



APRIL 10, 2018

USER MANUAL

MID – MY IDENTITY

CILLIAN MC NEILL - 14352621

SCHOOL OF COMPUTING

SUPERVISOR: GEOFF HAMILTON

Table of Contents

INTRODUCTION	2
INSTALLATION INSTRUCTIONS	3
OVERVIEW	4
<i>Hardware Requirements</i>	<i>4</i>
<i>Software Prerequisites</i>	<i>4</i>
BACKEND APPLICATION	5
Overview	5
Application Configuration	5
Building Hyperledger.....	6
Building Application	6
Testing Application	6
MOBILE APPLICATION	7
ADMIN CONSOLE	8
<i>Dummy Admin Backend.....</i>	<i>8</i>
<i>User Interface.....</i>	<i>8</i>
USAGE INSTRUCTIONS	9
MOBILE APPLICATION	10
Overview	10
Profile Creation	11
Identity Type Creation.....	11
Identity Type Submission	12
Information Request Creation.....	13
Information Review & Response	14
BACKEND APPLICATION	15
Overview	15
Swagger Interface	15
Backend Authentication.....	16
ADMIN CONSOLE	17
Overview	17
Party Creation	17
View Party.....	18
View Submission	18
Create Identity Type.....	19
Update Identity Type.....	19

Introduction

This document outlines the steps taken to install and use the MiD application and the various platforms that it has been deployed to.

Installation is dependent on the platform that you are deploying to. It's important to note that the Mobile application and Admin console will not function without the Backend application. Each installation has its own set of prerequisites before installing and should be adhered to strictly to ensure proper performance of the application.

Usage of the application will vary depending on the platform. Instructions and accompanying screenshots will be included for each platform. Note that the Admin console is for testing only and **should not** be used in a production environment. It will function as a good testing ground for the application but your own implementation of it (or an extended version) should be implemented.

Glossary

- **Blockchain**
 - A continuously growing list of records, called blocks, which are linked and secured using cryptography. Each block typically contains a link to a previous block along with any relevant data. Thanks to this and the overall design, blockchains are inherently resistant to modification of the data stored on them.
- **Distributed Ledger**
 - Replicated, synchronized and shared digital data that is spread across multiple locations.
- **Distributed Systems**
 - A model of computer networks in which the systems pass messages to one another to complete a task. The distribution of these systems allows for concurrent work to be done and without a single point of failure.
- **JSON**
 - A format that is easy for humans to read/write and computers to parse. Used in the transmission of messages across a network.
- **Hyperledger Fabric**
 - A branch of the Hyperledger Project (open source blockchain tools and distributed ledgers) originally created by IBM. It is a permissioned blockchain infrastructure using concepts such as roles between nodes and “smart contracts” to facilitate trading.
- **Node**
 - A JavaScript based development platform
- **APK – Android Package Kit**
 - A package file format for the Android operating system.
- **API – Application Programming Interface**
 - A set of defined methods to allow for communication between various software components.
- **SAFe – Scaled Agile Framework**
 - A form of the agile methodology employed in large organisations. It promotes collaboration and scalability with numerous teams.

Installation Instructions

Overview

The section will detail the steps you will have to take to install the application. No step should be skipped to ensure proper performance. We will break down the prerequisites necessary to run the application so please read over and install the software & libraries listed.

Hardware Requirements

Below lists the minimum hardware requirements for the application. It has been tested and is guaranteed working on the below minimum specification so be warned when deviating from this.

- Minimum Specifications
 - CPU: Intel Pentium
 - RAM: 4gb
 - HDD: 20GB free space
 - OS: Ubuntu Linux 14.04 / 16.04 LTS (both 64-bit), or Mac OS 10.12
- Recommended Specifications
 - CPU: Intel i5 or AMD equivalent
 - RAM: 8gb
 - HDD: 20GB free space
 - OS: Ubuntu Linux 14.04 / 16.04 LTS (both 64-bit), or Mac OS 10.12
- Mobile Requirements
 - Android Version: 5.0+
 - HDD: 10-20MB free space
 - Requires: Camera

Software Prerequisites

Below lists the software used to develop the application. If you wish to build the application yourself, you must have the below software:

- [Tomcat 9](#)
- [MySQL](#)
- [NodeJs](#)
- [Maven](#)
- [Hyperledger Framework](#)
 - [Prerequisites](#)
 - [MiD deployment Script](#)
- [Android Device](#)

Backend Application

Overview

The backend application of MiD is the hub through which all other applications within MiD communicate through. As such, it must be deployed before any other application is used. I have broken this section into what is required to build and deploy the application and what is required to run the included integration and unit tests.

