Mmadu User Guide

Table of Contents

1. Overview

1.1. What is Mmadu?

Mmadu (pronounced um-a-du) is derived from an Igbo word meaning people.

Have you ever found yourself in a state of deja-vu when ever you write user management logic? Mmadu provides a set of tools that ensures that you never have to repeat user management logic. It helps to manage users, takes care of registration work flows and exposes APIs for authentication.

With Mmadu, you can have a dedicated system for managing users across all your applications.

1.2. Features

1.2.1. User Authentication

By calling an API, you can authenticate users of your application with Mmmadu. By passing a username and (encrypted) password, the rest api can determine whether the username/password combination is correct. This provides a basic method of integration with web security frameworks.

1.2.2. Domain Management

By creating an managing domains, you can logically separate users. Users in a domain are typically used for one application. My storing users in multiple domains, you can use a single mmadu installation to manage users across multiple applications.

1.2.3. User Management

The user service provides basic CRUD operations to help you manage and query your users.

1.2.4. Token Management

Mmadu utilizes tokens to secure its APIs. You can create tokens that grant access to one domain or utilize the admin token which has access to all domains.

1.2.5. User Registration

Mmadu provides the web logic for registering users. With configured fields and field types, the registration api generates dynamic and configurable registration forms that domain users fill to be registered in an application.

1.3. What's New

Version	Features
v1.3.0	- Introduced user registration - Added docker installation features
v1.1.0	- User management API redesigned and made user friendly - Bug fixes
v1.0.0	- Domain and user management API was introduced - Token security for domain and user management API was introduced.
v0.0.1	- Api for creating and updating user information created. - Api for authenticating username and password information has been created.

1.4. Getting Started

Mmadu can be installed in a number of ways since it is built and distributed with jar files. The easiest way to install mmadu is using docker.

1.4.1. Getting Started with Docker

Requirements

- Docker
- Docker Compose

Installation

1. Create a folder (we will use mmadu as the created folder)

```
mkdir mmadu && cd mmadu
```

2. Download the docker compose file

```
wget https://raw.githubusercontent.com/geraldoyudo/mmadu/v1.3.0/docker/docker-compose.yml
```

3. Download the default environment file

```
wget https://raw.githubusercontent.com/geraldoyudo/mmadu/v1.3.0/docker/.env
```

4. Make the necessary modifications to suite your configuration (you can use it as is)
5. Start the services

```
docker-compose up
```

Using the default settings

The docker compose installation comes pre-configured with default settings. You can access the api from each service using the default admin token 2222. The user management service, registration service and token service are at ports 15551, 15552 and 15553 respectively.

NOTE	Do not use default settings for configuration on production.

Using the services

To find out more about how to consume the service API, please see API

2. Architecture

Mmadu is built using a service oriented architecture. Each basic functionality is encapsulated in an independent service.

The core service is the user management service which handles authentication, user and domain management. The token service is used by all other services for token authentication.

Currently, mmadu has the following services:

Service	Description
Token Service	Service for managing and checking tokens and permissions.
User management Service	Manages domain authentication as well as administration of domains and users.
Registration Service	Provides a dynamic registration form for each domain. Exposes API endpoints to configure registration form fields.

3. API

3.1. Security

If token security is enabled, you will need a token to access the api.

Simply set the domain-auth-token header to the value of the token. For more information on tokens, see token management.

3.1.1. Permissions

A token has admin permissions, while others have domain permissions. The admin token can access all parts of the API, while domain tokens can only access parts of the api that returns resources pertaining to the domain.

3.2. Token Service

Mmadu service uses token based security. Access to API can be done using admin and domain tokens. The default port for the token service is 15553

3.2.1. Admin Token

An admin token is automatically created with an id "admin-token". This token will grant you access to all resources in the API.

3.2.2. Domain Tokens

Domain tokens are created and assigned to a domain configuration. Every domain can have one domain token. Multiple domains can share a domain token. Domain token grants the API user access to user resources for that domain.

3.2.3. Token Encryption

If encryption is enabled, the token values will be encrypted before storing in the database. It uses

an AES CBC cipher to encrypt token values.

Encryption can be enabled by setting the environment property `mmadu.domain.encrypt-keys` to `true`.

Token values are encrypted under the master-key.

Changing the Encryption Master Key

To change the encryption master key, set the environment property `mmadu.security.master-key` to the desired value.

3.2.4. Generating a Token

A GET request will create a new token and return the token value. If token encryption is enabled, the token value returned is encrypted.

Request

```
GET /token/generate HTTP/1.1
domain-auth-token: 2222
Host: localhost:8080
```

Response Fields

This API returns the newly generated token

Path	Type	Description
`id`	`String`	The token ID
`value`	`String`	The token value (Encrypted under master key if encryption is enabled)

Example:

```
HTTP/1.1 200 OK
Pragma: no-cache
X-XSS-Protection: 1; mode=block
Expires: 0
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Content-Length: 314
Content-Type: application/json;charset=UTF-8
Cache-Control: no-cache, no-store, max-age=0, must-revalidate

{
  "id" : "5d1dd7316fc6c30f1c2fa625",
  "value" :
"2613ec6bcf71f75a1f02f5ddb44092c800ea97417ee78724f2e043eb19f53c8a870f00cde4fb798434aaa
0d130dd76285b4dfc57b6eb7bd62eca82763a773dfc2f5d9468ef90a88c23db432a6077377aaaaff8cce4e
954ddd9206e2ee7f6c73a8e3d06e4957d461657b87857a1e2c8ea1bc7d3064531fb9cef7fb0f1a65cd95e"
}
```

3.2.5. Retrieving a Token

A GET request will get a token with the ID.

