Seed Auth API Documentation

Release 0.0.0

Praekelt Foundation

May 17, 2016

Contents

Contents:

CHAPTER 1

HTTP API

Seed Auth's HTTP API.

1.1 Authentication

Authentication is done via token authentication.The tokens endpoint can be used to create a token. The token can then be placed in the header for requests to the other endpoints.

Example request:

```
POST /endpoint/ HTTP/1.1
Content-Type: application/json
Authorization: Token 9944b09199c62bcf9418ad846dd0e4bbdfc6ee4b
```

1.2 Permissions

While you can store any string as a permission, there are a few permissions that have specific meaning within the Seed Auth API.

Create permissions are not linked to specific instances, whereas admin, read, and write permissions are.

A user automatically has read/write permissions for themselves.

All users have read permissions for all users and organizations.

All members of an organization have read access to that organization's teams.

Permissions can be assigned either to a user or a team. All users can add and remove permissions for all teams and users, except for the following permissions, which can only be set be set by a person with org:admin permission.

org:admin Can create/read/write/delete users, and teams that are part of the organization, and can add users to the organization that they are an admin for. Can read/write the organization that they are admin for.

org:write Can modify an organization's details, including adding existing users and teams to the organization.

team:admin Can modify the team they have permission for, and add and remove existing users to that team.

team:read Can view the team they have permission for.

user:create Can create new users.

`GET /permissions`
Verify that an existing token is valid, and return the token's permissions.

Response Headers

- Authorization – "Token " followed by the token to verify

Status Codes

- 200 OK – The token is valid.
- 400 Bad Request – The token is invalid.

Example request:

```
GET /token HTTP/1.1
Authorization: Token 9944b09199c62bcf9418ad846dd0e4bbdfc6ee4b
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  {
      "id": 2,
      "type": "org:admins",
      "object_id": "3",
      "properties": {}
  },
  {
      "id": 5,
      "type": "foo:bar:baz",
      "object_id": "7",
      "properties": {
          "example": "property",
          "number": 7
      }
  }
]
```

1.3 Pagination

When the results set is larger than a configured amount, the data is broken up into pages.

You can navigate to specific pages using the 'page' parameter. Links to the next and previous page (if available) will be provided in the 'Link' header.

Example:

```
GET /endpoint/ HTTP/1.1
Authorization: JWT .....


HTTP/1.1 200 OK
Content-Type: application/json
Link: <https://example.com/endpoint/?page=2>; rel="next"

[....]
```

1.4 Tokens

For the token endpoints, no authentication is required.

POST /tokens/

Create a new token for the provided user. This will invalidate all other tokens for that user.

Request JSON Object

- **username** (*str*) – The username of the user to create the token for.
- **password** (*str*) – The password of the user to create the token for.

Response JSON Object

- **token** (*str*) – The generated token.

Status Codes

- 201 Created – When the token is successfully generated.
- 400 Bad Request – When the user credentials are incorrect.

Example request:

```
POST /tokens/ HTTP/1.1
Content-Type: application/json

{
  "username": "testuser",
  "password": "testpassword"
}
```

Example response:

```
HTTP/1.1 201 Created
Content-Type: application/json

{
  "token": "9944b09199c62bcf9418ad846dd0e4bbdfc6ee4b"
}
```

1.5 Password resets

For the password reset endpoints, no authentication is required.

To reset a user's password, the following steps should be followed:

1. Make a request to the reset endpoint. This will make an HTTP request to the preconfigured endpoint with the user's details, and a token.
2. Make a request to the confirm endpoint, with the provided token and the new password.

POST /passwords/resets/

Start the process for resetting a user's password.

Request JSON Object

- **email** (*str*) – The email of the user to reset the password for.
- **app** (*str*) – The application that the token should go to, configured in settings. This value is optional, defaults to the default configured application.

Status Codes

- 202 Accepted – The password reset process was started, or the username doesn't exist. The same code is returned for both as to not leak user information

Example request:

```
POST /passwords/resets/ HTTP/1.1
Content-Type: application/json

{"email":"jonsnow@castleblack.org","app":"numi"}
```

Example response:

```
HTTP/1.1 202 Accepted
```

POST /passwords/confirmations/

Reset the users password using the provided token.

Request JSON Object

- **token** (*str*) – The provided token.
- **password** (*str*) – The new password.

Status Codes

- 204 No Content – The password was successfully reset.
- 400 Bad Request – The token was incorrect.

