Arigi REST API

Kastelo Inc.

1.1

5/12 2019

CONTENTS

CHAPTER

ONE

INTRODUCTION

1.1 Principle of Operation

In this document, to set means to create or update an object. Objects are both created and updated using the http put method and every request is idempotent. Set operations typically return 204 No Content on success, unless otherwise specified. This also means that to update an attribute on an object you need to post the complete object.

Operations that are described as get will return a single JSON object. Operations that are described as list will instead return a JSON array of objects. In both cases the expected successful status is 200 OK with other codes indicating an error.

All delete operations use the http delete method and return 204 No Content on success. Delete operations are not idempotent, in that they will instead return 404 Not Found if the object to be deleted does not exist. The client may choose to interpret this as success in that the object in question does not exist after the operation.

All data exchanged is in JSON format. All attribute names follow the lowerCamelCase convention.

It is useful to use the arigi CLI when exploring the REST API. Specifically, running `arigi --api-verbose ...any command...` will show, in addition to the command results, the exact operations performed, data sent, and data received.

1.2 Data Model

Arigi uses an internal data model based on the following objects:

Devices represent Syncthing instances. These are identified by their (Syncthing) device ID, and must have an API key and API port.

Templates contain configuration fragments that can be evaluated and applied to devices.

Tags can be attached to devices and templates in order to tie them together.

CHAPTER

TWO

AUTHENTICATION

Authentication in the API uses *tokens*. A token is acquired by the login method using a regular Arigi username and password. The token is then passed in the Authorization header as the Bearer scheme.

`POST /api/v1/login`

Log in by acquiring a token.

Example request body:

```
{
    "username": "admin",
    "password": "arigi"
}
```

Example response body:

```
{
    "result":"ok",
    "token":"MTUwOTE3NzMz...L8XrRbgCngaFabirdX0="
}
```

Subsequent API requests should then include the following header:

```
Authorization: Bearer MTUwOTE3NzMz...L8XrRbgCngaFabirdX0=
```

Note that this scheme is only secure over a secure channel such as HTTPS. Arigi uses HTTPS by default.

CHAPTER

THREE

DEVICE MANAGEMENT

3.1 List devices

GET /api/v1/devices
Returns a list of devices.

Example response:

```
[
    {
        "deviceID": "T5ZNYVY-VDRQ2HH-FP26UHC-FO6NOJ3-YQWK7AV-...",
        "apiKey": "uRT9cSDVnaKe6F3GEN2YycEwiqZHtNjv",
        "apiPort": 8384,
        "apiAddress": "localhost",
        "label": "s1",
        "createdAt": "2017-10-28T10:06:48.087317052+02:00",
        "updatedAt": "2017-10-28T10:06:48.087317052+02:00"
    }
]
```

3.2 Get device

GET /api/v1/devices/(**string:** *deviceID*)
Example response:

```
{
    "deviceID": "T5ZNYVY-VDRQ2HH-FP26UHC-FO6NOJ3-YQWK7AV-...",
    "apiKey": "uRT9cSDVnaKe6F3GEN2YycEwiqZHtNjv",
    "apiPort": 8384,
    "apiAddress": "localhost",
    "label": "s1",
    "createdAt": "2017-10-28T10:06:48.087317052+02:00",
    "updatedAt": "2017-10-28T10:06:48.087317052+02:00"
}
```

3.3 Set Device

PUT /api/v1/devices/(**string:** *deviceID*)
Example request:

```
{
    "deviceID": "T5ZNYVY-VDRQ2HH-FP26UHC-FO6NOJ3-YQWK7AV-...",
    "apiKey": "uRT9cSDVnaKe6F3GEN2YycEwiqZHtNjv",
    "apiPort": 8384,
    "apiAddress": "localhost",
    "label": "s1"
}
```

The `apiAddress` attribute is optional - when blank or missing, Arigi will instead use dynamic discovery to find the device address. The `deviceID` attribute is optional but must then match the device ID in the URL if set.

3.4 Delete Device

DELETE /api/v1/devices/(**string:** *deviceID*)
The response is `204 No Content` on successfull delete.

3.5 Get Device Status

GET /api/v1/devices/(**string:** *deviceID*)**/status**
This endpoint mirrors Syncthing's API call to get device status.

Example response:

```
{
    "myID": "T5ZNYVY-VDRQ2HH-FP26UHC-FO6NOJ3-YQWK7AV-...",
    "alloc": 15395616,
    "cpuPercent": 0.019787337440146772,
    "discoveryEnabled": true,
    "discoveryMethods": 8,
    "goRoutines": 65,
    "pathSeparator": "/",
    "sys": 33286392,
    "tilde": "/Users/jb",
    "uptime": 71,
    "createdAt": "2017-10-28T10:14:43.721492134+02:00",
    "updatedAt": "2017-10-28T10:14:43.721492134+02:00"
}
```

3.6 Get Device Version

GET /api/v1/devices/(**string:** *deviceID*)**/version**
This endpoint mirrors Syncthing's API call to get the device version.

Example response:

```
{
    "device": "T5ZNYVY-VDRQ2HH-FP26UHC-FO6NOJ3-YQWK7AV-...",
    "arch": "amd64",
    "codename": "Dysprosium Dragonfly",
    "longVersion": "syncthing v0.14.40-rc.3+5-gf8015c9a ...",
    "os": "darwin",
    "version": "v0.14.40-rc.3+5-gf8015c9a",
    "createdAt": "2017-10-28T10:14:43.721879991+02:00",
    "updatedAt": "2017-10-28T10:14:43.721879991+02:00"
}
```

3.7 List Device Tags

GET /api/v1/devices/(**string:** *deviceID*)**/tags**
Returns the current set of tags for the device.

