MAY 20, 2018

USER MANUAL

MID – MY IDENTITY

CILLIAN MC NEILL - 14352621

SCHOOL OF COMPUTING

SUPERVISOR: GEOFF HAMILTON

Table of Contents

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 it is being deploying to. It is 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 of this) 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. Due to this character, as well as the overall design of blockchains, they 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 that will need to be taken to install the application. No step should be skipped to ensure proper performance. I 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:

- 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. 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 building the application, it must ensured 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 the 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&charact
erEncoding=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 the location it's deployed to changes, update mid.hyperledger within the properties file to the new URL.

Google Firebase Configuration

Google Firebase is used to manage notifications to and from users and identifying parties. For simple integration of an instance of Google Firebase, 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, build MiD's network definitions into a Business Network Archive (BNA) file and deploy it to the docker containers created within the initial setup of 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 the terminal is going to be shut 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 the 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 method of running and should be adhered to if attempting to run them. It is recommended to run all of these tests before attempting to use the backend, so that it can be ensured that there are no errors with the 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 building the application, but will also run the unit tests and return any errors that occur. Note that the project will not 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 of 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, 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+. A signed version of the application can be built but this is not necessary if it is not being deployed to the play store.
Alternatively, the MiD application can be downloaded from Google Play and the server endpoint can be edited so that it is pointing at a different installation of the backend, 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 they make 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 the correct database URL and add the 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 a 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 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 the cli is installed, 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 the internet connection.

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 it is then available to answer requests (assuming both the dummy backend and MiD backend are deployed). If it needs to be built 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 relative linking can be used.
The folder can be renamed to a custom name and will be used as the endpoint on Tomcat. It can now be dropped into Tomcat to start receiving requests.

Usage Instructions

Mobile Application

Overview

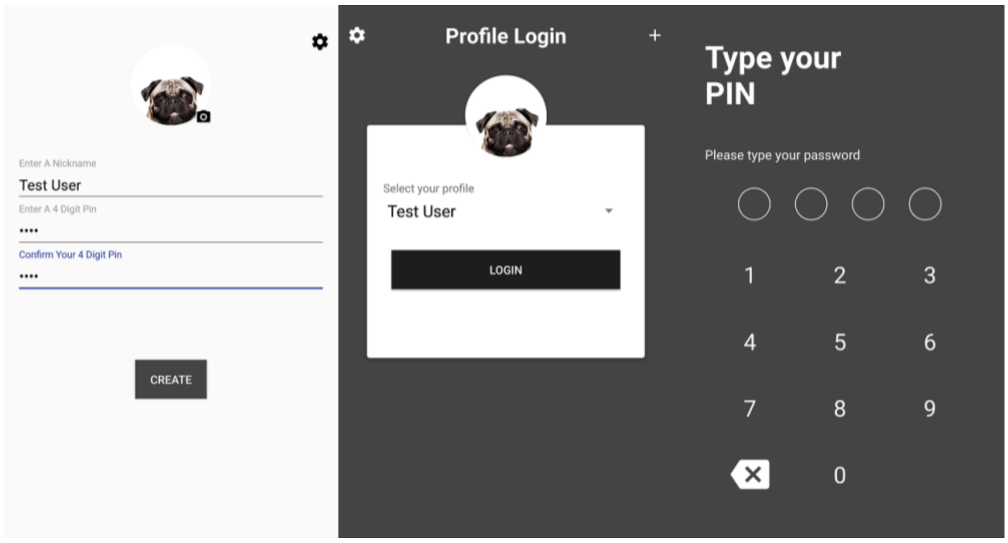
The mobile application is the user's link into MiD. It allows them to create and store identities to be verified by the corresponding parties. It also allows for the requesting of information from another user. All submissions/requests can be created and viewed through this application. In this section 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. It is possible for a third party to implement their own version of the UI as all this involves doing is providing 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 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 a pin they have created. Note that currently there is no support to change a user's pin, and a profile can be deleted but the pin cannot be changed.