Path Parameters

Table 1. /token/retrieve/{tokenId}

Parameter	Description
`tokenId`	The token ID

Example:

```
GET /token/retrieve/1 HTTP/1.1
domain-auth-token: 2222
Host: localhost:8080
```

Response Fields

This API returns an AppUser with these fields:

Path	Type	Description
`id`	`String`	The token ID
`value`	`String`	The token value (Encrypted under master key if encryption is enabled)

Example:

```
HTTP/1.1 200 OK
Pragma: no-cache
X-XSS-Protection: 1; mode=block
Expires: 0
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Content-Length: 39
Content-Type: application/json;charset=UTF-8
Cache-Control: no-cache, no-store, max-age=0, must-revalidate

{
  "id" : "1",
  "value" : "1234"
}
```

3.2.6. Resetting a Token

A GET request will generate a new token value for the token.

Path Parameters

Table 2. /token/reset/{tokenId}

Parameter	Description
`tokenId`	The token ID

Example:

```
GET /token/reset/1 HTTP/1.1
domain-auth-token: 2222
Host: localhost:8080
```

Response Fields

This API returns a token with the following fields:

Path	Type	Description
`id`	`String`	The token ID
`value`	`String`	The token value (Encrypted under master key if encryption is enabled)

Example:

```
HTTP/1.1 200 OK
Pragma: no-cache
X-XSS-Protection: 1; mode=block
Expires: 0
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Content-Length: 291
Content-Type: application/json;charset=UTF-8
Cache-Control: no-cache, no-store, max-age=0, must-revalidate

{
  "id" : "1",
  "value" :
"4ae568b84d271d8c34c5146fefbf46a47614d33b212a9525985eb895a70e2cb186a3c6bddd062088430ce
6c9a8bb458798edb1cedb2067c82dd3c5ed4a9b5664ba1742181d7d12a32703fcc12b43da2ed495a34f5c2
fd6d365a0d0ec668f49a8c4994b10600441e43c61a59204c70542cad0c6f992fd1aeb2b44a7f8b2136db3"
}
```

3.2.7. Setting a Domain Token

A POST request will set a domain's token.

Request Fields

Path	Type	Description
`tokenId`	`String`	The id of the token
`domainId`	`String`	The domain id

Example:

```
POST /token/setDomainAuthToken HTTP/1.1
domain-auth-token: 2222
Content-Length: 39
Content-Type: application/json
Host: localhost:8080

{"tokenId":"1","domainId":"1111111111"}
```

Response Fields

This API returns a 204 no content response.

Example:

```
HTTP/1.1 204 No Content
Pragma: no-cache
X-XSS-Protection: 1; mode=block
Expires: 0
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
```

3.2.8. Getting the token associated to a Domain

A GET request will get a domain's token.

Path Parameters

Table 3. /token/domainAuth/{domainId}

Parameter	Description
`domainId`	The domain ID

Example:

```
GET /token/domainAuth/1111111111 HTTP/1.1
domain-auth-token: 2222
Host: localhost:8080
```

Response Fields

This API returns a token with these fields:

Path	Type	Description
`tokenId`	`String`	The token ID
`domainId`	`String`	The domain ID

Example:

```
HTTP/1.1 200 OK
Pragma: no-cache
Content-Length: 53
X-XSS-Protection: 1; mode=block
Expires: 0
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Content-Type: application/json;charset=UTF-8
Cache-Control: no-cache, no-store, max-age=0, must-revalidate

{
  "tokenId" : "1",
  "domainId" : "1111111111"
}
```

3.3. User Management Service

The user management service provides APIs for authentication and user management. The default port for the user management service is 15551

3.3.1. Authentication

Using the rest api, you can authenticate users given a username and password.

A post request authenticates a user on a domain.

Request Fields

Path	Type	Description
`username`	`String`	The user identification
`password`	`String`	The user's password

Path Parameters

Table 4. /domains/{domainId}/authenticate

Parameter	Description
`domainId`	The user authentication domain id

Example:

```
POST /domains/test-app/authenticate HTTP/1.1
Content-Type: application/json
domain-auth-token: 1234
Host: localhost:8080
Content-Length: 63

{
  "username" : "test-user",
  "password" : "my-password"
}
```

Response Fields

This API returns a JSON response with the following fields:

Path	Type	Description
`status`	`String`	The authentication status. One of the following: AUTHENTICATED, USERNAME_INVALID, PASSWORD_INVALID

Example:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Length: 34

{
  "status" : "AUTHENTICATED"
}
```

3.3.2. Managing Domains

A client application references users in one domain. A domain is referenced by a domain id. Applications can share the same user base by using the same domain id.

Security

If token security is enabled, the domain api can be accessed with the admin token.

Token security is enabled by default. To disable, set the environment property `mmadu.domain.api-security-enabled` to `false`.

Creating A domain

A POST request will create a domain

Request Fields

Path	Type	Description
`name`	`String`	The domain name
`id`	`String`	ID of the domain (optional, auto-generated)

Example:

```
POST /appDomains HTTP/1.1
domain-auth-token: 2222
Host: localhost:8080
Content-Length: 51

{
  "id" : "00111111",
  "name" : "new-domain"
}
```

Response Fields

This API returns a HTTP 201 CREATED response with an empty body.

Getting a domain with an ID

A GET request will get a domain with an ID.

Path Parameters

Table 5. /appDomains/{domainId}

Parameter	Description
`domainId`	The domain ID

Example:

```
GET /appDomains/00111111 HTTP/1.1
domain-auth-token: 2222
Host: localhost:8080
```

Response Fields

This API returns an AppDomain with these fields:

Path	Type	Description
`name`	`String`	The domain name
`_links`	`map`	Domain item resource links

Example:

```
HTTP/1.1 200 OK
Content-Type: application/hal+json;charset=UTF-8
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Length: 223

{
  "name" : "new-domain",
  "_links" : {
    "self" : {
      "href" : "http://localhost:8080/appDomains/00111111"
    },
    "appDomain" : {
      "href" : "http://localhost:8080/appDomains/00111111"
    }
  }
}
```

Getting All Domains

A GET request will get all domains.

Example:

```
GET /appDomains HTTP/1.1
domain-auth-token: 2222
Host: localhost:8080
```

Response Fields

This API returns a list of all domains with these fields:

Path	Type	Description
_embedded.appDomains.[].name	String	The name of the domain
_embedded.appDomains.[]._links	map	Domain item resource links
_links	map	Resource links
page	map	Page information

Example:

```
HTTP/1.1 200 OK
Content-Type: application/hal+json;charset=UTF-8
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Length: 1421

{
  "_embedded" : {
    "appDomains" : [ {
      "name" : "global-config",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8080/appDomains/0"
        },
        "appDomain" : {
          "href" : "http://localhost:8080/appDomains/0"
        }
      }
    }, {
      "name" : "test",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8080/appDomains/1"
        },
        "appDomain" : {
          "href" : "http://localhost:8080/appDomains/1"
        }
      }
    }, {
      "name" : "test",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8080/appDomains/test-app"
        },
        "appDomain" : {
          "href" : "http://localhost:8080/appDomains/test-app"
```

```
        }
      }
    }, {
      "name" : "Test domain For Config",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8080/appDomains/1111111111"
        },
        "appDomain" : {
          "href" : "http://localhost:8080/appDomains/1111111111"
        }
      }
    } ]
  },
  "_links" : {
    "self" : {
      "href" : "http://localhost:8080/appDomains{?page,size,sort}",
      "templated" : true
    },
    "profile" : {
      "href" : "http://localhost:8080/profile/appDomains"
    }
  },
  "page" : {
    "size" : 20,
    "totalElements" : 4,
    "totalPages" : 1,
    "number" : 0
  }
}
```

Updating A Domain with an ID

A PATCH request will update a domain with an ID.

Path Parameters

Table 6. /appDomains/{domainId}

Parameter	Description
`domainId`	The domain ID

Example:

```
PATCH /appDomains/00111111 HTTP/1.1
Content-Type: application/json
domain-auth-token: 2222
Host: localhost:8080
Content-Length: 23

{"name":"changed-name"}
```

Response Fields

This API returns a HTTP 204 NO CONTENT response.

Example:

```
HTTP/1.1 204 No Content
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
```

Removing A Domain

A DELETE request will remove a domain with an ID.

Path Parameters

Table 7. /appDomains/{domainId}

Parameter	Description
`domainId`	The domain ID

Example:

```
DELETE /appDomains/00111111 HTTP/1.1
domain-auth-token: 2222
Host: localhost:8080
```

Response Fields

This API returns a HTTP 204 NO CONTENT response.

Example:

```
HTTP/1.1 204 No Content
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
```

3.3.3. Users

This contains user information for a specific user. A user could have a username, password, roles, authorities and any other properties that the client deems fit.

3.3.4. Creating A User

A POST request will create a user.

Fixed Request Fields

Path	Type	Description
`username`	`String`	The user's username (must be unique)
`id`	`String`	The user's id (unique identifier used to reference user in your application)
`password`	`String`	The user's password
`roles`	`Array`	The user's assigned roles
`authorities`	`Array`	The user's granted authorities

Example:

```
POST /domains/test-app/users HTTP/1.1
Content-Length: 116
Content-Type: application/json
domain-auth-token: 1234
Host: localhost:8080

{"id":"123","username":"user","password":"password","roles":["admin"],"authorities":["
manage-users"],"color":"blue"}
```

As seen in the example, you can also add custom properties like the "color" property.

Response Fields

This API returns a HTTP 201 CREATED response with an empty body.

3.3.5. Getting A User with an ID

A GET request will get a user with an ID.

Path Parameters

Table 8. /domains/{domainId}/users/{userId}

Parameter	Description
`userId`	The user's ID
`domainId`	The domain id of the user

Example:

```
GET /domains/test-app/users/123453432 HTTP/1.1
domain-auth-token: 1234
Host: localhost:8080
```

Response Fields

This API returns an AppUser with these fields as well as custom fields provided by the client:

Path	Type	Description
`id`	`String`	The user's id
`username`	`String`	Username of the user
`password`	`String`	password of the user
`roles`	`string list`	List of roles assigned to this user
`authorities`	`string list`	List of authorities given to ths user

Example:

```
HTTP/1.1 200 OK
Pragma: no-cache
X-XSS-Protection: 1; mode=block
Expires: 0
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Content-Length: 159
Content-Type: application/json;charset=UTF-8
Cache-Control: no-cache, no-store, max-age=0, must-revalidate

{"id":"123453432","username":"test-user","password":"my-password","roles":["admin-
role"],"authorities":["admin"],"favourite-colour":"blue","country":"Nigeria"}
```

As you can see, this user has two extra properties: `favourite-color` and `country`.

3.3.6. Getting All Users In a Domain

A GET request will get all users regardless of domains.

Request Parameters

The request includes page information to request for particular sets in the list.

Parameter	Description
`page`	page number to request
`size`	maximum number of items in page

Example:

```
GET /domains/test-app/users?page=0&size=10 HTTP/1.1
domain-auth-token: 1234
Host: localhost:8080
```

Response Fields

This API returns a list of all users with the fields below. The response also has fields that display the page information of result. These fields are shown in the example response and are self explanatory.

Path	Type	Description
`content.[].id`	`String`	The user's unique identification
`content.[].username`	`String`	Username of the user
`content.[].password`	`String`	password of the user
`content.[].roles`	`string list`	List of roles assigned to this user
`content.[].authorities`	`string list`	List of authorities given to ths user

Example:

```
HTTP/1.1 200 OK
Pragma: no-cache
X-XSS-Protection: 1; mode=block
Expires: 0
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Content-Type: application/json;charset=UTF-8
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Length: 663

{"content":[{"id":"123453432","username":"test-user0","password":"my-
password0","roles":["admin-
role"],"authorities":["admin"],"country":"Nigeria","favourite-
color":"blue"},{"id":"123453432","username":"test-user1","password":"my-
password1","roles":["admin-
role"],"authorities":["admin"],"country":"Nigeria","favourite-
color":"blue"},{"id":"123453432","username":"test-user2","password":"my-
password2","roles":["admin-
role"],"authorities":["admin"],"country":"Nigeria","favourite-
color":"blue"}],"totalElements":3,"last":true,"totalPages":1,"size":10,"number":0,"num
berOfElements":3,"sort":{"sorted":false,"unsorted":true,"empty":true},"first":true,"em
pty":false}
```

3.3.7. Removing A User with an ID

A DELETE request will remove a user with an ID.

Path Parameters

Table 9. /domains/{domainId}/users/{userId}

Parameter	Description
`domainId`	The user's domain ID
`userId`	The user's ID

Example:

```
DELETE /domains/test-app/users/123453432 HTTP/1.1
domain-auth-token: 1234
Host: localhost:8080
```

Response Fields

This API returns a HTTP 204 NO CONTENT response.

Example:

```
HTTP/1.1 204 No Content
Pragma: no-cache
X-XSS-Protection: 1; mode=block
Expires: 0
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
```

3.3.8. Updating A User with an ID

A PUT request will update a user with an ID. Not, this api will completely overwrite the properties of the existing user. There are no partial updates.

Path Parameters

Table 10. /domains/{domainId}/users/{userId}

Parameter	Description
`domainId`	The user's domain ID
`userId`	The user's ID

Example:

```
PUT /domains/test-app/users/123453432 HTTP/1.1
Content-Length: 171
Content-Type: application/json
domain-auth-token: 1234
Host: localhost:8080

{"id":"123453432","username":"changed-username","password":"changed-
password","roles":["admin-role"],"authorities":["admin"],"favourite-
colour":"blue","country":"Nigeria"}
```

Response Fields

This API returns a HTTP 204 NO CONTENT response.

Example:

```
HTTP/1.1 204 No Content
Pragma: no-cache
X-XSS-Protection: 1; mode=block
Expires: 0
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
```

3.3.9. Getting A User with a username and domain id

A GET request will get a user with the specified username and domain id.

Path Parameters

Table 11. /domains/{domainId}/users/load

Parameter	Description
`domainId`	The domain id of the user

Request Parameters

Parameter	Description
`username`	The username of the user

Example:

```
GET /domains/test-app/users/load?username=test-user HTTP/1.1
domain-auth-token: 1234
Host: localhost:8080
```

Response Fields

This API returns an AppUser with the fields below and other custom fields provided by the domain client.

Path	Type	Description
`id`	`String`	The user's id
`username`	`String`	Username of the user
`password`	`String`	password of the user
`roles`	`string list`	List of roles assigned to this user
`authorities`	`string list`	List of authorities given to ths user

Example:

```
HTTP/1.1 200 OK
Pragma: no-cache
X-XSS-Protection: 1; mode=block
Expires: 0
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Content-Length: 159
Content-Type: application/json;charset=UTF-8
Cache-Control: no-cache, no-store, max-age=0, must-revalidate

{"id":"123453432","username":"test-user","password":"my-password","roles":["admin-
role"],"authorities":["admin"],"favourite-colour":"blue","country":"Nigeria"}
```

3.3.10. Querying Users

A GET request can search for users by username, and other custom fields created by the client.

NOTE	Query string can only support custom string, integer and boolean fields currently.
WARNING	Ensure to put individual criteria in parenthesis

Request Parameters

The request includes page information to request for particular sets in the list.

Parameter	Description
`query`	The query search string. Use any of your custom properties for this search including username
`page`	page number to request
`size`	maximum number of items in page

Example:

```
GET /domains/test-
app/users/search?page=0&size=10&query=%28country+equals+%27Nigeria%27%29+and+%28favour
ite-color+equals+%27blue%27%29 HTTP/1.1
domain-auth-token: 1234
Host: localhost:8080
```

Response Fields

This API returns a list of all users with the fields below. The response also has fields that display the page information of result. These fields are shown in the example response and are self explanatory.

Path	Type	Description
content.[].id	String	The user's unique identification
content.[].username	String	Username of the user
content.[].password	String	password of the user
content.[].roles	string list	List of roles assigned to this user
content.[].authorities	string list	List of authorities given to ths user

Example:

```
HTTP/1.1 200 OK
Pragma: no-cache
X-XSS-Protection: 1; mode=block
Expires: 0
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Content-Type: application/json;charset=UTF-8
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Length: 663