Application Configuration

Database Configuration

Before you build the application, you must ensure that the application will connect to the correct database when it launches.

Within “src/main/resources” edit the application.properties file to point to the correct database. Update the “spring.datasource.url” variable to point to your database and add your username and password to “spring.datasource.username” and “spring.datasource.password” respectively.

```
## Spring DATASOURCE (DataSourceAutoConfiguration & DataSourceProperties)
spring.datasource.url =
jdbc:mysql://localhost:3306/mid?autoReconnect=true&useUnicode=true&characterEncoding=UTF-
8&allowMultiQueries=true&useSSL=false&createDatabaseIfNotExist=true
spring.datasource.username = root
spring.datasource.password=Pa88w0rd

## Hibernate Properties
# The SQL dialect makes Hibernate generate better SQL for the chosen
database
spring.jpa.properties.hibernate.dialect =
org.hibernate.dialect.MySQL5Dialect
# Hibernate ddl auto (create, create-drop, validate, update)
spring.jpa.hibernate.ddl-auto=create
```

Hyperledger Configuration

Hyperledger is deployed by default to localhost:3000/api. This is set within mid.properties at “src/main/resources”. If you want to change where this is deployed to then update mid.hyperledger within this properties file to your new URL.

Google Firebase Configuration

Google Firebase is used to manage notifications to and from users and identifying parties. For simple integration of your instance of Google Firebase just edit the mid.fcm variable within mid.properties at “src/main/resources”.

```
mid.fcm=Insert your Google fcm here
mid.hyperledger=http://192.168.0.152:3000/api
```

Building Hyperledger

Hyperledger is the blockchain infrastructure that is used to store and return certificates. To start, you must build MiD's network definitions into a Business Network Archive (BNA) file and deploy it to the docker containers created when you set up the [Hyperledger Framework](#).

I have [created a small script](#) to create the BNA file and deploy it to the docker containers on that machine. Simply ensure the paths are correct within the file and run it. This will deploy the BNA file and create endpoints at localhost:3000/api for the backend application to use.

Note: These endpoints are only live if this script is running so be sure to create a different session for this script to run if you intend to shut the terminal down. This can be accomplished by running "screen" to start a new session, starting the script then pressing "CTRL+A" followed by "CTRL+D".

Building Application

This application makes use of maven to manage dependencies and build/test the entire application. If one wished to build the application, they must run one command through maven.

Within the route directory of "Backend" run the command:

```
mvn clean install DskipTests=true
```

This will run a maven goal that will build the project and package it into a war file that can be interpreted and deployed by Tomcat. Place it within the "webapps" folder to begin deployment

Testing Application

This application has two types of testing, unit and integration testing. Each one has their own way of running and should be adhered to if attempting to run them yourself. It's recommended to run all these tests before you attempt to use the backend so that you can ensure there are no errors with your deployment.

```
mvn test
```

Unit Testing

These are the tests that ensure basic functionality within the application. They are also used to ensure code coverage for all included files. These tests can be run with the maven goal:

```
mvn test -P unit
```

This will carry out the same goal as if you were trying to build the application but will also run the unit tests and return any errors that occur. Note that the project won't properly build here if all the tests do not pass.

Integration Testing

These tests are used to ensure all the components of the backend are communicating correctly and will run several scenarios against all the endpoints of the application.

This is done through a set of behavioural driven development cucumber tests (BDDs) that are written in plain English and interpreted into different java test methods.

To run these tests, you need to ensure that an instance of the application is running locally (this includes the Hyperledger deployment) and run the command:

```
mvn test -P int
```

All cucumber scenarios will be run with this command. On completion it will return whether all the tests were successful or not.

Mobile Application

This application is designed specifically for devices running Android 5.0+. There is currently no support for devices running a lower version.

Building this application is relatively simple. Move into the source directory of “Mobile Application” and run:

```
gradlew assembleDebug
```

This will build a debug APK that can be installed to any android device running Android 5.0+. You can generate a signed version of the application but this isn't necessary if you aren't deploying it to the play store.

Alternatively you can download the MiD application from [Google Play](#) and edit the settings so that the server endpoint is pointing at your installation of the application rather than the standard [mid-secure.ie](#).