Enter A Nickname
Test User
Enter A 4 Digit Pin
••••
Confirm Your 4 Digit Pin
••••

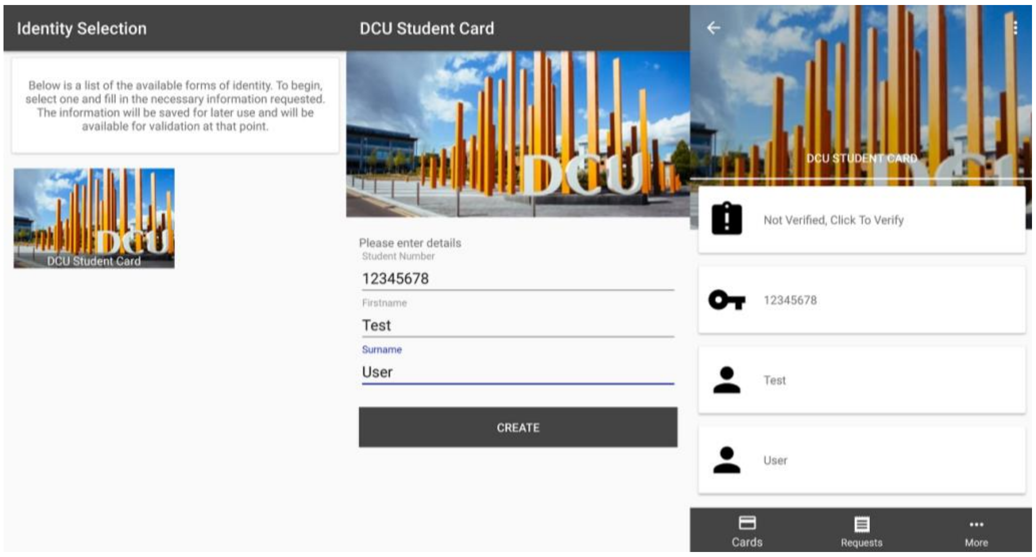
Select your profile
Test User

Please type your password

Identity Type Creation

Once a user has created a profile they need to create an initial identity type. The first screen is what the user is brought to when they 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 a basic form of the DCU student card. The user needs to enter their student number along with their first name and last name. Once they are 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, the status. You can see that as the identity has just been created, has not been verified. The identity must be verified before anything else can be done with this identity.



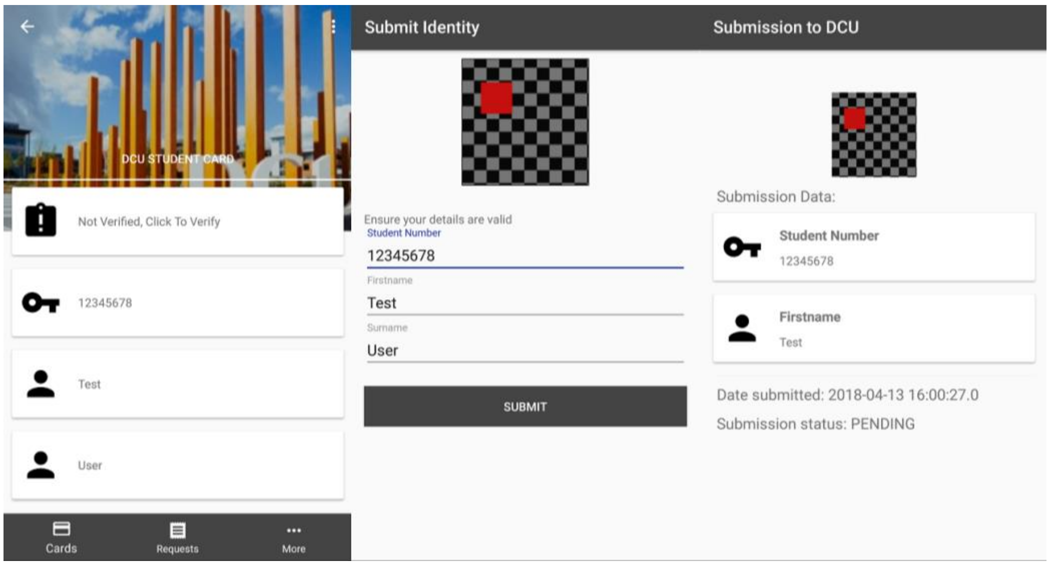
Below is a list of the available forms of identity. To begin, select one and fill in the necessary information requested. The information will be saved for later use and will be available for validation at that point.