{"content":[{"id":"123453432","username":"test-user0","password":"my-
password0","roles":["admin-
role"],"authorities":["admin"],"country":"Nigeria","favourite-
color":"blue"},{"id":"123453432","username":"test-user1","password":"my-
password1","roles":["admin-
role"],"authorities":["admin"],"country":"Nigeria","favourite-
color":"blue"},{"id":"123453432","username":"test-user2","password":"my-
password2","roles":["admin-
role"],"authorities":["admin"],"country":"Nigeria","favourite-
color":"blue"}],"totalElements":3,"last":true,"totalPages":1,"size":10,"number":0,"num
berOfElements":3,"sort":{"sorted":false,"unsorted":true,"empty":true},"first":true,"em
pty":false}
```

3.3.11. Partial User update

A PATCH request can update users partially based on a query criteria

NOTE	Query string can only support custom string, integer and boolean fields currently.
WARNING	Ensure to put individual criteria in parenthesis

Path Parameters

Table 12. /domains/{domainId}/users

Parameter	Description
domainId	The domain id of the user

Request

The PATCH request accepts a query string and an array of update operations.

Path	Type	Description
query	String	The query criteria for updating users
updates.[].operation	String	The kind of update operation to make: (SET, INCREMENT, ADD, REMOVE
updates.[].property	String	The property to update
updates.[].value	String	The value used by the update operation

Example:

```
PATCH /domains/test-app/users HTTP/1.1
Content-Length: 105
Content-Type: application/json
domain-auth-token: 1234
Host: localhost:8080

{"query":"(country equals
'Nigeria')","updates":[{"operation":"SET","property":"color","value":"green"}]}
```

Response Fields

The API returns 204 No content for a successful update.

Example:

```
HTTP/1.1 204 No Content
Pragma: no-cache
X-XSS-Protection: 1; mode=block
Expires: 0
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
```

3.4. Registration Service

The registration service provides an api to access the registration page for a domain. The registration page generates a form from the configured fields and field types of a domain.

3.4.1. Displaying the registration form

The registration form can be accessed in the browser using the url:

```
http://[registration-host]:[registration-port]/[domain-id]/register
```

For example, accessing the default domain with the default configuration can be done by entering the url below on your browser.

```
http://localhost:1552/0/register
```

The registration services redirects to the defaultRedirectionUrl specified in the registration profile for this domain. To specify a url to redirect to, add the redirectUrl parameter

```
http://localhost:1552/0/register?redirectUrl=http://myapp.com
```

3.4.2. Customizing Registration Forms

Registration forms for a domain can be customized by adding and removing fields and field types. The registration service ensures that form fields are created for each field.

3.4.3. Registration Profiles

Each domain has a registration profile object that is used for configuring the applications behaviour during registration.

NOTE	There can be only one registration profile for a domain.

Creating a Registration Profile

A POST Request will create a registration profile.

Fixed Request Fields

Path	Type	Description
`id`	`string`	Registration profile ID (auto generated if absent)
`domainId`	`String`	domain ID

Path	Type	Description
defaultRoles	Array	Roles assigned to newly registered users
defaultAuthorities	Array	Authorities assigned to newly registered users
defaultRedirectUrl	String	Url to redirect to after registration success
headerOne	String	Main registration page title - h1
headerTwo	String	Secondary registration page title - h2
headerThree	String	Tertiary registration page title - h3
instruction	String	Instruction message for registering users displayed at the top of the form
submitButtonTitle	String	Submit button text

Example:

```
POST /repo/registrationProfiles HTTP/1.1
domain-auth-token: 2222
Content-Length: 358
Host: localhost:8080

{
  "id" : "1",
  "domainId" : "0",
  "defaultRedirectUrl" : "http://my.app.com/home",
  "defaultRoles" : [ "member" ],
  "defaultAuthorities" : [ "view-list" ],
  "headerOne" : "My App",
  "headerTwo" : "Register",
  "headerThree" : "Fill all required fields",
  "instruction" : "Ensure that all fields are filled",
  "submitButtonTitle" : "Go"
}
```

Response Fields

This API returns a HTTP 201 CREATED response with an empty body.

Obtaining Registration Profile by Profile ID

A GET Request will return a registration profile with its ID

Path Parameters

Table 13. /repo/registrationProfiles/{profileId}

Parameter	Description
`profileId`	The registration profile ID

Example:

```
GET /repo/registrationProfiles/1 HTTP/1.1
domain-auth-token: 2222
Host: localhost:8080
```

Response Fields

This API returns the registration profile with the following fields:

Path	Type	Description
`id`	`string`	Registration profile ID (auto generated if absent)
`domainId`	`String`	domain ID
`defaultRoles`	`Array`	Roles assigned to newly registered users
`defaultAuthorities`	`Array`	Authorities assigned to newly registered users
`defaultRedirectUrl`	`String`	Url to redirect to after registration success
`headerOne`	`String`	Main registration page title - h1
`headerTwo`	`String`	Secondary registration page title - h2
`headerThree`	`String`	Tertiary registration page title - h3
`instruction`	`String`	Instruction message for registering users displayed at the top of the form
`submitButtonTitle`	`String`	Submit button text

Example:

```
HTTP/1.1 200 OK
Pragma: no-cache
X-XSS-Protection: 1; mode=block
Expires: 0
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Content-Type: application/hal+json;charset=UTF-8
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Length: 563

