University of Lethbridge

3720 group project proposal

Web API Documentation

Team Aegir

Lorenzo Conrad
Mathew Richards
Steven Deutekom

December 1, 2019

Contents

1 Overview

Issue Tracker Express RESTful API can be used to track issues for a project. This document describes the API's endpoints and gives example requests and responses for each one. There are three sections. First methods for getting, creating, filtering, updating, and deleting issues are described. These are followed by methods for getting, creating, updating and deleting users. Lastly, the methods for getting, creating, updating, filtering, and deleting comments are given. Each section consists of some general information on the specific endpoint followed by an example request and response. There may also be lists of possible variations to the endpoint or specific error responses. The general error responses will be given in this section.

1.1 Request Format

For technical reasons our API only uses the GET and POST methods. For this reason the specific format of a request changes a little to accommodate the different kinds of requests. Table one gives an list of all available endpoints. Each request starts with the url of the server and is followed by */trackEx* and then *issues, comments, or users*. After this parameters can be added with a *?* and are separated by *&*. Any parameter that takes an argument will be formatted *key=value*.

POST requests must send information in the body as a JSON string. Any missing fields will be given default values. The available fields for each object type will be detailed in the respective section.

Any endpoint with *id* as a parameter will return as if only an id was given and ignore other parameters. The *id* parameter is used to find a single object and cannot be combined with other parameters.

Service	Method	Endpoint	Parameters
Get Issue(s)	GET	/trackEx/issues	id
Filtered Issue List	GET	/trackEx/issues	tag, priority, status
Create/Update issue	POST	/trackEx/issues	
Remove Issue	GET	/trackEx/issues?delete	id
Get users(s)	GET	/trackEx/users	id
Create/update user	POST	/trackEx/user	
Remove User	GET	/trackEx/users?delete	id
Get comment(s)	GET	/trackEx/comments	id, issue
Create/update comment	POST	/trackEx/comment	
Remove Comment	GET	/trackEx/comments?delete	id

Table 1: Service Endpoints

1.2 Response Format

The format of all responses is as follows:

```
{
    "status": "some status",
    "response": "a JSON object or message",
}
```

The status of the response will be *ok* or *fail*. If the status is *ok* the response body will be a JSON object or list of JSON objects for the requested objects. If the status is *fail* then the response object will contain an error message. Table 2 lists the possible error messages. Any HTTP errors mean that the endpoint given is incorrect or the server is not functioning.

Error	Message
No object with the requested id exists	"invalid id"
No Issue with id when filtering comments	"invalid issue id"

Table 2: Error Messages

2 Issues

Issues are the main component of the issue tracking system. Table 3 lists the available fields for Issues with their type and default values. When creating and updating an issue all fields can be specified. If an *id* is given when making a POST request it will update the issue with that *id* if it exists, or return an error if there is no issue with that *id*. It is not possible to create an issue with a custom *id*, the server assigns them. All fields that refer to other objects use integers to keep track of the objects *id*. This means that getting associated objects will need to be done with subsequent API calls. Priority is specified by an integer. This is to allow you to use your own priority system. Status is specified by an integer, but the server maps them to the following, starting at 0, {NEW, ASSIGNED, FIXED, WONT_FIX}. So there are currently only 4 valid values for status. Any other values will result in an error.

If an issue is removed all comments associated with that issue will be removed from the system.

A list of issues can be filtered by priority, tag, and status. It is possible to combine these in any way. The response will contain a list of all issues that satisfy each of the constraints. If there are no issues with a constraint then an empty list will be returned. In the future it may be possible to filter by more parameters

like assignee, creator or multiple tags. We would also like to add the ability to search for keywords or phrases in the titles and descriptions of all issues.

Field	Type	Default Value
id	int	Next available id
title	string	”empty”
description	string	”empty”
priority	int	-1
status	int	0,1,2,3
assignee	int	-1
creator	int	-1
tags	list of strings	[ ]

Table 3: Issue Fields

2.1 Get All Issues

Method:

`GET`

Request:

`/trackEx/issues`

Response:

```
{
    "status": "ok",
    "response": [
        {
            "id":1,
            "title":"Howard hates slack"
            "description":"Stop wasting his time",
            "assignee":2,
            "creator":-1,
            "priority":10,
            "status":1,
            "tags":["apps","time"],
        },
        {
            "id":2,
            "title":"My lunch is gone"
            "description":"People are stealing from the fridge in the break room!",
            "assignee":2,
            "creator":-1,
            "priority":10,
            "status":1,
            "tags":["fridge","stolen"],
        },
    ]
}
```

