
        "NVIDIA Quadro 600"
      ]
    }
  ]
}
```

get

Retrieve all the informations about a specific agent by providing its ID. The last action time is a UNIX timestamp and if the configuration on the server is set to hide the IP of the agents, the value will just be *Hidden* instead of the IP.

```
{
  "section": "agent",
  "request": "get",
  "agentId": 2,
  "accessKey": "mykey"
}
```

```
{
  "section": "agent",
  "request": "get",
  "response": "OK",
  "name": "cracker1",
  "devices": [
    "Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz",
    "NVIDIA Quadro 600"
  ],
  "owner": {
    "userId": 1,
    "username": "htp"
  },
```

```
  "isCpuOnly": false,
  "isTrusted": true,
  "isActive": true,
  "token": "OlBfAp7YQh",
  "extraParameters": "--force",
  "errorFlag": 2,
  "lastActivity": {
    "action": "getTask",
    "time": 1531316240,
    "ip": "127.0.0.1"
  }
}
```

setActive

Set an agent active/inactive.

```
{
  "section": "agent",
  "request": "setActive",
  "active": false,
  "agentId": 2,
  "accessKey": "mykey"
}
```

```
{
  "section": "agent",
  "request": "setActive",
  "response": "OK"
}
```

changeOwner

Either set an owner for an agent or remove the owner from it. The user can either be specified by providing the user ID or the username. If no owner should be specified, the user value must be *null*.

```
{
  "section": "agent",
  "request": "changeOwner",
  "user": 1,
  "agentId": 2,
  "accessKey": "mykey"
}
```

```
{
  "section": "agent",
  "request": "changeOwner",
  "user": "testuser",
  "agentId": 2,
  "accessKey": "mykey"
}
```

```
{
  "section": "agent",
  "request": "changeOwner",
```

```
  "user": null,
  "agentId": 2,
  "accessKey": "mykey"
}
```

```
{
  "section": "agent",
  "request": "changeOwner",
  "response": "OK"
}
```

setName

Set the name of the agent.

```
{
  "section": "agent",
  "request": "setName",
  "name": "cracker1",
  "agentId": 2,
  "accessKey": "mykey"
}
```

```
{
  "section": "agent",
  "request": "setName",
  "response": "OK"
}
```

setCpuOnly

Set if an agent is CPU only or not.

```
{
  "section": "agent",
  "request": "setCpuOnly",
  "cpuOnly": true,
  "agentId": 2,
  "accessKey": "mykey"
}
```

```
{
  "section": "agent",
  "request": "setCpuOnly",
  "response": "OK"
}
```

setExtraParams

Set agent specific command line parameters for the agent which are included in the cracker command line call on the agent.

```
{
  "section": "agent",
  "request": "setExtraParams",
  "extraParameters": "-d 1,2",
  "agentId": 2,
  "accessKey": "mykey"
}
```

```
{
  "section": "agent",
  "request": "setExtraParams",
  "response": "OK"
}
```

setErrorFlag

Set how errors on the agent should be handled on the server. Following values can be given as *ignoreErrors* value:

0 In case of an error, the error message gets saved on the server and the agent will be put into inactive state.

1 In case of an error, the error message gets saved on the server, but the agent will be given further chunks to work on if he requests so.

2 In case of an error, nothing will be saved on the server and the agent can continue to work and will not put into inactive state.

```
{
  "section": "agent",
  "request": "setErrorFlag",
  "ignoreErrors": 0,
  "agentId": 2,
  "accessKey": "mykey"
}
```

```
{
  "section": "agent",
  "request": "setErrorFlag",
  "response": "OK"
}
```

setTrusted

Set if an agent is trusted or not.

```
{
  "section": "agent",
  "request": "setTrusted",
  "trusted": false,
  "agentId": 2,
  "accessKey": "mykey"
}
```

```
{
  "section": "agent",
  "request": "setTrusted",
  "response": "OK"
}
```

listVouchers

Lists all currently existing vouchers on the server which can be used to register new agents.

```
{
  "section": "agent",
  "request": "listVouchers",
  "accessKey": "mykey"
}
```

```
{
  "section": "agent",
  "request": "getBinaries",
  "response": "OK",
  "vouchers": [
    "sM2q6CwiPY",
    "xkw782a3x9",
    "2drg6Vsqor",
    "AZyY8dK1ao"
  ]
}
```

createVoucher

Create a new voucher on the server. It is optional to specify a voucher code otherwise the server will just generate a random one. The server always sends back the created voucher.

```
{
  "section": "agent",
  "request": "createVoucher",
  "voucher": "mySpecificVoucher",
  "accessKey": "mykey"
}
```

```
{
  "section": "agent",
  "request": "createVoucher",
  "accessKey": "mykey"
}
```

```
{
  "section": "agent",
  "request": "createVoucher",
  "response": "OK",
  "voucher": "Gjawgidkr4"
}
```

deleteVoucher

Delete a voucher from the server.

```
{
  "section": "agent",
  "request": "deleteVoucher",
  "voucher": "Gjawgidkr4",
  "accessKey": "mykey"
}
```

```
{
  "section": "agent",
  "request": "deleteVoucher",
  "response": "OK"
}
```

getBinaries

Lists which agent binaries are available on the server to be used for agents.

```
{
  "section": "agent",
  "request": "getBinaries",
  "accessKey": "mykey"
}
```

```
{
  "section": "agent",
  "request": "getBinaries",
  "response": "OK",
   "apiUrl": "http:\/\/localhost\/hashtopolis\/src\/api\/api\/server.php",
  "binaries": [
    {
      "name": "csharp",
      "os": "Windows, Linux(mono), OS X(mono)",
      "url": "http:\/\/localhost\/hashtopolis\/src\/api\/agents.php?download=1",
      "version": "0.52.2",
      "filename": "hashtopolis.exe"
    },
    {
      "name": "python",
      "os": "Windows, Linux, OS X",
      "url": "http:\/\/localhost\/hashtopolis\/src\/api\/agents.php?download=2",
      "version": "0.1.4",
      "filename": "hashtopolis.zip"
    }
  ]
}
```

Tasks (*task*)

Used to access all functions around tasks. Please note that the user must have access to the groups where a task is belonging to, to retrieve it and to be able to apply changes.

listTasks

List all tasks on the server. There are two task types:

0 Normal Task

1 Supertask

```
{
  "section": "task",
  "request": "listTasks",
  "accessKey": "mykey"
}
```

```
{
  "section": "task",
  "request": "listTasks",
  "response": "OK",
  "tasks": [
    {
      "taskId": 7587,
      "name": "test 2",
      "type": 0,
      "hashlistId": 1,
      "priority": 5
    },
    {
      "supertaskId": 33,
      "name": "Increment ?a",
      "type": 1,
      "hashlistId": 1,
      "priority": 3
    },
    {
      "supertaskId": 32,
      "name": "Supertask Test",
      "type": 1,
      "hashlistId": 1,
      "priority": 0
    },
    {
      "taskId": 7580,
      "name": "test 1",
      "type": 0,
      "hashlistId": 1,
      "priority": 0
    }
  ]
}
```

getTask

Get the details for a specific task. Note that this request can only be done with tasks or subtasks, but not with supertasks.