Admin Console

Dummy Admin Backend

For the admin console to function we need to adhere to the MiD security policy. This means that for every user they need a corresponding public and private key. These keys will be used to encrypt and decrypt a token belonging to the user. The user will encrypt a secret token and send it with every request it makes to the server. The server will keep track of the user's public key and will decrypt an auth header containing the associated token and match it to what it has on file.

The admin console is a simple Angular interface and as such, it cannot create and keep track of the keys that the backend requires. As such, a dummy backend has been created to act as a middle man between the admin UI and MiD. It will wrap all requests made to MiD in the correct auth header for that user and keep track of all public/private keys.

Dummy Admin Backend Configuration

All files for this can be found within "Web Application/Web Backend". This backend, similar to the MiD backend, contains an application.properties file within "src/main/resources". From here, edit the file to point to the correct database. Update the "spring.datasource.url" variable to point to your database and add your username and password to "spring.datasource.username" and "spring.datasource.password" respectively.

Dummy Admin Backend Build

Like the MiD backend it can be built with a simple maven goal:

```
mvn clean install
```

This will run a maven goal that will build the project and package it into a war file that can be interpreted by Tomcat. This war file can be dropped into your tomcat instance's webapps folder and tomcat will deploy it.

User Interface

User Interface Configuration

Build User Interface

All files for this can be found within "Web Application/Web UI". This is the test admin console to ensure party functionality and **should not** be used as a production UI. It can be extended to work as a secure and functional UI, but it is recommended to integrate the MiD's endpoints into an existing UI of the identifying party's creation.

The interface can be deployed with a few simple commands. Assuming you have node, you can install the ng-cli with:

```
npm i ng-cli
```

Once you have the cli installed you can move into the Web Application/Web UI directory and run the below command:

```
npm install
```

This will install all the libraries necessary to build and deploy the admin console. It may take several minutes depending on your internet connection so give this time.

You now must ensure that the UI is pointing to the dummy backend so that it may begin to send a receive request. Failing to do this will result in calls not being secured and denied by MiD.

```
@Injectable()
export class Globals {
  endpoint: string = 'https://mid-secure.ie/mid-admin';
}
```

With all the libraries installed and the corrected endpoint configured, the UI can be deployed with:

```
ng serve
```

This will build and deploy the UI to localhost:4200 where you can begin making requests (assuming both the dummy backend and MiD backend are deployed. If you wish to build it into a set of files that can be deployed to something like Tomcat, run the following command:

```
ng build
```

This will build the files into the “/dist” folder. Edit the “index.html” page to use “.” as its base-href so that we use relative linking.

The folder can be renamed to a path you’d like to use on the server and dropped into Tomcat to being receiving requests.

Usage Instructions

Mobile Application

Overview

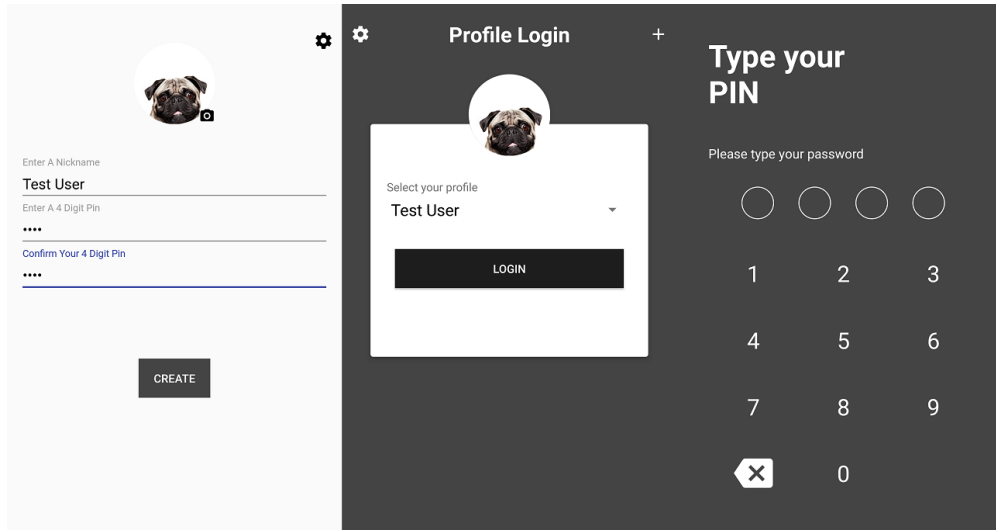
The mobile application is the user's link into MiD. It allows them to create and store identities to be verified by the corresponding parties. It also allows for the requesting of information from another user. All submissions/requests can be created and viewed through this application. In this section I will outline how one goes about making use of all the primary features of the application:

- Profile Creation
- Local Identity Creation
- Submission Creation
- Request Creation
- Request Review and Update

The application leverages the endpoint calls found within the MiD backend. Its possible for a third party to implement their own version of the UI as all we're doing is providing an easy access to the API calls for users.

Profile Creation

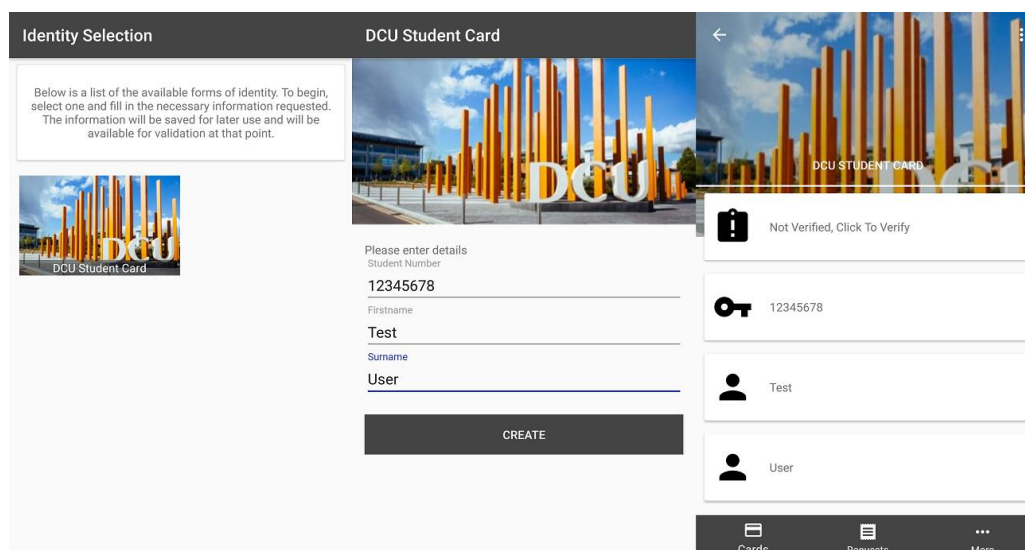
A user profile only requires a username and 4-digit pin. When the user enters in the requested information a profile is created locally and linked to on the server. With the profile created the user only needs to log in with the created pin. Note currently there is no support to change a user's pin, a profile can be deleted but the pin cannot be changed.



Identity Type Creation

Once a user has created a profile they need to create an initial identity type. The first screen is what you're brought to when you first log in. The app will get all current forms of identity and display them to the user. Once the user has selected a form of identity they will need to enter in the required information for that type.

Below we see an example of the DCU student card (a basic form of it). The user needs to enter their student number along with their first name and last name. Once they're happy with what they have entered they can tap "Create" and the identity will be saved. The final screen is the main page of the application. It shows all current forms of identity, their information and most important of all, its status. We can see that as we have just created the identity it has not been verified so we must do this before we can do anything else with this identity.



Identity Type Submission

If we wish to use this form of identity it must be verified by the identifying party that created it, which in this case is DCU. We must prove that we are the owner of the information being submitted. A user can do this by supplying the information requested by this type of identity and a current image of themselves.