Example request:

```
POST /password/confirmations/ HTTP/1.1
Content-Type: application/json

{"password":"gh0st","token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibm
```

Example response:

```
HTTP/1.1 204 No Content
```

1.6 Organizations

Organizations provide a grouping of users, although users do not have to belong to an organization, and they can also belong to many organizations. Teams have to belong to exactly one organization, but an organization can have many teams.

POST /organizations/

Creates a new organization.

Response JSON Object

- **name** (*str*) – The name of the created organization.
- **id** (*int*) – The id of the created organization.
- **teams** (*list*) – The list of teams that the organization has.
- **users** (*list*) – The list of users that are part of the organization.

Status Codes

- 201 Created – When the organization is successfully generated.
- 400 Bad Request – When there is invalid information to create the organization.

Example request:

```
POST /organizations/ HTTP/1.1
Content-Type: application/json

{"name":"Nights Watch"}
```

Example response:

```
HTTP/1.1 201 Created
Content-Type: application/json

{"name":"Nights Watch","id":4,"teams":[],"url":"https://example.org/organizations/4","users":[]}
```

GET /organizations/
Get a list of existing organizations

Example request:

```
GET /organizations/ HTTP/1.1
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[{"name":"Nights Watch","id":4,"teams":[],"url":"https://example.org/organizations/4","users":[]
```

GET /organizations/ (**int:** *organization_id*)
Get the details of an organization.

Response JSON Object

- **name** (*str*) – The name of the created organization.
- **id** (*int*) – The id of the created organization.
- **teams** (*list*) – The list of teams that the organization has.

Example request:

```
GET /organizations/4 HTTP/1.1
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{"name":"Night's Watch","id":4,"teams":[],"url":"https://example.org/organizations/4","users":[]
```

PUT /organizations/ (**int:** *organization_id*)
Update an existing organization.

Request JSON Object

- **name** (*str*) – The name of the organization.

Response JSON Object

- **id** (*int*) – The id of the created organization.

- **teams** (*list*) – The list of teams that the organization has.
- **users** (*list*) – The list of users that are part of the organization.

Status Codes

- 200 OK – When the organization is successfully generated.
- 400 Bad Request – When there is invalid information to update the organization.

Example request:

```
PUT /organizations/4 HTTP/1.1
Content-Type: application/json

{"name": "Brotherhood Without Banners"}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{"name":"Brotherhood Without Banners","id":4,"teams":[],"url":"https://example.org/organizations
```

DELETE /organizations/ (**int:** *organization_id*)

Remove an existing organization.

status 204 Resource successfully deleted

Example request:

```
DELETE /organizations/4 HTTP/1.1
```

Example response:

```
HTTP/1.1 204 No Content
```

POST /organizations/ (**int:** *organization_id*) **/users/**

Add a user to an existing organization.

Request JSON Object

- **user_id** (*int*) – The ID of the user to add.

Status Codes

- 204 No Content – User was successfully added.

Example request:

```
POST /organizations/4/users/ HTTP/1.1
Content-Type: application/json

{"user_id": 2}
```

Example response:

```
HTTP/1.1 204 No Content
```

DELETE /organizations/ (**int:** *organization_id*) **/users/**

int: *user_id* Remove a user from an organization.

Status Codes

- 204 No Content – User was successfully removed from an organization

Example request:

```
DELETE /organizations/4/users/2 HTTP/1.1
```

Example response:

```
HTTP/1.1 204 No Content
```

1.7 Teams

GET /teams/
Get a list of all the teams you have read access to.

Example request:

```
GET /teams/ HTTP/1.1
```

Example response:

```
HTTP/1.1 200 OK

[
    {
        "id": 4,
        "name": "admins",
        "permissions": [],
        "users": [],
        "url": "https://example.org/teams/4",
        "organization": 7
    }
]
```

GET /teams/
Allows filtering of teams to retreive a subset.

Query Parameters

- **type_contains** (*string*) – The type field on one of the resulting team's permissions must contain this string.
- **object_id** (*string*) – All the object_id fields on one of the resulting team's permissions must equal this string.