```
{
  "section": "task",
  "request": "getTask",
  "taskId": 7587,
  "accessKey": "mykey"
}
```


```
{
  "section": "task",
  "request": "getTask",
  "response": "OK",
  "taskId": 7587,
  "name": "testing",
  "attack": "#HL# -a 0 top10000.txt -r dive.rule",
  "chunksize": 600,
  "color": null,
  "benchmarkType": "speed",
  "statusTimer": 5,
  "priority": 0,
  "isCpuOnly": false,
  "isSmall": false,
  "skipKeyspace": 0,
  "keyspace": 10000,
  "dispatched": 10000,
  "hashlistId": 1,
  "imageUrl": "http:\/\/localhost\/hashtopolis\/src\/api\/taskimg.php?task=7587",
  "files": [
    {
      "fileId": 2,
      "filename": "dive.rule",
      "size": 887155
    },
    {
      "fileId": 3653,
      "filename": "top10000.txt",
      "size": 76508
    }
  ],
  "speed": 0,
  "searched": 10000,
  "chunkIds": [
    31
  ],
  "agents": [
    {
      "agentId": 2,
      "benchmark": "0",
      "speed": 0
    }
  ],
  "isComplete": false,
  "usePreprocessor": false,
  "preprocessorId": 0,
```

```
  "preprocessorCommand": ""
}
```

listSubtasks

List all subtasks of a given running supertask.

```
{
  "section": "task",
  "request": "listSubtasks",
  "supertaskId": 33,
  "accessKey": "mykey"
}
```

```
{
  "section": "task",
  "request": "listSubtasks",
  "response": "OK",
  "subtasks": [
    {
      "taskId": 7582,
      "name": "?a?a?a",
      "priority": 0
    },
    {
      "taskId": 7583,
      "name": "?a?a?a?a",
      "priority": 0
    },
    {
      "taskId": 7584,
      "name": "?a?a?a?a?a",
      "priority": 0
    },
    {
      "taskId": 7585,
      "name": "?a?a?a?a?a?a",
      "priority": 0
    }
  ]
}
```

getChunk

Get details about a specific chunk. Progress is given in percents, start/length/checkpoint are in relation to the keyspace.

```
{
  "section": "task",
  "request": "getChunk",
  "chunkId": 30,
  "accessKey": "mykey"
}
```

```
{
  "section": "task",
```

```
  "request": "getChunk",
  "response": "OK",
  "chunkId": 30,
  "start": 23141360,
  "length": 5785340,
  "checkpoint": 28926700,
  "progress": 100,
  "taskId": 7585,
  "agentId": 2,
  "dispatchTime": 1531313146,
  "lastActivity": 1531313738,
  "state": 4,
  "cracked": 0,
  "speed": 0
}
```

createTask

Create a new task (one example with files and one without).

```
{
  "section": "task",
  "request": "createTask",
  "name": "API Task",
  "hashlistId": 1,
  "attackCmd": "#HL# -a 0 -r dive.rule example.dict",
  "chunksize": 600,
  "statusTimer": 5,
  "benchmarkType": "speed",
  "color": "5D5D5D",
  "isCpuOnly": false,
  "isSmall": false,
  "skip": 0,
  "crackerVersionId": 2,
  "files": [
    1,
    2
  ],
  "priority": 100,
  "preprocessorId": 0,
  "preprocessorCommand": "",
  "accessKey": "mykey"
}
```

```
{
  "section": "task",
  "request": "createTask",
  "name": "API Task BF",
  "hashlistId": 1,
  "attackCmd": "#HL# -a 3 ?l?l?l?l?l?l?l",
  "chunksize": 600,
  "statusTimer": 5,
  "benchmarkType": "speed",
  "color": "5D5D5D",
  "isCpuOnly": false,
  "isSmall": true,
```

```
  "skip": 0,
  "crackerVersionId": 2,
  "files": [],
  "priority": 99,
  "preprocessorId": 0,
  "preprocessorCommand": "",
  "accessKey": "mykey"
}
```

```
{
  "section": "task",
  "request": "createTask",
  "response": "OK",
  "taskId": 101
}
```

runPretask

Create a task based on a preconfigured task.

```
{
  "section": "task",
  "request": "runPretask",
  "name": "API Run pretask",
  "hashlistId": 1,
  "pretaskId": 1,
  "crackerVersionId": 2,
  "accessKey": "mykey"
}
```

```
{
  "section": "task",
  "request": "runPretask",
  "response": "OK"
}
```

runSupertask

Create a supertask out of a configured preconfigured task collection.

```
{
  "section": "task",
  "request": "runSupertask",
  "hashlistId": 1,
  "supertaskId": 1,
  "crackerVersionId": 2,
  "accessKey": "mykey"
}
```

```
{
  "section": "task",
  "request": "runSupertask",
  "response": "OK"
}
```

setTaskPriority

Set the priority for a task.

```
{
  "section": "task",
  "request": "setTaskPriority",
  "taskId": 7580,
  "priority": 9000,
  "accessKey": "mykey"
}
```

```
{
  "section": "task",
  "request": "setTaskPriority",
  "response": "OK"
}
```

setSupertaskPriority

Set the priority for a supertask.

```
{
  "section": "task",
  "request": "setSupertaskPriority",
  "supertaskId": 42,
  "supertaskPriority": 9000,
  "accessKey": "mykey"
}
```

```
{
  "section": "task",
  "request": "setSupertaskPriority",
  "response": "OK"
}
```

setTaskName

Set the name for a task.

```
{
  "section": "task",
  "request": "setTaskName",
  "taskId": 7580,
  "name": "New Task Name",
  "accessKey": "mykey"
}
```

```
{
  "section": "task",
  "request": "setTaskName",
  "response": "OK"
}
```

setTaskColor

Set the color of a task.