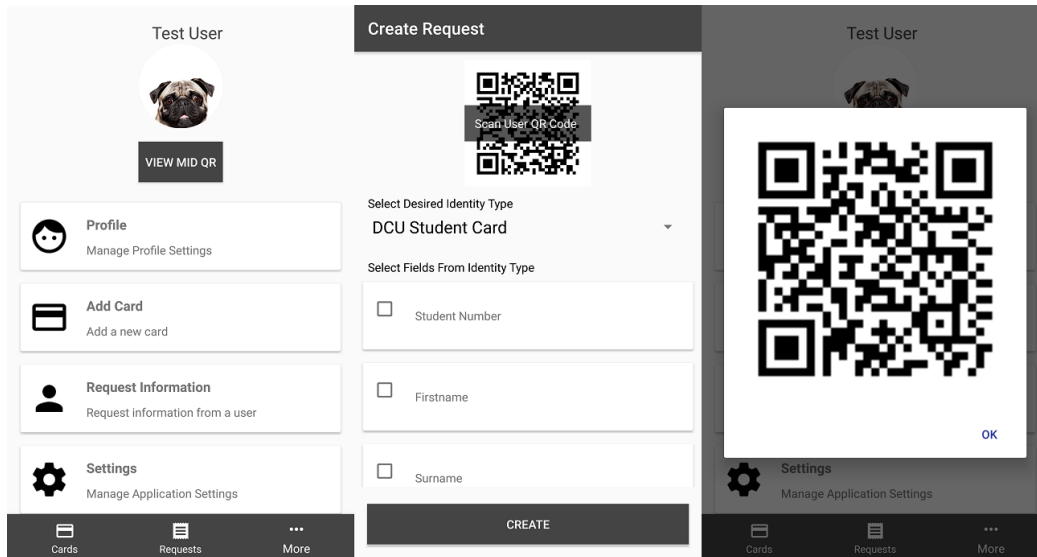
Above we see the initial submission creation process. The user double checks what they entered initially, takes a picture of themselves (above is a dummy image) and selects “Submit”. The submission is sent to MiD where it can be viewed by the staff within DCU. We are shown the created submission and its status.

In a real-world scenario these submissions could take multiple working days. The user can see the submission at any time under the “Requests” but when the submission has been accepted/rejected they will get a notification informing them of it.

Above we see the result of a processed application. The user receives a notification letting them know that it has been accepted. When the user taps this they are brought to the submission with the updated status and a link to the created certificate. This certificate is placed on the blockchain within MiD. Anybody can now verify that the information from this identity type is correct by using the reference linked to the submission for this identity type.

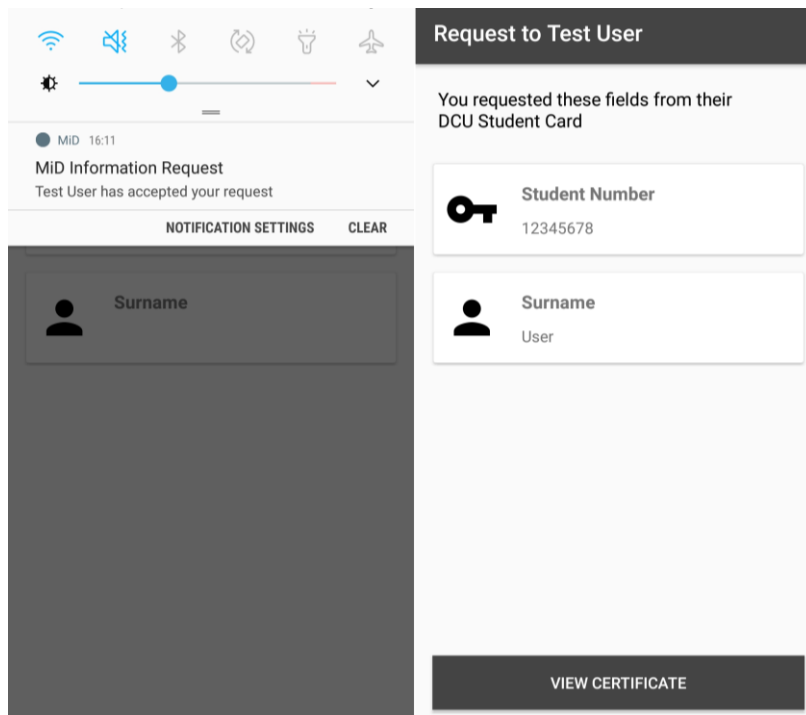
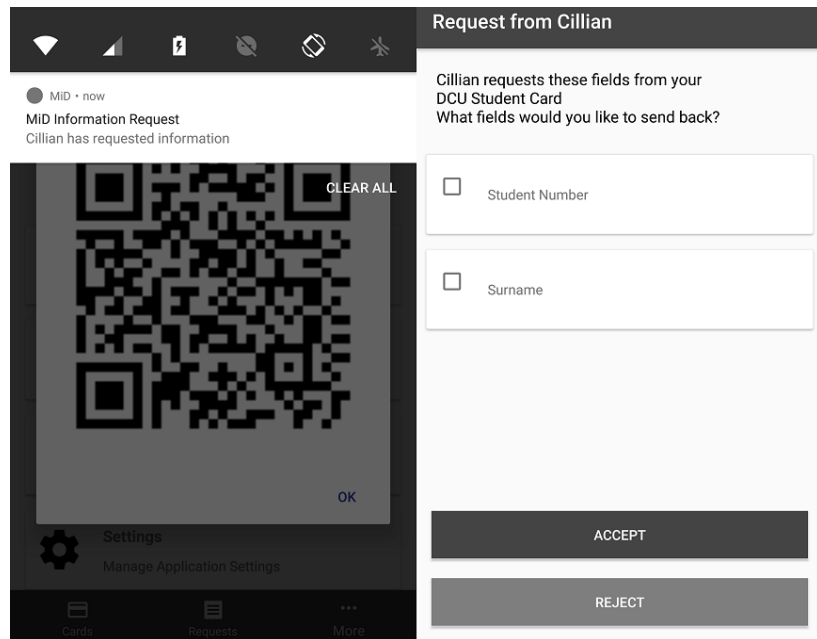
Information Request Creation