2.2 Get A Single Issue

Method:

```
GET
```

Request:

```
/trackEx/issues?id=2
```

Response:

```
{
    "status": "ok",
    "response":
        {
            "id":2,
            "title":"My lunch is gone"
            "description":"People are stealing from the fridge in the break room!",
            "assignee":2,
            "creator":-1,
            "priority":10,
            "status":1,
            "tags":["fridge","stolen"],
        },
}
```

2.3 Get A Filtered List Of Issues

Method:

```
GET
```

Request:

```
/trackEx/issues?filter&priority=10&status=1&tag=apps
```

Response:

```
{
    "status": "ok",
    "response": [
        {
            "id":1,
            "title":"Howard hates slack"
            "description":"Stop wasting his time",
            "assignee":2,
            "creator":-1,
            "priority":10,
            "status":1,
            "tags":["apps","time"],
        },
    ]
}
```

Request Variations:

```
/trackEx/issues?filter&status=1
/trackEx/issues?filter&priority=10
/trackEx/issues?filter&tag=apps
/trackEx/issues?filter&priority=10&tag=apps
/trackEx/issues?filter&status=1&tag=apps
/trackEx/issues?filter&priority=10&status=1
```

2.4 Create An Issue

Method:

```
POST
```

Request:

```
/trackEx/issues

== Request Body ==

{
    "title":"Howard hates slack"
    "description":"Stop wasting his time",
    "assignee":2,
    "creator":-1,
    "priority":10,
    "status":1,
    "tags":["apps","time"],
}
```

Response:

```
{
    "status": "ok",
    "response": [
        {
            "id":1,
            "title":"Howard hates slack"
            "description":"Stop wasting his time",
            "assignee":2,
            "creator":-1,
            "priority":10,
            "status":1,
            "tags":["apps","time"],
        },
    ]
}
```

2.5 Update An Issue

Method:

```
POST
```

Request:

```
/trackEx/issues

== Request Body ==

{
    "id": 1,
    "title":"Howard hates slack"
    "description":"Stop wasting his time",
    "assignee":2,
    "creator":-1,
    "priority":10,
    "status":3,
    "tags":["apps","time"],
}
```

Response:

```
{
    "status": "ok",
    "response": [
        {
            "id":1,
            "title":"Howard hates slack"
            "description":"Stop wasting his time",
            "assignee":2,
            "creator":-1,
            "priority":10,
            "status":3,
            "tags":["apps","time"],
        },
    ]
}
```

2.6 Delete An Issue

Method:

```
GET
```

Request:

```
/trackEx/issues?delete&id=2
```

Response:

```
{
    "status": "ok",
    "response":
        {
            "id":2,
            "title":"My lunch is gone"
            "description":"People are stealing from the fridge in the break room!",
            "assignee":2,
            "creator":-1,
            "priority":10,
            "status":1,
            "tags":["fridge","stolen"],
        },
}
```

3 Users

Users allow you to keep track of the different people using your system. This makes it possible to know who created an issue or a comment and to assign issues to specific users. Table 4 lists all the fields for users with their types and default values. As with issues POST requests with an id in the body will update a user. Without an id POST requests will create a new user with an id assigned by the server. The pic field uses an integer to keep track of what picture to use when displaying the user. Our website defines 6 picture and uses -1 to keep track of users without a picture. If you create your own web client it is up to you to actually map these numbers to pictures or decide how many pictures to allow. If you automate using the API these values will not be useful. The blurb is meant as an "about me" section where a user can put a little text that will describe or distinguish them from other users.

If a user is removed from the system all comments and issues that they are associated with will be updated with a -1.