```
{
  "section": "task",
  "request": "setTaskColor",
  "taskId": 7580,
  "color": "78ABCD",
  "accessKey": "mykey"
}
```

```
{
  "section": "task",
  "request": "setTaskColor",
  "response": "OK"
}
```

setTaskCpuOnly

Set if a task is a CPU only task or not.

```
{
  "section": "task",
  "request": "setTaskCpuOnly",
  "taskId": 7580,
  "isCpuOnly": false,
  "accessKey": "mykey"
}
```

```
{
  "section": "task",
  "request": "setTaskCpuOnly",
  "response": "OK"
}
```

setTaskSmall

Set if a task is small or not.

```
{
  "section": "task",
  "request": "setTaskSmall",
  "taskId": 7580,
  "isSmall": false,
  "accessKey": "mykey"
}
```

```
{
  "section": "task",
  "request": "setTaskSmall",
  "response": "OK"
}
```

taskUnassignAgent

Unassign an agent from his task.

```
{
  "section": "task",
  "request": "taskUnassignAgent",
  "agentId": 2,
  "accessKey": "mykey"
}
```

```
{
  "section": "task",
  "request": "taskUnassignAgent",
  "response": "OK"
}
```

taskUnassignAgent

Assign an agent to this task. Note that the agent might be re-assigned to another task if there is one with a higher priority.

```
{
  "section": "task",
  "request": "taskAssignAgent",
  "agentId": 2,
  "taskId": 5,
  "accessKey": "mykey"
}
```

```
{
  "section": "task",
  "request": "taskAssignAgent",
  "response": "OK"
}
```

deleteTask

Completely delete a task.

```
{
  "section": "task",
  "request": "deleteTask",
  "taskId": 7580,
  "accessKey": "mykey"
}
```

```
{
  "section": "task",
  "request": "deleteTask",
  "response": "OK"
}
```

purgeTask

Purge all task data and reset it to initial state.

```
{
  "section": "task",
  "request": "purgeTask",
  "taskId": 7591,
  "accessKey": "mykey"
}
```

```
{
  "section": "task",
  "request": "purgeTask",
  "response": "OK"
}
```

setSupertaskName

Set the name of a running supertask.

```
{
  "section": "task",
  "request": "setSupertaskName",
  "supertaskId": 43,
  "name": "New Supertask Name",
  "accessKey": "mykey"
}
```

```
{
  "section": "task",
  "request": "setSupertaskName",
  "response": "OK"
}
```

deleteSupertask

Delete a running supertask. This includes all contained subtasks.

```
{
  "section": "task",
  "request": "deleteSupertask",
  "supertaskId": 43,
  "accessKey": "mykey"
}
```

```
{
  "section": "task",
  "request": "deleteSupertask",
  "response": "OK"
}
```

archiveTask

Archive a task.

```
{
  "section": "task",
  "request": "archiveTask",
  "taskId": 7601,
  "accessKey": "mykey"
}
```

```
{
  "section": "task",
  "request": "archiveTask",
  "response": "OK"
}
```

archiveSupertask

Archive a supertask (including all subtasks).

```
{
  "section": "task",
  "request": "archiveSupertask",
  "supertaskId": 54,
  "accessKey": "mykey"
}
```

```
{
  "section": "task",
  "request": "archiveSupertask",
  "response": "OK"
}
```

getCracked

Retrieve all cracked hashes by a given task.

```
{
  "section": "task",
  "request": "getCracked",
  "taskId": 100,
  "accessKey": "mykey"
}
```

```
{
  "section": "task",
  "request": "getCracked",
  "response": "OK",
  "cracked": [
    {
      "hash": "098f6bcd4621d373cade4e832627b4f6",
      "plain": "test",
      "crackpos": "634721"
    },
    {
```

```
      "hash": "5f4dcc3b5aa765d61d8327deb882cf99",
      "plain": "password",
      "crackpos": "608529"
    }
  ]
}
```

Preconfigured Tasks (*pretask*)

Used to access all functions around preconfigured tasks.

setPretaskPriority

Set the priority of a preconfigured task.

```
{
  "section": "pretask",
  "request": "setPretaskPriority",
  "pretaskId": 1,
  "priority": 100,
  "accessKey": "mykey"
}

{
  "section": "pretask",
  "request": "setPretaskPriority",
  "response": "OK"
}
```

setPretaskName

Rename a preconfigured task.

```
{
  "section": "pretask",
  "request": "setPretaskName",
  "pretaskId": 1,
  "name": "New name",
  "accessKey": "mykey"
}

{
  "section": "pretask",
  "request": "setPretaskName",
  "response": "OK"
}
```

setPretaskColor

Set the color of a preconfigured task.

```
{
  "section": "pretask",
  "request": "setPretaskColor",
  "pretaskId": 1,
  "color": "FF00FF",
  "accessKey": "mykey"
}

{
  "section": "pretask",
  "request": "setPretaskColor",
  "response": "OK"
}
```

setPretaskChunksize

Set the chunk time for a preconfigured task.

```
{
  "section": "pretask",
  "request": "setPretaskChunksize",
  "pretaskId": 1,
  "chunksize": 300,
  "accessKey": "mykey"
}
```

```
{
  "section": "pretask",
  "request": "setPretaskChunksize",
  "response": "OK"
}
```

setPretaskCpuOnly

Set if a preconfigured task is a CPU only task or not.

```
{
  "section": "pretask",
  "request": "setPretaskCpuOnly",
  "pretaskId": 1,
  "isCpuOnly": true,
  "accessKey": "mykey"
}
```

```
{
  "section": "pretask",
  "request": "setPretaskCpuOnly",
  "response": "OK"
}
```

setPretaskSmall

Set if a preconfigured task is small or not.

```
{
  "section": "pretask",
  "request": "setPretaskSmall",
  "pretaskId": 1,
  "isSmall": true,
  "accessKey": "mykey"
}
```

```
{
  "section": "pretask",
  "request": "setPretaskSmall",
  "response": "OK"
}
```

deletePretask

Deletes a preconfigured task. This also includes removing it from supertasks if it is used there.

```
{
  "section": "pretask",
  "request": "deletePretask",
  "pretaskId": 1,
  "accessKey": "mykey"
}
```

```
{
  "section": "pretask",
  "request": "deletePretask",
  "response": "OK"
}
```

createPretask

Create a new preconfigured task

```
{
  "section": "pretask",
  "request": "createPretask",
  "name": "API Pretask",
  "attackCmd": "#HL# -a 0 example.dict",
  "chunksize": 600,
  "statusTimer": 5,
  "benchmarkType": "speed",
  "color": "",
  "isCpuOnly": false,
  "isSmall": true,
  "priority": 9000,
  "files": [
    1
  ],
  "crackerTypeId": 1,
  "accessKey": "mykey"
}
```

```
{
  "section": "pretask",
  "request": "createPretask",
  "response": "OK"
}
```

listPretasks

Lists all preconfigured tasks available on the server.