A user can have a verified identity requested from another user. With this it allows easy verification of one's identity either in person or online. To begin a request a user selects the "More" tab and taps "Request Information". They then select the identity type they want to request and then selects the fields they want to request from that information type. The user then scans the QR code of the user they want to request this information from. A request is then created, and the receiving user will get a notification.



Information Review & Response

The user will receive a notification letting them know that another user has requested information from them. They can review what is being requested and choose what fields they want to send back. It's important to note that you don't have to send back everything that is being requested. This is to give the user finer control over what is being given out to another user.



Above is the response that the sender will get once the receiver has reviewed what you have asked from them. We see that the user has accepted my request and has sent back the fields that have been asked for. An important thing to note here is that the request contains a link to the identities certificate. If this wasn't present, then the request isn't validated, so any information contained within could be incorrect or falsified.

Checks are made for request against the corresponding submissions to ensure falsification won't occur.

Backend Application

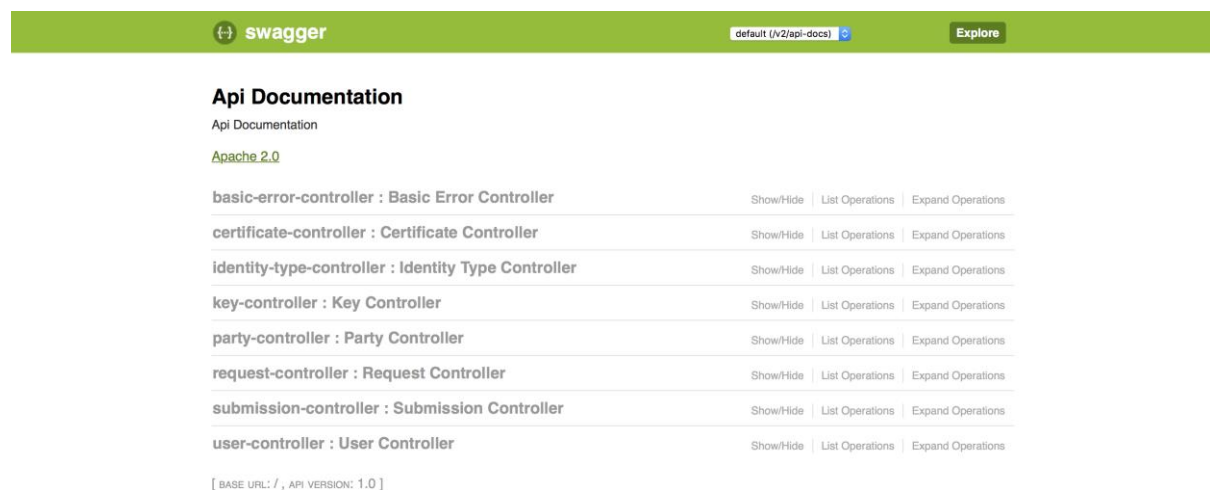
Overview

The backend application is the core of MiD. It is the hub through which all other branches of MiD communicate through. Usage of this part of the application is through endpoint calls. A user can call any of the endpoints, authentication permitting, and receive back data from the application.

Note: For detailed instructions on how to implement the authentication headers into an application that will talk to MiD please view the [API documentation](#) included with this documentation set.