{
  "domainId" : "0",
  "defaultRedirectUrl" : "http://my.app.com/home",
  "defaultRoles" : [ "member" ],
  "defaultAuthorities" : [ "view-list" ],
  "headerOne" : "My App",
  "headerTwo" : "Register",
  "headerThree" : "Fill all required fields",
  "instruction" : "Ensure that all fields are filled",
  "submitButtonTitle" : "Go",
  "_links" : {
    "self" : {
      "href" : "http://localhost:8080/repo/registrationProfiles/1"
    },
    "registrationProfile" : {
      "href" : "http://localhost:8080/repo/registrationProfiles/1"
    }
  }
}
```

Updating Registration Profile by ID

A PATCH Request will update a registration profile. The example below updates the defaultRegistrationUrl property.

Path Parameters

Table 14. /repo/registrationProfiles/{profileId}

Parameter	Description
`profileId`	The registration profile ID

Example:

```
PATCH /repo/registrationProfiles/1 HTTP/1.1
domain-auth-token: 2222
Host: localhost:8080
Content-Length: 48

{"defaultRedirectUrl":"http://modified.app.com"}
```

Response Fields

This API returns a HTTP 204 NO CONTENT response.

Example:

```
HTTP/1.1 204 No Content
Pragma: no-cache
X-XSS-Protection: 1; mode=block
Expires: 0
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
```

Deleting a Registration Profile by ID

A DELETE request will delete a registration profile.

WARNING	Deleting a registration profile will remote all fields associated with the profile and a registration page for that domain will be unavailable.

Path Parameters

Table 15. /repo/registrationProfiles/{profileId}

Parameter	Description
`profileId`	The registration profile ID

Example:

```
DELETE /repo/registrationProfiles/1 HTTP/1.1
domain-auth-token: 2222
Host: localhost:8080
```

Response Fields

This API returns a HTTP 204 NO CONTENT response.

Example:

```
HTTP/1.1 204 No Content
Pragma: no-cache
X-XSS-Protection: 1; mode=block
Expires: 0
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
```

3.4.4. Field Types

Field type objects specify the properties of a field type in the registration form. You can create a field type that can be used by fields in a registration form.

See Defining Markup for fields for more information on how to customize field types.

Creating a field Type

A POST request will create a field type

Request Fields

Path	Type	Description
id	string	Field Type ID
fieldTypePattern	String	Allowed string pattern for field type
min	String	Minimum value for field type
max	String	Maximum value for field type
style	Null	Style applied to all fields of the type
type	String	Data type of to field type (integer, date, time, datetime, string, decimal)
css	String	css for all fields with this type (can be overriden by field css)
script	String	Script for field
classes	Array	css classes for field
markup	String	HTML markup for field type
name	String	The field type name
enclosingElement	String	The element enclosing the field

Example:

```
POST /repo/fieldTypes HTTP/1.1
domain-auth-token: 2222
Host: localhost:8080
Content-Length: 521

{
  "id" : "1",
  "name" : "Age",
  "markup" : "<label for='$field.name' class='sr-only'>$field.label</label><input
type='number' id='$field.name' name='$field.name' class='form-control'
placeholder='$field.placeholder' $maxValue $minValue autofocus $required $inputField
$inputStyle $errorStyle >$errorDisplay",
  "fieldTypePattern" : "",
  "type" : "integer",
  "enclosingElement" : "div",
  "classes" : [ "form-control" ],
  "style" : null,
  "script" : "",
  "css" : "",
  "max" : "100",
  "min" : "10"
}
```

Response Fields

This API returns a HTTP 201 CREATED response with an empty body.

Getting a field type with an ID

A GET request will get a field type with an ID.

Path Parameters

Table 16. /repo/fieldTypes/{fieldTypeId}

Parameter	Description
`fieldTypeId`	The field type ID

Example:

```
GET /repo/fieldTypes/1 HTTP/1.1
domain-auth-token: 2222
Host: localhost:8080
```

Response Fields

This API returns a FieldType object with these fields:

Path	Type	Description
`id`	`string`	Field Type ID
`fieldTypePattern`	`String`	Allowed string pattern for field type
`min`	`String`	Minimum value for field type
`max`	`String`	Maximum value for field type
`style`	`Null`	Style applied to all fields of the type
`type`	`String`	Data type of to field type (integer, date, time, datetime, string, decimal)
`css`	`String`	css for all fields with this type (can be overriden by field css)
`script`	`String`	Script for field
`classes`	`Array`	css classes for field
`markup`	`String`	HTML markup for field type
`name`	`String`	The field type name
`enclosingElement`	`String`	The element enclosing the field

Example:

```
HTTP/1.1 200 OK
Pragma: no-cache
X-XSS-Protection: 1; mode=block
Expires: 0
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Content-Type: application/hal+json;charset=UTF-8
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Content-Length: 696

{
  "name" : "Age",
  "markup" : "<label for='$field.name' class='sr-only'>$field.label</label><input
type='number' id='$field.name' name='$field.name' class='form-control'
placeholder='$field.placeholder' $maxValue $minValue autofocus $required $inputField
$inputStyle $errorStyle >$errorDisplay",
  "fieldTypePattern" : "",
  "type" : "integer",
  "enclosingElement" : "div",
  "classes" : [ "form-control" ],
  "style" : null,
  "script" : "",
  "css" : "",
  "max" : "100",
  "min" : "10",
  "_links" : {
    "self" : {
      "href" : "http://localhost:8080/repo/fieldTypes/1"
    },
    "fieldType" : {
      "href" : "http://localhost:8080/repo/fieldTypes/1"
    }
  }
}
```

Getting All Field Types

A GET request will get all field types.

Example:

```
GET /repo/fieldTypes HTTP/1.1
domain-auth-token: 2222
Host: localhost:8080
```

Response Fields

This API returns a list of all field types with these fields:

Path	Type	Description
_embedded.fieldTypes.[].fieldTypePattern	String	Allowed string pattern for field type
_embedded.fieldTypes.[].min	String	Minimum value for field type
_embedded.fieldTypes.[].max	String	Maximum value for field type
_embedded.fieldTypes.[].type	String	Data type of to field type (integer, date, time, datetime, string, decimal)
_embedded.fieldTypes.[].css	String	css for all fields with this type (can be overriden by field css)
_embedded.fieldTypes.[].script	String	Script for field
_embedded.fieldTypes.[].classes	Array	css classes for field
_embedded.fieldTypes.[].markup	String	HTML markup for field type
_embedded.fieldTypes.[].name	String	The field type name
_embedded.fieldTypes.[].enclosingElement	String	The element enclosing the field

Example:

```
HTTP/1.1 200 OK
Pragma: no-cache
X-XSS-Protection: 1; mode=block
Content-Length: 1170
Expires: 0
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Content-Type: application/hal+json;charset=UTF-8
Cache-Control: no-cache, no-store, max-age=0, must-revalidate

{
  "_embedded" : {
    "fieldTypes" : [ {
      "name" : "Age",
      "markup" : "<label for='$field.name' class='sr-only'>$field.label</label><input
type='number' id='$field.name' name='$field.name' class='form-control'
placeholder='$field.placeholder' $maxValue $minValue autofocus $required $inputField
$inputStyle $errorStyle >$errorDisplay",
      "fieldTypePattern" : "",
      "type" : "integer",
      "enclosingElement" : "div",
      "classes" : [ "form-control" ],
      "style" : null,
      "script" : "",
      "css" : "",
```

```
      "max" : "100",
      "min" : "10",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8080/repo/fieldTypes/1"
        },
        "fieldType" : {
          "href" : "http://localhost:8080/repo/fieldTypes/1"
        }
      }
    } ]
  },
  "_links" : {
    "self" : {
      "href" : "http://localhost:8080/repo/fieldTypes{?page,size,sort}",
      "templated" : true
    },
    "profile" : {
      "href" : "http://localhost:8080/repo/profile/fieldTypes"
    }
  },
  "page" : {
    "size" : 20,
    "totalElements" : 1,
    "totalPages" : 1,
    "number" : 0
  }
}
```

Updating A Field Type

A PATCH request will update a field type

Path Parameters

Table 17. /repo/fieldTypes/{fieldTypeId}

Parameter	Description
`fieldTypeId`	The field type ID

Example:

```
PATCH /repo/fieldTypes/1 HTTP/1.1
domain-auth-token: 2222
Content-Length: 19
Host: localhost:8080

{"name":"New Type"}
```

Response Fields

This API returns a HTTP 204 NO CONTENT response.

Example:

```
HTTP/1.1 204 No Content
Pragma: no-cache
X-XSS-Protection: 1; mode=block
Expires: 0
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
```

Removing A Field Type

A DELETE request will remove a field type.

Path Parameters

Table 18. /repo/fieldTypes/{fieldTypeId}

Parameter	Description
`fieldTypeId`	The field type ID

Example:

```
DELETE /repo/fieldTypes/1 HTTP/1.1
domain-auth-token: 2222
Host: localhost:8080
```

Response Fields

This API returns a HTTP 204 NO CONTENT response.

Example:

```
HTTP/1.1 204 No Content
Pragma: no-cache
X-XSS-Protection: 1; mode=block
Expires: 0
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
```

3.4.5. Fields

Fields contain information used in rendering each field in a registration form. Fields utilize

properties of their field types in order to render an adequate representation of the field. Fields also specify the user property that the input field is bound to. Fields can override and extend certain properties of the field types.

See Defining Markup for fields for more information on how to customize fields.

Creating a field

A POST request will create a field

Request Fields

Path	Type	Description
`id`	`string`	Field ID
`pattern`	`String`	Allowed string pattern for field
`style`	`String`	css style for input element in field
`label`	`String`	Input field label
`placeholder`	`String`	input field placeholder
`fieldTypeId`	`String`	Id of the field's type
`property`	`String`	User property that the field will set
`order`	`Number`	Order of the field in Layout
`required`	`Boolean`	If the field input is required or not
`name`	`String`	the form name of the field
`domainId`	`String`	the ID of the domain

Example:

```
POST /repo/fields HTTP/1.1
Content-Length: 251
domain-auth-token: 2222
Host: localhost:8080

{
  "id" : "1",
  "domainId" : "1",
  "name" : "username",
  "placeholder" : "Enter Username",
  "property" : "username",
  "fieldTypeId" : "1",
  "style" : "",
  "label" : "Username",
  "order" : 1,
  "pattern" : "",
  "required" : true
}
```

Response Fields

This API returns a HTTP 201 CREATED response with an empty body.

Getting a field type with an ID

A GET request will get a field with an ID.

Path Parameters

Table 19. /repo/fields/{fieldId}

Parameter	Description
`fieldId`	The field ID

Example:

```
GET /repo/fields/1 HTTP/1.1
domain-auth-token: 2222
Host: localhost:8080
```

Response Fields

This API returns a Field object with these fields:

Path	Type	Description
`id`	`string`	Field ID
`pattern`	`String`	Allowed string pattern for field

Path	Type	Description
`style`	`String`	css style for input element in field
`label`	`String`	Input field label
`placeholder`	`String`	input field placeholder
`fieldTypeId`	`String`	Id of the field's type
`property`	`String`	User property that the field will set
`order`	`Number`	Order of the field in Layout
`required`	`Boolean`	If the field input is required or not
`name`	`String`	the form name of the field
`domainId`	`String`	the ID of the domain

Example:

```
HTTP/1.1 200 OK
Pragma: no-cache
X-XSS-Protection: 1; mode=block
Expires: 0
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Content-Type: application/hal+json;charset=UTF-8
Content-Length: 414
Cache-Control: no-cache, no-store, max-age=0, must-revalidate

{
  "domainId" : "1",
  "name" : "username",
  "placeholder" : "Enter Username",
  "property" : "username",
  "fieldTypeId" : "1",
  "style" : "",
  "label" : "Username",
  "order" : 1,
  "pattern" : "",
  "required" : true,
  "_links" : {
    "self" : {
      "href" : "http://localhost:8080/repo/fields/1"
    },
    "field" : {
      "href" : "http://localhost:8080/repo/fields/1"
    }
  }
}
```

Getting All Fields in Domain

A GET request will get all fields in a domain.

Request Parameters

Parameter	Description
`domainId`	Domain ID of the request

Example:

```
GET /repo/fields/search/findByDomainId?domainId=1 HTTP/1.1
domain-auth-token: 2222
Host: localhost:8080
```

Response Fields

This API returns a list of all fields with these properties:

Path	Type	Description
`_embedded.fields.[].pattern`	`String`	Allowed string pattern for field
`_embedded.fields.[].style`	`String`	css style for input element in field
`_embedded.fields.[].label`	`String`	Input field label
`_embedded.fields.[].placeholder`	`String`	input field placeholder
`_embedded.fields.[].fieldTypeId`	`String`	Id of the field's type
`_embedded.fields.[].property`	`String`	User property that the field will set
`_embedded.fields.[].order`	`Number`	Order of the field in Layout
`_embedded.fields.[].required`	`Boolean`	If the field input is required or not
`_embedded.fields.[].name`	`String`	the form name of the field
`_embedded.fields.[].domainId`	`String`	the ID of the domain

Example:

```
HTTP/1.1 200 OK
Pragma: no-cache
X-XSS-Protection: 1; mode=block
Expires: 0
X-Frame-Options: DENY
Content-Length: 658
X-Content-Type-Options: nosniff
Content-Type: application/hal+json;charset=UTF-8
Cache-Control: no-cache, no-store, max-age=0, must-revalidate

{
  "_embedded" : {
    "fields" : [ {
      "domainId" : "1",
      "name" : "username",
      "placeholder" : "Enter Username",
      "property" : "username",
      "fieldTypeId" : "1",
      "style" : "",
      "label" : "Username",
      "order" : 1,
      "pattern" : "",
      "required" : true,
      "_links" : {
        "self" : {
          "href" : "http://localhost:8080/repo/fields/1"
        },
        "field" : {
          "href" : "http://localhost:8080/repo/fields/1"
        }
      }
    } ]
  },
  "_links" : {
    "self" : {
      "href" : "http://localhost:8080/repo/fields/search/findByDomainId"
    }
  }
}
```

Updating A Field

A PATCH request will update a field

Path Parameters

Table 20. /repo/fields/{fieldId}

Parameter	Description
`fieldId`	The field ID

Example:

```
PATCH /repo/fields/1 HTTP/1.1
domain-auth-token: 2222
Content-Length: 34
Host: localhost:8080

{"placeholder":"Enter a Username"}
```

Response Fields

This API returns a HTTP 204 NO CONTENT response.

Example:

```
HTTP/1.1 204 No Content
Pragma: no-cache
X-XSS-Protection: 1; mode=block
Expires: 0
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
```

Removing A Field

A DELETE request will remove a field.

Path Parameters

Table 21. /repo/fields/{fieldId}

Parameter	Description
`fieldId`	The field ID

Example:

```
PATCH /repo/fields/1 HTTP/1.1
domain-auth-token: 2222
Content-Length: 34
Host: localhost:8080

{"placeholder":"Enter a Username"}
```

Response Fields

This API returns a HTTP 204 NO CONTENT response.

Example:

```
HTTP/1.1 204 No Content
Pragma: no-cache
X-XSS-Protection: 1; mode=block
Expires: 0
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
```

3.4.6. Defining Markup for Fields

You define how you want your registration fields by configuring fields and field types. The markup property in the field type is used to define the markup of that specific field type. An age field type markup can be defined as shown:

```
<label for='$field.name' class='sr-only'>$field.label</label>
<input type='number' id='$field.name' name='$field.name' class='form-control'
placeholder='$field.placeholder' $maxValue $minValue autofocus $required $inputField
$inputStyle $errorStyle >
$errorDisplay"
```

Mmadu registration service uses apache velocity to generate markup for fields. It provies the following in the template context for use in defining fields. You reference context objects by adding the `$` prefix. For example, the `$field.name` returns the name property in the field object.

Object Name	Description
field	The field object
type	The field type object
inputField	Input attribute and value that marks an element as a model input
inputStyle	style attribute, any element with this property will have the field type style property appended to that element
errorStyle	style attribute, any element with this property will have the style appended with the error style when the field contains errors
errorDisplay	HTML markup for displaying field errors

Object Name	Description
maxValue	max attribute, any element with this will have the max attribute set to the max value of the field type
minValue	min attribute, any element with this will have the min attribute set to the min value of the field type