Example request:

```
GET /teams/?permission_contains=org&object_id=3 HTTP/1.1
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
    {
        "id": 4,
        "name": "organization admins",
        "users": [],
        "permissions":
            [
```

```
                    {
                        "id": 2,
                        "type": "org:admins",
                        "object_id": "3",
                        "properties": {}
                    }
                ],
            "url": "https://example.org/teams/4",
            "organization": 3
        },
        {
            "id": 7,
            "name": "organization editors",
            "users": [],
            "permissions":
                [
                    {
                        "id": 3,
                        "type": "org:write",
                        "object_id": "3",
                        "properties": {}
                    }
                ],
            "url": "https://exmple.org/teams/6",
            "organization": 3
        }
    ]
```

POST /teams/

Create a new team.

Request JSON Object

- **name** (*str*) – The name of the team.
- **organization** (*int*) – The id of the organization that the team belongs to.

Response JSON Object

- **id** (*int*) – The ID of the created team.
- **url** (*str*) – The URL of the created team.
- **name** (*str*) – the name of the team.
- **users** (*list*) – The list of users that belong to this team.
- **organization** (*int*) – The id of the organization that the team belongs to.
- **permissions** (*list*) – The permission list of the team. Each permission is an object containing the fields "id", "type", "object_id", and "properties", where "id" is the id of the permission, "type" is a string representing the type of permission, "object_id" is a string or null, representing the object that the permission acts on, and properties is a flat object to add any additional properties.

Status Codes

- 201 Created – Successfully created team.
- 400 Bad Request – Missing required information to create team.

Example request:

```
POST /teams/ HTTP/1.1
Content-Type: application/json

{
    "name": "Lord Commanders",
    "organization": 7
}
```

Example response:

```
HTTP/1.1 201 Created
Content-Type: application/json

{
    "id": 2,
    "name": "Lord Commanders",
    "users": [],
    "permissions": [],
    "url": "https://example.org/teams/2",
    "organization": 7
}
```

GET /teams/ (**int:** *team_id*)
Get the details of a team.

Response JSON Object

- **id** (*int*) – the ID of the team.
- **url** (*str*) – the URL of the team.
- **name** (*str*) – the name of the team.
- **users** (*list*) – The list of users that belong to this team.
- **organization** (*int*) – The id of the organization that the team belongs to.
- **permissions** (*list*) – The permission list of the team. Each permission is an object containing the fields "id", "type", "object_id", and "properties", where "id" is the id of the permission, "type" is a string representing the type of permission, "object_id" is a string or null, representing the object that the permission acts on, and properties is a flat object to add any additional properties.

Status Codes

- 200 OK – Successfully retrieved team.

Example request:

```
GET /teams/2 HTTP/1.1
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "id": 2,
    "name": "Lord Commanders",
    "permissions": [],
    "users": [],
    "url": "https://example.org/teams/2",
```

```
        "organization": 7
    }
```

PUT /teams/ (**int:** *team_id*)
Update the details of a team.

Request JSON Object

- **name** (*str*) – The name of the team.

Response JSON Object

- **id** (*int*) – the id of the updated team.
- **url** (*str*) – The URL of the updated team.
- **name** (*str*) – the name of the team.
- **users** (*list*) – The list of users that belong to this team.
- **organization** (*int*) – The id of the organization that the team belongs to.
- **permissions** (*list*) – The permission list of the team. Each permission is an object containing the fields "id", "type", "object_id", and "properties", where "id" is the id of the permission, "type" is a string representing the type of permission, "object_id" is a string or null, representing the object that the permission acts on, and properties is a flat object to add any additional properties.

Status Codes

- 200 OK – successfully updated team.

Example request:

```
PUT /teams/2 HTTP/1.1
Content-Type: application/json

{
    "name": "Brotherhood without banners"
}
```

Example reponse:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "id": 2,
    "name": "Brotherhood without banners",
    "permissions": [],
    "users": [],
    "url": "https://example.org/teams/2",
    "organization": 7
}
```

DELETE /teams/ (**int:** *team_id*)
Remove a team.

Status Codes

- 204 No Content – Team successfully deleted.

Example request:

```
DELETE /teams/2 HTTP/1.1
```

Example response:

```
HTTP/1.1 204 No Content
```

POST /teams/ (**int:** *team_id*) **/permissions/**

Add a permission to a team.

Request JSON Object

- **type** (*str*) – The string representing the permission.
- **object_id** (*str*) – The id of the object that the permission acts on. "null" if it doesn't act on any object.
- **properties** (*obj*) – A single layer object that can contain any amount of keys. Used to add additional information that might be useful to external applications.

Response JSON Object

- **id** (*int*) – the id of the team.
- **url** (*str*) – the URL of the team.
- **name** (*str*) – the name of the team.
- **users** (*list*) – The list of users that belong to this team.
- **organization** (*int*) – The id of the organization that the team belongs to.
- **permissions** (*list*) – The permission list of the team. Each permission is an object containing the fields "id", "type", "object_id", and "properties", where "id" is the id of the permission, "type" is a string representing the type of permission, "object_id" is a string or null, representing the object that the permission acts on, and properties is a flat object to add any additional properties.

Status Codes

- 200 OK – successfully added permission to the team.

Example request:

```
POST /teams/2/permissions/ HTTP/1.1
Content-Type: application/json

{
    "type": "org:admin",
    "object_id": "2",
    "properties": {}
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "id": 2,
    "name": "Lord Commanders",
    "users": [],
    "permissions": [
        {
            "id": 17,
```

```
                "type": "org:admin",
                "object_id": "2",
                "properties": {}
            }
        ],
        "url": "https://example.org/teams/2",
        "organization": 7
    }
```

DELETE /teams/ (**int:** *team_id*) **/permissions/**
int: *permission_id* Remove a permission from a team.

Response JSON Object

- **id** (*int*) – the id of the team.
- **url** (*str*) – The URL of the team.
- **name** (*str*) – the name of the team.
- **users** (*list*) – The list of users that belong to this team.
- **organization** (*int*) – The id of the organization that the team belongs to.
- **permissions** (*list*) – The permission list of the team. Each permission is an object containing the fields "id", "type", "object_id", and "properties", where "id" is the id of the permission, "type" is a string representing the type of permission, "object_id" is a string or null, representing the object that the permission acts on, and properties is a flat object to add any additional properties.

Status Codes

- 200 OK – successfully removed permission from the team.

Example request:

```
DELETE /teams/2/permissions/17 HTTP/1.1
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "id": 2,
    "name": "Lord Commanders",
    "permissions": [],
    "users": [],
    "url": "https://example.org/teams/2",
    "organization": 7
}
```

POST /teams/ (**int:** *team_id*) **/users/**
Add an existing user to an existing team.

Request JSON Object

- **user_id** (*int*) – The ID of the user to add to the team.

Response JSON Object

- **id** (*int*) – the id of the team.
- **url** (*str*) – The URL of the team.

- **name** (*str*) – the name of the team.
- **users** (*list*) – The list of users that belong to this team.
- **organization** (*int*) – The id of the organization that the team belongs to.
- **permissions** (*list*) – The permission list of the team. Each permission is an object containing the fields "id", "type", "object_id", and "properties", where "id" is the id of the permission, "type" is a string representing the type of permission, "object_id" is a string or null, representing the object that the permission acts on, and properties is a flat object to add any additional properties.

Status Codes

- 200 OK – successfully added the user to the team.

Example request:

```
POST /teams/2/users/ HTTP/1.1
Content-Type: application/json

{
    "user_id": 1
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "id": 2,
    "name": "Lord Commanders",
    "permissions": [],
    "users":
        [
            {
                "id": 1,
                "url": "https://example.org/users/1"
            }
        ],
    "url": "https://example.org/teams/2",
    "organization": 7
}
```

DELETE /teams/ (**int:** *team_id*) **/users/1**

Remove a user from a team.

Response JSON Object

- **id** (*int*) – the id of the team.
- **url** (*str*) – The URL of the team.
- **name** (*str*) – the name of the team.
- **users** (*list*) – The list of users that belong to this team.
- **organization** (*int*) – The id of the organization that the team belongs to.
- **permissions** (*list*) – The permission list of the team. Each permission is an object containing the fields "id", "type", "object_id", and "properties", where "id" is the id of the permission, "type" is a string representing the type of permission, "object_id" is a string or

null, representing the object that the permission acts on, and properties is a flat object to add any additional properties.

Status Codes

- 200 OK – successfully removed the user from the team.

Example request:

```
DELETE /teams/2/users/1 HTTP/1.1
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "id": 2,
    "name": "Lord Commanders",
    "permissions": [],
    "users": [],
    "url": "https://example.org/teams/2",
    "organization": 7
}
```

1.8 Users

GET /users/

Get a list of all users.

Example request:

```
GET /users/ HTTP/1.1
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
    {
        "id": 1,
        "url": "https://example.org/users/1",
        "first_name": "Jon",
        "last_name": "Snow",
        "email": "jonsnow@castleblack.net",
        "admin": false,
        "teams": [
            {
                "id": 2,
                "url": "https://example.org/teams/2"
            }
        ],
        "organizations": [
            {
                "id": 4,
                "url": "https://example.org/organizations/4"
            }
        ]
```