Swagger Interface

The allow for easier documentation of the application Swagger has been implemented into the backend. This is a library that allows for the exposing and documentation of every endpoint available to the user. By visiting https://mid-secure.ie/mid/swagger_ui.html (replacing the hostname with where you placed your implementation of the backend application)



A user only needs to click on any of the controllers listed to see the available calls. Each call has documented inputs/outputs that can be tested.

submission-controller : Submission Controller		Show/Hide	List Operations	Expand Operations
user-controller : User Controller		Show/Hide	List Operations	Expand Operations
POST	/user	createUser		
DELETE	/user/{id}	deleteUser		
GET	/user/{id}	getUser		
PUT	/user/{id}	updateUser		
PUT	/user/{id}/token	updateUserToken		

[BASE URL: /mid , API VERSION: 1.0]

Each call is colour coded and marked to show the different HTTP types (POST, PUT, GET, DELETE) along with the title of the call within MiD. This gives user's the general idea of what each call will do. For example, The GET to "/user/{id}" is called getUser, so we can assume that it will get a user based on the id that is passed in.

GET
/user/{id}
getUser

Response Class (Status 200)
OK

Model | Example Value

```

{
  "fcmToken": "string",
  "id": "string",
  "keyId": "string",
  "nickname": "string",
  "publicKey": "string",
  "status": "string"
}

```

Response Content Type */*

Parameters

Parameter	Value	Description	Parameter Type	Data Type
id	(required)	id	path	string

Response Messages

HTTP Status Code	Reason	Response Model	Headers
401	Unauthorized		
403	Forbidden		
404	Not Found		

Try it out!

Clicking on any of the calls gives you the above overview of what you will enter as a parameter and what you will get back. It also allows you to test out calls to the server. Requests with legal data can be viewed through this interface to gain a better understanding of how it all works.

Backend Authentication

A large portion of the endpoints makes use of a form of authentication. If a user wishes to test all the endpoints, then they must conform to the authentication policy. Authentication is in the form of a basic auth header containing the user id and that user's token. The token must be encrypted with their private key. The server will validate the token by unencrypting it with your public key. It will ensure you are allowed access to the content being requested.

If a user has ready access to the encrypted token, then they can use tools such as [Postman](#) and paste the token into the header fields.

Admin Console

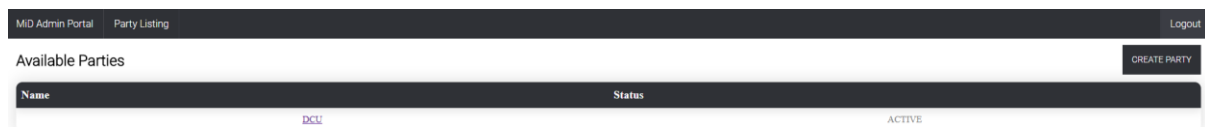
Overview

The admin console is a testing platform to demonstrate the functions that a party can carry out. It's important to note that the admin interface isn't at all appropriate in a production environment but provides a good base for an identifying party to build off. This section breaks down the common workflows of an identifying party and how you can go about carrying them out.

Note that you must implement the dummy web backend for this to function properly. MiD makes use of token-based authentication and the dummy backend wraps all requests made by the UI with the correct authentication token headers. You may implement your own but if you'd like a working example of it you can find it in the source code.

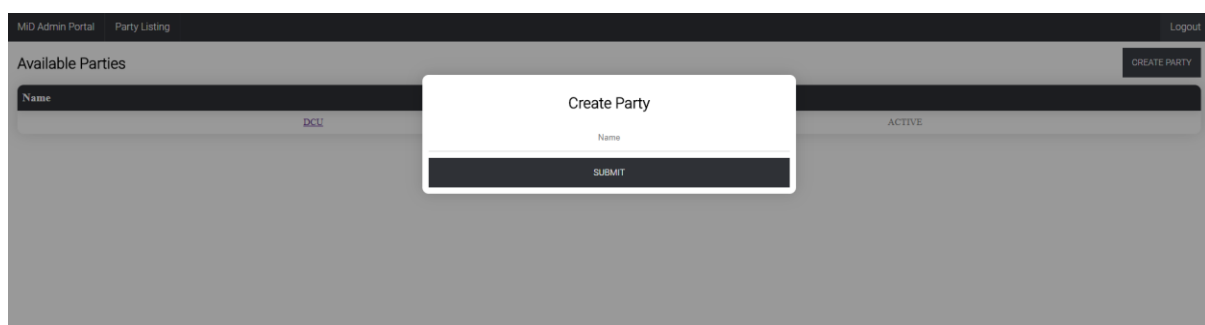
Party Creation

The initial page you're brought to within the UI (past the dummy login page) is a party listing page. Here you can see all the currently created parties within the system.