```
{
  "section": "pretask",
  "request": "listPretasks",
  "accessKey": "mykey"
}
```

```
{
  "section": "pretask",
  "request": "listPretasks",
  "response": "OK",
  "pretasks": [
    {
      "pretaskId": 1,
      "name": "Test Pre",
      "priority": 0
    },
    {
      "pretaskId": 7,
      "name": "Brute ?a 8",
      "priority": 0
    }
  ]
}
```


getPretask

Get complete information about a preconfigured task.


```
{
  "section": "pretask",
  "request": "getPretask",
  "pretaskId": 7,
  "accessKey": "mykey"
}
```



```
{
  "section": "pretask",
  "request": "getPretask",
  "response": "OK",
  "pretaskId": 7,
  "name": "Brute ?a 8",
  "attackCmd": "#HL# -a 3 ?a?a?a?a?a?a?a?a",
  "chunksize": 600,
  "color": null,
  "benchmarkType": "runtime",
  "statusTimer": 5,
  "priority": 0,
  "isCpuOnly": false,
  "isSmall": false,
  "files": []
}
```

Supertasks (*supertask*)

Used to access all functions around prepared supertasks.

importSupertask

Create a supertask configuration with a given list of masks.

```
{
  "section": "supertask",
  "request": "importSupertask",
  "name": "Mask Supertask",
  "isCpuOnly": false,
  "isSmall": false,
  "masks": [
    "?d?d?d?d",
    "?l?d?d?d?d",
    "?d?d?d?d?d",
    "?u?d?d?d?d"
  ],
  "optimizedFlag": true,
  "crackerTypeId": 1,
  "benchtype": "speed",
  "accessKey": "mykey"
}
```

```
{
  "section": "supertask",
  "request": "importSupertask",
  "response": "OK"
}
```

listSupertasks

Lists the available supertasks on the server which group preconfigured tasks together.

```
{
  "section": "supertask",
  "request": "listSupertasks",
  "accessKey": "mykey"
}
```

```
{
  "section": "supertask",
  "request": "listSupertasks",
  "response": "OK",
  "supertasks": [
    {
      "supertaskId": 1,
      "name": "Supertask Test"
    },
    {
      "supertaskId": 2,
      "name": "Increment ?a"
    }
  ]
}
```

getSupertask

Get detail information of a supertask.

```
{
  "section": "supertask",
  "request": "getSupertask",
  "supertaskId": 2,
  "accessKey": "mykey"
}
```

```
{
  "section": "supertask",
  "request": "getSupertask",
  "response": "OK",
  "supertaskId": 2,
  "name": "Increment ?a",
  "pretasks": [
    {
      "pretaskId": 2,
      "name": "?a?a?a",
      "priority": 6
    },
    {
      "pretaskId": 3,
      "name": "?a?a?a?a",
      "priority": 5
    },
    {
      "pretaskId": 4,
      "name": "?a?a?a?a?a",
      "priority": 4
    },
    {
      "pretaskId": 5,
      "name": "?a?a?a?a?a?a",
      "priority": 3
    },
    {
      "pretaskId": 6,
      "name": "?a?a?a?a?a?a?a",
      "priority": 2
    },
    {
      "pretaskId": 1,
      "name": "Test Pre",
      "priority": 0
    }
  ]
}
```

createSupertask

Create a new supertask out of existing preconfigured tasks.

```
{
  "section": "supertask",
```

```
  "request": "createSupertask",
  "name": "Mixed Supertask",
  "pretasks": [
    7,
    8
  ],
  "accessKey": "mykey"
}
```

```
{
  "section": "supertask",
  "request": "createSupertask",
  "response": "OK"
}
```

setSupertaskName

Rename a supertask to a new name.

```
{
  "section": "supertask",
  "request": "setSupertaskName",
  "supertaskId": 4,
  "name": "Other Name",
  "accessKey": "mykey"
}
```

```
{
  "section": "supertask",
  "request": "setSupertaskName",
  "response": "OK"
}
```

deleteSupertask

Delete a supertask.

```
{
  "section": "supertask",
  "request": "deleteSupertask",
  "supertaskId": 4,
  "accessKey": "mykey"
}
```

```
{
  "section": "supertask",
  "request": "deleteSupertask",
  "response": "OK"
}
```

bulkSupertask

Create a supertask with a base command and replace the FILE placeholder with every file it should iterate over.

```
{
  "section": "supertask",
  "request": "bulkSupertask",
  "name": "User API Bulk Test",
  "isCpuOnly": false,
  "isSmall": false,
  "crackerTypeId": 1,
  "benchmarkType": "speed",
  "attackCmd": "#HL# example.dict -r FILE",
  "basefiles": [
    1
  ],
  "iterfiles": [
    2,
    3
  ],
  "accessKey": "mykey"
}
```

```
{
  "section": "supertask",
  "request": "bulkSupertask",
  "response": "OK"
}
```

Hashlists (*hashlist*)

Used to access all functions around hashlists.

listsHashlists

List all hashlists (excluding superhashlists);

```
{
  "section": "hashlist",
  "request": "listHashlists",
  "accessKey": "mykey"
}
```

```
{
  "section": "hashlist",
  "request": "listHashlists",
  "response": "OK",
  "hashlists": [
    {
      "hashlistId": 1,
      "hashtypeId": 0,
      "name": "Hashcat Example",
      "format": 0,
      "hashCount": 6494
    },
    {
      "hashlistId": 3,
      "hashtypeId": 14800,
      "name": "iTunes test for splitting",
      "format": 0,
      "hashCount": 1
    },
    {
      "hashlistId": 4,
      "hashtypeId": 6242,
      "name": "truecrypt test",
      "format": 2,
      "hashCount": 1
    }
  ]
}
```

getHashlist

Get information about a specific hashlist.