Example response:

```
[
    "s1"
]
```

3.8 Tag a Device

PUT /api/v1/devices/(**string:** *deviceID*)**/tags/**
string: *tag* Set a tag on a given device. No request body is necessary. The response contains the resulting set of tags for the device, same format as the list tags call.

3.9 Untag a Device

DELETE /api/v1/devices/(**string:** *deviceID*)**/tags/**
string: *tag* The response contains the resulting set of tags for the device, same format as the list tags call. This is possibly the empty list.

CHAPTER

FOUR

TEMPLATE MANAGEMENT

Templates are the fragments that make up the configuration system. Each template has a numeric ID which is automatically assigned by Arigi and uniquely identifies the template in the API, and a label which is human readable and not necessarily unique (although that is probably best).

A template has the following attributes:

label Human friendly label.

priority Numeric priority index.

op Operation to apply.

key Key into the configuration object where we apply the template operation.

template HJSON or Python (Starlark) template.

4.1 List Templates

GET /api/v1/templates
Returns the list of templates.

Example response:

```
[
    {
        "id": 1,
        "createdAt": "2017-10-28T10:20:05.535808929+02:00",
        "updatedAt": "2017-10-28T10:20:05.535808943+02:00",
        "label": "Disable NAT",
        "priority": 50,
        "key": "options.natEnabled",
        "op": "set",
        "template": "false\n"
    }
]
```

4.2 Set Template

POST /api/v1/templates

PUT /api/v1/templates/(**int:** *templateID*)
Creates (POST) or updates (PUT) an existing template.

Example request:

```
{
    "label": "Disable NAT",
    "priority": 50,
    "key": "options.natEnabled",
    "op": "set",
    "template": "false\n"
    }
```

4.3 Get Template

GET /api/v1/templates/(**int:** *templateID*)
Example response:

```
{
    "id": 1,
    "createdAt": "2017-10-28T10:20:05.535808929+02:00",
    "updatedAt": "2017-10-28T10:20:05.535808943+02:00",
    "label": "Disable NAT",
    "priority": 50,
    "key": "options.natEnabled",
    "op": "set",
    "template": "false\n"
}
```

4.4 Delete Template

DELETE /api/v1/templates/(**int:** *templateID*)
The response is `204 No Content` on successfull delete.

4.5 Tag a Template

PUT /api/v1/templates/(**int:** *templateID*)**/tags/**
string: *tag* The response contains the resulting set of tags for the template.

Example response:

```
[
    "s1"
]
```

4.6 Untag a Template

DELETE /api/v1/templates/(**int:** *templateID*)**/tags/**
string: *tag* The response contains the resulting set of tags for the template, which is possibly the empty list.

4.7 Evaluate a Template (Existing Template)

GET /api/v1/templates/(**int:** *templateID*)**/evaluate**
Evaluates the template against an automatically chosen example device and returns the results, or an error. The request body is empty.

TBD: Example

4.8 Evaluate a Template (New Template)

POST /api/v1/templates/evaluate
Evaluates the template against an automatically chosen example device and returns the results, or an error. The template to evaluate is sent as the request body.

TBD: Example

CHAPTER

FIVE

TAG MANAGEMENT

5.1 List Tags

GET /api/v1/tags
Returns the complete set of tags currently in existence.

Example response:

```
[
    "s1"
]
```

5.2 Delete Tag

DELETE /api/v1/tags/(**string:** *tag*)
The tag is removed from all devices and all templates. Returns the list of remaining tags.

5.3 List Devices for Tag

GET /api/v1/tags/(**string:** *tag*)**/devices**
TBD: Example

5.4 List Templates for Tag

GET /api/v1/tags/(**string:** *tag*)**/templates**
TBD: Example

CHAPTER

SIX

FOLDER STATUS

6.1 List Folders

GET /api/v1/folders
Returns the list of folder IDs and their labels.

Example response:

```
[
    {
        "folder": "2fyjq-jrfpr",
        "label": "Foton/2017"
    },
    {
        "folder": "4vcqq-sciqk",
        "label": "Transfer"
    }
]
```

6.2 List Folder Status

GET /api/v1/folders/(**string:** *folderID*)
Returns the list of folder summaries, one per device that shares the folder.

Example response:

```
[
    {
        "device": "GB33MBV-EKIA4IA-RNNU2YP-MHYRNL7-AWPB5JR-...",
        "folder": "lightroom",
        "createdAt": "2017-09-27T20:30:39.824251Z",
        "updatedAt": "2017-10-28T09:04:51.017775Z",
        "globalBytes": 21775074403,
        "globalDeleted": 11,
        "globalDirectories": 21486,
        "globalFiles": 24727,
        "globalSymlinks": 0,
        "localBytes": 21775074403,
        "localDeleted": 11,
```

(continues on next page)

(continued from previous page)

```
        "localDirectories": 21486,
        "localFiles": 24727,
        "localSymlinks": 0,
        "inSyncBytes": 21775074403,
        "inSyncFiles": 24727,
        "needBytes": 0,
        "needDeletes": 0,
        "needDirectories": 0,
        "needFiles": 0,
        "needSymlinks": 0,
        "ignorePatterns": false,
        "sequence": 95173,
        "version": 95173
    },
    ...
]
```