Field	Type	Default Value
id	int	Next available id
name	string	"empty"
blurb	string	"empty"
pic	int	-1

Table 4: User Fields

3.1 Get All Users

Method:

```
GET
```

Request:

```
/trackEx/users
```

Response:

```
{
    "status": "ok",
    "response": [
        {
            "id":1,
            "name":"Jimmy Cricket"
            "blurb":"Wish upon a star!",
            "pic":1
        },
        {
            "id":2,
            "name":"Peppa Pig"
            "blurb":"Happy all the time",
            "pic":2
        },
    ]
}
```

3.2 Get A Single User

Method:

```
GET
```

Request:

```
/trackEx/users?id=2
```

Response:

```
{
    "status": "ok",
    "response":
        {
            "id":2,
            "name":"Peppa Pig"
            "blurb":"Happy all the time",
            "pic":2
        }
}
```

3.3 Create A User

Method:

```
POST
```

Request:

```
/trackEx/users

== Request Body ==

{
    "name":"Jimmy Cricket"
    "blurb":"Wish upon a star!",
    "pic":1
}
```

Response:

```
{
    "status": "ok",
    "response":
        {
            "id":1,
            "name":"Jimmy Cricket"
            "blurb":"Wish upon a star!",
            "pic":1
        }
}
```

3.4 Update A User

Method:

```
POST
```

Request:

```
/trackEx/users

== Request Body ==

{
    "id":1,
    "name":"Jimmy Cricket"
    "blurb":"Spelled my name wrong!",
    "pic":4
}
```

Response:

```
{
    "status": "ok",
    "response":
        {
            "id":1,
            "name":"Jimmy Cricket"
            "blurb":"Wish upon a star!",
            "pic":1
        }
}
```

3.5 Delete A User

Method:

```
GET
```

Request:

```
/trackEx/users?delete&id=2
```

Response:

```
{
    "status": "ok",
    "response":
        {
            "id":2,
            "name":"Peppa Pig"
            "blurb":"Happy all the time",
            "pic":2
        }
}
```

4 Comments

Comments allow users to communicate ideas or information for an issue. Table 5 shows the comment fields and their type and default value. Like with other objects POST requests with an id will update an existing comment and without an id will create a new comment with an id assigned by the server. Each comment should have an issue id to keep track of which issue it is attached to. To improve this feature it is possible to get a list of only the comments that are associated with an issue with the *issue* parameter. Comments are kept sequentially and do not allow nesting replies. If a comment is removed all other comments will move up one in the list.

Field	Type	Default Value
id	int	Next available id
text	string	""
user	int	-1

Table 5: Comment Fields

4.1 Get All Comments

Method:

```
GET
```

Request:

```
/trackEx/comments
```

Response:

```
{
    "status": "ok",
    "response": [
        {
            "id": 1,
            "user_id": 1,
            "issue_id": 1,
            "text": "You better! The boss is mad!"
        },
        {
            "id":2,
            "user_id": 2,
            "issue_id": 1,
            "text": "I'll get started on this right away"
        }
    ]
}
```

4.2 Get A Single Comment

Method:

```
GET
```

Request:

```
/trackEx/comments?id=2
```

Response:

```
{
    "status": "ok",
    "response":
        {
            "id":2,
            "user_id": 2,
            "issue_id": 1,
            "text":"I'll get started on this right away"
        }
}
```

4.3 Get Comments For A Single Issue

Method:

```
GET
```

Request:

```
/trackEx/comments?issue=1
```

Response:

```
{
    "status": "ok",
    "response": [
        {
            "id": 1,
            "user_id": 1,
            "issue_id": 1,
            "text": "You better! The boss is mad!"
        },
        {
            "id":2,
            "user_id": 2,
            "issue_id": 1,
            "text": "I'll get started on this right away"
        }
    ]
}
```

4.4 Create A Comment

Method:

```
POST
```

Request:

```
/trackEx/comments

== Request Body ==
{
    "user_id": 2,
    "issue_id": 1,
    "text":"I'll get started on this right away"
}
```

Response:

```
{
    "status": "ok",
    "response":
        {
            "id":1,
            "user_id": 2,
            "issue_id": 1,
            "text":"I'll get started on this right away"
        }
}
```

4.5 Update A Comment

Method:

```
POST
```

Request:

```
/trackEx/comments

== Request Body ==

{
    "id": 1,
    "user_id": 2,
    "issue_id": 1,
    "text":"I'll get started in a week"
}
```

Response:

```
{
    "status": "ok",
    "response":
        {
            "id":1,
            "user_id": 2,
            "issue_id": 1,
            "text":"I'll get started in a week"
        }
}
```

4.6 Delete A Comment

Method:

```
GET
```

Request:

```
/trackEx/comments?delete&id=1
```

Response:

```
{
    "status": "ok",
    "response":
        {
            "id":1,
            "user_id": 2,
            "issue_id": 1,
            "text":"I'll get started on this right away"
        }
}
```