```
{
  "section": "hashlist",
  "request": "getHashlist",
  "hashlistId": 1,
  "accessKey": "mykey"
}
```

```
{
  "section": "hashlist",
```

```
  "request": "getHashlist",
  "response": "OK",
  "hashlistId": 1,
  "hashtypeId": 0,
  "name": "Hashcat Example",
  "format": 0,
  "hashCount": 6494,
  "cracked": 3382,
  "accessGroupId": 1,
  "isHexSalt": false,
  "isSalted": false,
  "isSecret": false,
  "saltSeparator": ":",
  "notes": "This hashlist is from blahblah...",
  "useBrain": false
}
```


createHashlist

Create a new hashlist. Please note that it is not ideal to create large hashlists with the API as you have to send the full data. The hashlist data should always be base64 (using UTF-8) encoded. Hashcat brain can only be used if it is activated in the server config.


```
{
  "section": "hashlist",
  "request": "createHashlist",
  "name": "API Hashlist",
  "isSalted": false,
  "isSecret": true,
  "isHexSalt": false,
  "separator": ":",
  "format": 0,
  "hashtypeId": 3200,
  "accessGroupId": 1,
  "data": "JDJ5JDEyJDcwMElMNlZ4TGwyLkEvS2NISmJEYmVKMGFhcWVxYUdrcHhlc0FFZC5jWFBQUU4vWjNVN1c2",
  "useBrain": false,
  "brainFeatures": 0,
  "accessKey": "mykey"
}
```

```
{
  "section": "hashlist",
  "request": "createHashlist",
  "response": "OK",
  "hashlistId": 101
}
```


setHashlistName

Set the name of a hashlist.


```
{
  "section": "hashlist",
  "request": "setHashlistName",
  "name": "BCRYPT easy",
  "hashlistId": 5,
```

```
  "accessKey": "mykey"
}
```

```
{
  "section": "hashlist",
  "request": "setHashlistName",
  "response": "OK"
}
```

setSecret

Set if a hashlist is secret or not.

```
{
  "section": "hashlist",
  "request": "setSecret",
  "isSecret": false,
  "hashlistId": 5,
  "accessKey": "mykey"
}
```

```
{
  "section": "hashlist",
  "request": "setSecret",
  "response": "OK"
}
```

importCracked

Add some cracked hashes from an external source for this hashlist. The data must be base64 (using UTF-8) encoded.

```
{
  "section": "hashlist",
  "request": "importCracked",
  "hashlistId": 5,
  "separator": ":",
  "data": "JDJ5JDEyJDcwMElMNlZ4TGwyLkEvS2NISmJEYmVKMGFhcWVxYUdrcHhlc0FFZC5jWFBQUU4vWjNVN1c2OnRlc3Q=",
  "accessKey": "mykey"
}
```

```
{
  "section": "hashlist",
  "request": "importCracked",
  "response": "OK",
  "linesProcessed": 1,
  "newCracked": 1,
  "alreadyCracked": 0,
  "invalidLines": 0,
  "notFound": 0,
  "processTime": 0,
  "tooLongPlains": 0
}
```

exportCracked

Exports the cracked hashes in hash:plain format to a new file. The response includes the informations about the created file.

```
{
  "section": "hashlist",
  "request": "exportCracked",
  "hashlistId": 5,
  "accessKey": "mykey"
}

{
  "section": "hashlist",
  "request": "exportCracked",
  "response": "OK",
  "fileId": 7567,
  "filename": "Pre-cracked_5_19-07-2018_14-45-52.txt"
}
```

generateWordlist

Generates a wordlist of all plaintexts of the cracked hashes of this hashlist. The response includes the informations about the created file.

```
{
  "section": "hashlist",
  "request": "generateWordlist",
  "hashlistId": 5,
  "accessKey": "mykey"
}

{
  "section": "hashlist",
  "request": "generateWordlist",
  "response": "OK",
  "fileId": 7568,
  "filename": "Wordlist_5_19.07.2018_14.47.20.txt"
}
```

exportLeft

Generates a left list with all hashes which are not cracked. The response returns informations about the created file. This only works for plaintext hashlists!

```
{
  "section": "hashlist",
  "request": "exportLeft",
  "hashlistId": 1,
  "accessKey": "mykey"
}

{
  "section": "hashlist",
  "request": "exportLeft",
  "response": "OK",
  "fileId": 7569,
  "filename": "Leftlist_1_19-07-2018_14-49-02.txt"
}
```

deleteHashlist

Delete a hashlist and all according hashes. This will remove a hashlist from the superhashlists it is member of.

```
{
  "section": "hashlist",
  "request": "deleteHashlist",
  "hashlistId": 5,
  "accessKey": "mykey"
}
```

```
{
  "section": "hashlist",
  "request": "deleteHashlist",
  "response": "OK"
}
```

getHash

Search if a hash is found on the server. This searches on all hashlists which the user has access to.

```
{
  "section": "hashlist",
  "request": "getHash",
  "hash": "0021ca52049c734ac0d3d6f92042abf7",
  "accessKey": "mykey"
}
```

```
{
  "section": "hashlist",
  "request": "getHash",
  "response": "ERROR",
  "message": "Hash was not found or is not cracked!"
}
```

```
{
  "section": "hashlist",
  "request": "getHash",
  "hash": "00428d94d9482d8c7037b6865521b3fd",
  "accessKey": "mykey"
}
```

```
{
  "section": "hashlist",
  "request": "getHash",
  "response": "OK",
  "hash": "00428d94d9482d8c7037b6865521b3fd",
  "crackpos": 12467,
  "plain": "wellgetthem"
}
```

getCracked

Retrieve all cracked hashes of a given hashlist.

```
{
  "section": "hashlist",
  "request": "getCracked",
  "hashlistId": "1",
  "accessKey": "mykey"
}
```

```
{
  "section": "hashlist",
  "request": "getCracked",
  "response": "OK",
  "cracked": [
    {
      "hash": "098f6bcd4621d373cade4e832627b4f6",
      "plain": "test",
      "crackpos": "634721"
    },
    {
      "hash": "5f4dcc3b5aa765d61d8327deb882cf99",
      "plain": "password",
      "crackpos": "608529"
    }
  ]
}
```

Superhashlists (*superhashlist*)

Used to access all functions around superhashlists.

listSuperhashlists

List all superhashlists on the server.

```
{
  "section": "superhashlist",
  "request": "listSuperhashlists",
  "accessKey": "mykey"
}
```

```
{
  "section": "superhashlist",
  "request": "listSuperhashlists",
  "response": "OK",
  "superhashlists": [
    {
      "hashlistId": 2,
      "hashtypeId": 0,
      "name": "Test superhashlist",
      "hashCount": 6494
    }
  ]
}
```

getSuperhashlist

Get detailed information about a superhashlist.