These created parties are what's used to create identity types which users will create submissions for. An example of this is a party called DCU creating a student card. Users can now create local copies of this identity type and request verification of their information from DCU. Submissions are handled within later pages of this UI. If a user wishes to create a party, they click on "Create Party" and enter a username. This username will be the name that users of the application will see in their submissions and with anything tied to identity types belonging to that party.

When a user clicks "Submit" that party is created within MiD and returned to the UI. The dummy backend will take care of the storage of this created party along with the tokens and keys linked to it. If one were to create their own implementation of this UI then they must take care of this key generation and token storage.



View Party

With the party successfully created you can view the parties page. This page contains the basic information that is linked to the party. The first section lists out all current pending submissions that are made to that party. The user can refresh this list or click on one of the currently displayed submissions to get more information.

The second section of this page is where the party can create and edit identity types. The user can edit current identity types or create new ones through this section.

MID Admin Portal
Party Listing
DCU
Logout

Name: DCU

Pending Submission Listings
REFRESH LIST

Submission Id	User Id	Status	Date Submitted
c00678ea-1162-4cfe-adc0-431aa3b11222		ACTIVE	

Available Identity Types
CREATE IDENTITY TYPE


Name	Version Number	Status
DCU Student Card	1	ACTIVE

View Submission

Submissions are requests made to the owner of an identity type to validate the information the user has entered, verifying that it matches what the party have on file. A submission is made of these entries along with a current picture. The party will need to verify the validity of this information. Once reviewed, the submission they can click “Accept” or “Reject”. This will mark their identity as valid and a proof of the party’s verification is created and returned to the creator of the submission.

MID Admin Portal
Party Listing
DCU
Submission
Logout

Submission ID: 4fcb100d-0bb8-4734-a0b4-8ff1b58f8b42



Student Number
12345678

Firstname
Test

Surname
User

ACCEPT
REJECT

Create Identity Type

If a party wishes to accept submissions of their type of identity they must first create it within MiD. This is done on the party's page within the "Available Identity Types section". Click on "Create Identity Type" and enter in all the required information.

The screenshot shows the 'Create Identity Type' modal form overlaid on the 'DCU' party page. The modal has a title 'Create Identity Type' and a section 'General Information'. It contains the following fields: 'Identity Type Name', 'Cover Image URL', and 'Icon Image URL'. Below these is the 'Entry Fields' section, which includes a table with columns for 'Entry Field Name' and 'Firstname'. There is an 'ADD FIELD' button next to the table. At the bottom of the modal are 'SUBMIT' and 'CANCEL' buttons. The background shows the 'DCU' party page with sections for 'Pending Submission Listings' and 'Available Identity Types'.

An identity type contains a title, a cover and icon image (for the mobile applications UI) and the fields that make up the identity type. These fields can be one of several types, firstname, surname, birthday, key field (eg. Passport number) and address. Fields can be added to the application at the request of an identifying party, but these basic fields allow for the creation of simple identity types. Once an identity type is created it will appear in this section and can be updated by clicking on it

Update Identity Type

An identity type, once created, can be updated at any time. Any information entered initially can be updated with other information. It's very important to note that updates to the title or fields will invalidate all successful submissions to an identity type so only update if absolutely necessary.

The screenshot shows the 'Update Identity Type' form for the 'DCU Student Card' identity type. The form has a title 'Identity Type Name: DCU Student Card' and a section 'General Information'. It contains the following fields: 'Identity Type Name' (DCU Student Card), 'Cover Image URL' (https://www.dcu.ie/sites/default/files/Homepage_More_dcuincorporation.jpg), and 'Icon Image URL' (https://www.dcu.ie/sites/default/files/Homepage_More_dcuincorporation.jpg). Below these is the 'Entry Fields' section, which includes a table with columns for 'Entry Field Name' and 'Id Field'. There is an 'ADD FIELD' button next to the table. At the bottom of the form are 'SUBMIT' and 'DELETE' buttons.