```
        }
    ]
```

POST /users/

Create a new user.

Request JSON Object

- **first_name** (*str*) – The (optional) first name of the user.
- **last_name** (*str*) – The (optional) last name of the user.
- **email** (*str*) – The email address of the user.
- **password** (*str*) – The password for the user.
- **admin** (*bool*) – True if the user is an admin user.

Response JSON Object

- **id** (*int*) – The ID for the user.
- **url** (*str*) – The URL for the user.
- **first_name** (*str*) – The (optional) first name of the user.
- **last_name** (*str*) – The (optional) last name of the user.
- **email** (*str*) – The email address of the user.
- **admin** (*bool*) – True if the user is an admin user.
- **teams** (*list*) – A list of all the teams a user is a member of.
- **organizations** (*list*) – A list of all the organizations the user is a member of.

Status Codes

- 201 Created – Successfully created user.

Example request:

```
POST /users/ HTTP/1.1
Content-Type: application/json

{
    "first_name": "Jon",
    "last_name": "Snow",
    "email": "jonsnow@castleblack.net",
    "password": "gh0st",
    "admin": false
}
```

Example response:

```
HTTP/1.1 201 Created
Content-Type: application/json

{
    "id": 1,
    "url": "https://example.org/users/1",
    "first_name": "Jon",
    "last_name": "Snow",
    "email": "jonsnow@castleblack.net",
    "admin": false,
```

```
        "teams": [],
        "organizations": []
    }
```

GET **/users/** (**int:** *user_id*)
Get details on a specific user.

Response JSON Object

- **id** (*int*) – The ID for the user.
- **url** (*str*) – The URL for the user.
- **first_name** (*str*) – The (optional) first name of the user.
- **last_name** (*str*) – The (optional) last name of the user.
- **email** (*str*) – The email address of the user.
- **admin** (*bool*) – True if the user is an admin user.
- **teams** (*list*) – A list of all the teams a user is a member of.
- **organizations** (*list*) – A list of all the organizations the user is a member of.

Example request:

```
GET /users/1 HTTP/1.1
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "id": 1,
    "url": "https://example.org/users/1",
    "first_name": "Jon",
    "last_name": "Snow",
    "email": "jonsnow@castleblack.net",
    "admin": false,
    "teams": [
        {
            "id": 2,
            "url": "https://example.org/teams/2"
        }
    ],
    "organizations": [
        {
            "id": 4,
            "url": "https://example.org/organizations/4"
        }
    ]
}
```

PUT **/users/** (**int:** *user_id*)
Update the information of an existing user. Cannot update the password this way, see the "Password resets" section on how to update the user password.

Request JSON Object

- **first_name** (*str*) – The (optional) first name of the user.
- **last_name** (*str*) – The (optional) last name of the user.

- **email** (*str*) – The email address of the user.
- **admin** (*bool*) – True if the user is an admin user.

Response JSON Object

- **id** (*int*) – The ID for the user.
- **url** (*str*) – The URL for the user.
- **first_name** (*str*) – The (optional) first name of the user.
- **last_name** (*str*) – The (optional) last name of the user.
- **email** (*str*) – The email address of the user.
- **admin** (*bool*) – True if the user is an admin user.
- **teams** (*list*) – A list of all the teams a user is a member of.
- **organizations** (*list*) – A list of all the organizations the user is a member of.

Status Codes

- 200 OK – Successfully updated user.

Example request:

```
PUT /users/1 HTTP/1.1
Content-Type: application/json

{
    "first_name": "Jon",
    "last_name": "Snow",
    "email": "jonsnow@castleblack.org",
    "admin": true
}
```

Example response:

```
HTTP/1.1 201 Created
Content-Type: application/json

{
    "id": 1,
    "url": "https://example.org/users/1",
    "first_name": "Jon",
    "last_name": "Snow",
    "email": "jonsnow@castleblack.org",
    "admin": true,
    "teams": [],
    "organizations": []
}
```

DELETE /users/ (**int:** *user_id*)

Remove an existing user.

Status Codes

- 204 No Content – Successfully deleted the user.

Example request:

```
DELETE /users/1 HTTP/1.1
```

Example response:

```
HTTP/1.1 204 No Content
```

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

/organizations

/passwords

/permissions

/teams

/tokens

/users