```
{
  "section": "superhashlist",
  "request": "getSuperhashlist",
  "superhashlistId": 2,
  "accessKey": "mykey"
}
```

```
{
  "section": "superhashlist",
  "request": "getSuperhashlist",
  "response": "OK",
  "hashlistId": 2,
  "hashtypeId": 0,
  "name": "Test superhashlist",
  "hashCount": 6494,
  "cracked": 80,
  "accessGroupId": 1,
  "isSecret": false,
  "hashlists": [
    1
  ]
}
```

createSuperhashlist

Create a new superhashlist out of existing hashlists.

```
{
  "section": "superhashlist",
  "request": "createSuperhashlist",
  "name": "New Superhashlist",
  "hashlists": [
    1,
    2
  ],
  "accessKey": "mykey"
}
```

```
{
  "section": "superhashlist",
  "request": "createSuperhashlist",
  "response": "OK"
}
```

deleteSuperhashlist

Deletes a superhashlist. But the containing hashlists will not be removed.

```
{
  "section": "superhashlist",
  "request": "deleteSuperhashlist",
  "superhashlistId": 6,
  "accessKey": "mykey"
}
```

```
{
  "section": "superhashlist",
  "request": "deleteSuperhashlist",
  "response": "OK"
}
```

Files (*file*)

Used to access all functions around files. There are currently following file types available:

0 Wordlist

1 Rule

listFiles

List all available files.

```
{
  "section": "file",
  "request": "listFiles",
  "accessKey": "mykey"
}
```

```
{
  "section": "file",
  "request": "listFiles",
  "response": "OK",
  "files": [
    {
      "fileId": 1,
      "fileType": 0,
      "filename": "example.dict"
    },
    {
      "fileId": 2,
      "fileType": 1,
      "filename": "dive.rule"
    },
    {
      "fileId": 3,
      "fileType": 1,
      "filename": "generated.rule"
    },
    {
      "fileId": 3653,
      "fileType": 0,
      "filename": "top10000.txt"
    },
    {
      "fileId": 3654,
      "fileType": 1,
      "filename": "toggles4.rule"
    }
  ]
}
```

getFile

Get detailed informations of a file and also get a link to download it.

```
{
  "section": "file",
```

```
  "request": "getFile",
  "fileId": 1,
  "accessKey": "mykey"
}
```

```
{
  "section": "file",
  "request": "getFile",
  "response": "OK",
  "fileId": 1,
  "fileType": 0,
  "filename": "example.dict",
  "isSecret": true,
  "size": 1080240,
  "url": "getFile.php?file=1&apiKey=mykey"
}
```

renameFile

Rename an existing file. ATTENTION: this can affect tasks using this file! It is only recommended to change the name when the file is not used by any task!

```
{
  "section": "file",
  "request": "renameFile",
  "fileId": 1,
  "filename": "example.txt",
  "accessKey": "mykey"
}
```

```
{
  "section": "file",
  "request": "renameFile",
  "response": "OK"
}
```

setSecret

Set if an existing file is secret or not.

```
{
  "section": "file",
  "request": "setSecret",
  "fileId": 1,
  "isSecret": false,
  "accessKey": "mykey"
}
```

```
{
  "section": "file",
  "request": "setSecret",
  "response": "OK"
}
```

deleteFile

Deletes a file from the server. This is only possible if the file is not used in any task.

```
{
  "section": "file",
  "request": "deleteFile",
  "fileId": 3654,
  "accessKey": "mykey"
}
```

```
{
  "section": "file",
  "request": "deleteFile",
  "response": "OK"
}
```

setFileType

Switch the file type of an existing file.

```
{
  "section": "file",
  "request": "setFileType",
  "fileId": 1,
  "fileType": 1,
  "accessKey": "mykey"
}
```

```
{
  "section": "file",
  "request": "setFileType",
  "response": "OK"
}
```

addFile

There are multiple ways to add a file, either from an URL, from the import directory or inline. The filename is only relevant if it is added inline. Otherwise it will take the original filename. In case of the inline upload, the data must be base64 encoded (using UTF-8).

```
{
  "section": "file",
  "request": "addFile",
  "filename": "api_test_inline.txt",
  "fileType": 0,
  "source": "inline",
  "accessGroupId": 1,
  "data": "MTIzNA0KNTY3OA0KcGFzc3dvcmQNCmFiYw==",
  "accessKey": "mykey"
}
```

```
{
  "section": "file",
  "request": "addFile",
  "filename": "doesnt-matter.txt",
```

```
  "fileType": 0,
  "source": "import",
  "accessGroupId": 1,
  "data": "otherlist.txt",
  "accessKey": "mykey"
}


{
  "section": "file",
  "request": "addFile",
  "filename": "doesnt-matter.txt",
  "fileType": 1,
  "source": "url",
  "accessGroupId": 1,
  "data": "https://github.com/hashcat/hashcat/raw/master/rules/best64.rule",
  "accessKey": "mykey"
}


{
  "section": "file",
  "request": "addFile",
  "response": "OK"
}
```

Crackers (*cracker*)

Used to access all functions around crackers and their versions.

listCrackers

Lists all different types of crackers available.

```
{
  "section": "cracker",
  "request": "listCrackers",
  "accessKey": "mykey"
}
```

```
{
  "section": "cracker",
  "request": "listCrackers",
  "response": "OK",
  "crackers": [
    {
      "crackerTypeId": 1,
      "crackerTypeName": "hashcat"
    }
  ]
}
```

getCracker

Get detailed informations of cracker, especially all available versions.

```
{
  "section": "cracker",
  "request": "getCracker",
  "crackerTypeId": 1,
  "accessKey": "mykey"
}
```

```
{
  "section": "cracker",
  "request": "getCracker",
  "response": "OK",
  "crackerTypeId": 1,
  "crackerTypeName": "hashcat",
  "crackerVersions": [
    {
      "versionId": 1,
      "version": "4.1.0",
      "downloadUrl": "https:\/\/hashcat.net\/files\/hashcat-4.1.0.7z",
      "binaryBasename": "hashcat"
    },
    {
      "versionId": 3,
      "version": "4.1.1",
      "downloadUrl": "https:\/\/hashcat.net\/beta\/hashcat-4.1.1-15.7z",
      "binaryBasename": "hashcat"
    }
  ]
}
```

deleteCracker

Deletes a complete cracker type including all versions configured. This is only possible if the type is not used in any of the preconfigured/normal tasks.

```
{
  "section": "cracker",
  "request": "deleteCracker",
  "crackerTypeId": 1,
  "accessKey": "mykey"
}
```

```
{
  "section": "cracker",
  "request": "deleteCracker",
  "response": "OK"
}
```

deleteVersion

Deletes a specific cracker version. This is only possible if the type is not used in any of the tasks.

```
{
  "section": "cracker",
  "request": "deleteVersion",
  "crackerVersionId": 3,
  "accessKey": "mykey"
}
```

```
{
  "section": "cracker",
  "request": "deleteVersion",
  "response": "OK"
}
```

createCracker

Creates a new cracker type.

```
{
  "section": "cracker",
  "request": "createCracker",
  "crackerName": "My Generic Cracker",
  "accessKey": "mykey"
}
```

```
{
  "section": "cracker",
  "request": "createCracker",
  "response": "OK"
}
```

addVersion

Add a new version to an existing cracker type.

```
{
  "section": "cracker",
  "request": "addVersion",
  "crackerTypeId": 2,
  "crackerBinaryVersion": "1.0.0",
  "crackerBinaryBasename": "cracker",
  "crackerBinaryUrl": "https://example.org/download.7z",
  "accessKey": "mykey"
}
```

```
{
  "section": "cracker",
  "request": "addVersion",
  "response": "OK"
}
```

updateVersion

Update the data for an existing cracker version. All values need to be provided, but they do not have to change all.

```
{
  "section": "cracker",
  "request": "updateVersion",
  "crackerVersionId": 4,
  "crackerBinaryVersion": "1.0.0",
  "crackerBinaryBasename": "cracker",
  "crackerBinaryUrl": "https://example.org/archive/download.7z",
  "accessKey": "mykey"
}
```

```
{
  "section": "cracker",
  "request": "updateVersion",
  "response": "OK"
}
```

Server Config (*config*)

Used to access server configuration.

listSections

Lists the available config sections.

```
{
  "section": "config",
  "request": "listSections",
  "accessKey": "mykey"
}
```

```
{
  "section": "config",
  "request": "listSections",
  "response": "OK",
  "configSections": [
    {
      "configSectionId": 1,
      "name": "Cracking\/Tasks"
    },
    {
      "configSectionId": 2,
      "name": "Yubikey"
    },
    {
      "configSectionId": 3,
      "name": "Finetuning"
    },
    {
      "configSectionId": 4,
      "name": "UI"
    },
    {
      "configSectionId": 5,
      "name": "Server"
    }
  ]
}
```

listConfig

List all currently known config values.

```
{
  "section": "config",
  "request": "listConfig",
  "accessKey": "mykey"
}
```

```
{
  "section": "config",
  "request": "listConfig",
  "response": "OK",
```

```
  "items": [
    {
      "item": "agenttimeout",
      "configSectionId": "1",
      "itemDescription": "Time in seconds the server will consider a client inactive or timed out."
    },
            ...
            ...
    {
      "item": "benchtime",
      "configSectionId": "1",
      "itemDescription": "Time in seconds an agent should benchmark a task"
    }
  ]
}
```

getConfig

Get the type and specific value of a config item. The following config types exist:

string basically everything is accepted as string

email similar to string, except that it's tested if it is a valid email

number all kind of numeric value

checkbox boolean value (true or false)

```
{
  "section": "config",
  "request": "getConfig",
  "configItem": "hashlistAlias",
  "accessKey": "mykey"
}
```

```
{
  "section": "config",
  "request": "getConfig",
  "response": "OK",
  "item": "hashlistAlias",
  "configType": "string",
  "value": "#HL#"
}
```

setConfig

Update a value for a config item. Force only needs to be set to true if a new item should be created. Otherwise it will throw an error when it detects an unknown item.

```
{
  "section": "config",
  "request": "setConfig",
  "configItem": "contact",
  "value": "test@example.org",
  "force": false,
  "accessKey": "mykey"
}
```

```
{
  "section": "config",
  "request": "setConfig",
  "response": "ERROR",
  "message": "Unknown config item!"
}
```

```
{
  "section": "config",
  "request": "setConfig",
  "configItem": "contactEmail",
  "value": "test@example.org",
  "force": false,
  "accessKey": "mykey"
}
```

```
{
  "section": "config",
  "request": "setConfig",
  "response": "OK"
}
```

User Management (*user*)

Used to access user management.

listUsers

List all users on the server.

```
{
  "section": "user",
  "request": "listUsers",
  "accessKey": "mykey"
}
```

```
{
  "section": "user",
  "request": "listUsers",
  "response": "OK",
  "users": [
    {
      "userId": 1,
      "username": "htp"
    },
    {
      "userId": 2,
      "username": "testuser"
    }
  ]
}
```

getUser

Get detail information about a user account.

```
{
  "section": "user",
  "request": "getUser",
  "userId": 2,
  "accessKey": "mykey"
}
```

```
{
  "section": "user",
  "request": "getUser",
  "response": "OK",
  "userId": 2,
  "username": "testuser",
  "email": "place@hold.er",
  "rightGroupId": 6,
  "registered": 1530788681,
  "lastLogin": 1531740557,
  "isValid": true,
  "sessionLifetime": 3600
}
```

createUser

Create a new user account (Please note that the server will try to send an email to this address and an error might occur when no email sending is configured!).

```
{
  "section": "user",
  "request": "createUser",
  "username": "blabla",
  "email": "admin@example.org",
  "rightGroupId": 6,
  "accessKey": "mykey"
}
```

```
{
  "section": "user",
  "request": "createUser",
  "response": "OK"
}
```

disableUser

Disable an account, this will throw out the user of all open session and block any further login requests.

```
{
  "section": "user",
  "request": "disableUser",
  "userId": 3,
  "accessKey": "mykey"
}
```

```
{
  "section": "user",
  "request": "disableUser",
  "response": "OK"
}
```

enableUser

Enables a user account so it can log in.

```
{
  "section": "user",
  "request": "enableUser",
  "userId": 3,
  "accessKey": "mykey"
}
```

```
{
  "section": "user",
  "request": "enableUser",
  "response": "OK"
}
```

setUserPassword

Set a new password for the user. This will not affect open sessions, only new logins.

```
{
  "section": "user",
  "request": "setUserPassword",
  "userId": 3,
  "password": "password123",
  "accessKey": "mykey"
}
```

```
{
  "section": "user",
  "request": "setUserPassword",
  "response": "OK"
}
```

setUserRightGroup

Assigns the user to the specified right group regarding what it is allowed to access.

```
{
  "section": "user",
  "request": "setUserRightGroup",
  "userId": 3,
  "rightGroupId": 6,
  "accessKey": "mykey"
}
```

```
{
  "section": "user",
  "request": "setUserRightGroup",
  "response": "OK"
}
```

Group Management (*group*)

Used to access group management.

listGroups

List all existing groups.

```
{
  "section": "group",
  "request": "listGroups",
  "accessKey": "mykey"
}
```

```
{
  "section": "group",
  "request": "listGroups",
  "response": "OK",
  "groups": [
    {
      "groupId": 1,
      "name": "Default Group"
    }
  ]
}
```

getGroup

Get details about a group and its members.

```
{
  "section": "group",
  "request": "getGroup",
  "groupId": 1,
  "accessKey": "mykey"
}
```

```
{
  "section": "group",
  "request": "getGroup",
  "response": "OK",
  "groupId": 1,
  "name": "Default Group",
  "users": [
    1,
    2,
    3
  ],
  "agents": [
    2,
    3
  ]
}
```

createGroup

Create a new empty group.

```
{
  "section": "group",
  "request": "createGroup",
  "name": "Secret Group",
  "accessKey": "mykey"
}
```

```
{
  "section": "group",
  "request": "createGroup",
  "response": "OK"
}
```

addAgent

Add agent as a member of a group.

```
{
  "section": "group",
  "request": "addAgent",
  "groupId": 3,
  "agentId": 2,
  "accessKey": "mykey"
}
```

```
{
  "section": "group",
  "request": "addAgent",
  "response": "OK"
}
```

removeAgent

Remove an agent as member of a group.

```
{
  "section": "group",
  "request": "removeAgent",
  "groupId": 3,
  "agentId": 2,
  "accessKey": "mykey"
}
```

```
{
  "section": "group",
  "request": "removeAgent",
  "response": "OK"
}
```

addUser

Add a user as member of a group.

```
{
  "section": "group",
  "request": "addUser",
  "groupId": 3,
  "userId": 1,
  "accessKey": "mykey"
}
```

```
{
  "section": "group",
  "request": "addUser",
  "response": "OK"
}
```

removeUser

Remove a user as member of a group.

```
{
  "section": "group",
  "request": "removeUser",
  "groupId": 3,
  "userId": 1,
  "accessKey": "mykey"
}
```

```
{
  "section": "group",
  "request": "removeUser",
  "response": "OK"
}
```

deleteGroup

Deletes a group and its associations to users and agents. Please note that all hashlists/tasks belonging to this group will be assigned to the default group.

```
{
  "section": "group",
  "request": "deleteGroup",
  "groupId": 3,
  "accessKey": "mykey"
}
```

```
{
  "section": "group",
  "request": "deleteGroup",
  "response": "OK"
}
```

Access Management (*access*)

Used to manage permissions for user groups.

listGroups

List all user groups.

```
{
  "section": "access",
  "request": "listGroups",
  "accessKey": "mykey"
}
```

```
{
  "section": "access",
  "request": "listGroups",
  "response": "OK",
  "rightGroups": [
    {
      "rightGroupId": 1,
      "name": "Administrator"
    },
    {
      "rightGroupId": 6,
      "name": "Testing Group"
    }
  ]
}
```

getGroup

Get detailed information about a group. If it is the administrators group, the permissions will just be set to *ALL*.

```
{
  "section": "access",
  "request": "getGroup",
  "rightGroupId": 6,
  "accessKey": "mykey"
}
```

```
{
  "section": "access",
  "request": "getGroup",
  "response": "OK",
  "rightGroupId": 6,
  "name": "Testing Group",
  "permissions": {
    "viewHashlistAccess": false,
    "manageHashlistAccess": false,
    "createHashlistAccess": false,
    "createSuperhashlistAccess": false,
    "viewHashesAccess": false,
    "viewAgentsAccess": false,
    "manageAgentAccess": false,
    "createAgentAccess": false,
```

```
    "viewTaskAccess": false,
    "runTaskAccess": true,
    "createTaskAccess": false,
    "manageTaskAccess": false,
    "viewPretaskAccess": true,
    "createPretaskAccess": false,
    "managePretaskAccess": false,
    "viewSupertaskAccess": false,
    "createSupertaskAccess": false,
    "manageSupertaskAccess": false,
    "viewFileAccess": false,
    "manageFileAccess": false,
    "addFileAccess": false,
    "crackerBinaryAccess": false,
    "serverConfigAccess": false,
    "userConfigAccess": false
  },
  "members": [
    2,
    3
  ]
}
```

createGroup

Creates a new group.

```
{
  "section": "access",
  "request": "createGroup",
  "name": "read only users",
  "accessKey": "mykey"
}
```

```
{
  "section": "access",
  "request": "createGroup",
  "response": "OK"
}
```

deleteGroup

Delete a group. This is only possible if no user is associated with this group.

```
{
  "section": "access",
  "request": "deleteGroup",
  "rightGroupId": 7,
  "accessKey": "mykey"
}
```

```
{
  "section": "access",
  "request": "deleteGroup",
  "response": "OK"
}
```

setPermissions

Change some permissions on this group. Note that all the permissions which are not included in the request will be set to false.

```
{
  "section": "access",
  "request": "setPermissions",
  "rightGroupId": 6,
  "permissions": {
    "viewHashlistAccess": true
  },
  "accessKey": "mykey"
}
```

```
{
  "section": "access",
  "request": "setPermissions",
  "response": "OK"
}
```

Account (*account*)

Manage the own account, associated with the API key.

getInformation

Get general information about the account.

```
{
  "section": "account",
  "request": "getInformation",
  "accessKey": "mykey"
}
```

```
{
  "section": "account",
  "request": "getInformation",
  "response": "OK",
  "userId": 1,
  "email": "htp@htp.htp",
  "rightGroupId": 1,
  "sessionLength": 36000
}
```

setEmail

Update the email of the user.

```
{
  "section": "account",
  "request": "setEmail",
  "email": "place@hold.er",
  "accessKey": "mykey"
}
```

```
{
  "section": "account",
  "request": "setEmail",
  "response": "OK"
}
```

setSessionLength

Set the time in seconds after a session of this user times out.

```
{
  "section": "account",
  "request": "setSessionLength",
  "sessionLength": 6000,
  "accessKey": "mykey"
}
```

```
{
  "section": "account",
  "request": "setSessionLength",
  "response": "OK"
}
```

changePassword

Change the password of the user account.

```
{
  "section": "account",
  "request": "changePassword",
  "newPassword": "password",
  "oldPassword": "htp",
  "accessKey": "mykey"
}
```

```
{
  "section": "account",
  "request": "changePassword",
  "response": "OK"
}
```