Please enter details
Student Number
12345678
Firstname
Test
Surname
User

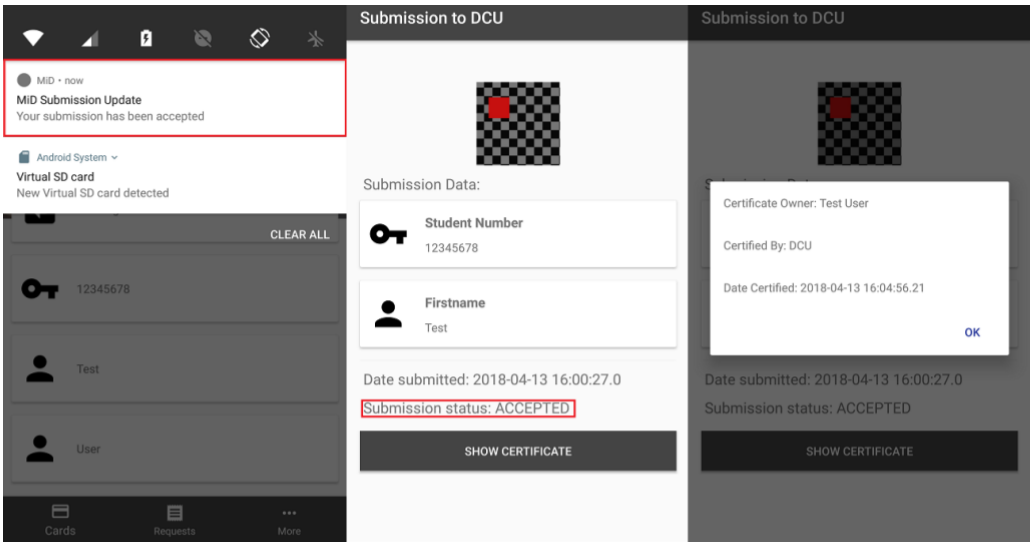
Not Verified, Click To Verify
12345678
Test
User

Identity Type Submission

If a user wishes to use this form of identity, it must be verified by the identifying party that created it, which in this case is DCU. The user must prove that they 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.



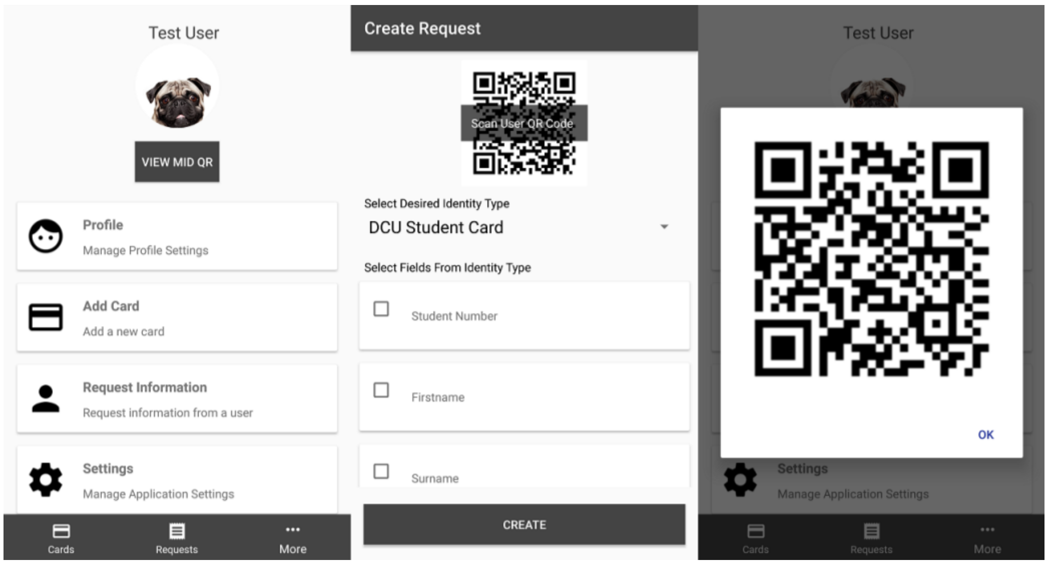
Above, we can see the initial submission creation process. The user double checks what they entered initially, takes a picture of themselves (above is a dummy image) and selects "Submit". The submission is sent to MiD where it can be viewed by the staff within DCU. Users are then shown the created submission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


MiD • now
MiD Submission Update
Your submission has been accepted
Android System
Virtual SD card
New Virtual SD card detected
CLEAR ALL
12345678
Test
User
Cards
Requests
More
Submission to DCU
Submission Data:
Student Number
12345678
Firstname
Test
Date submitted: 2018-04-13 16:00:27.0
Submission status: ACCEPTED
SHOW CERTIFICATE
Submission to DCU
Certificate Owner: Test User
Certified By: DCU
Date Certified: 2018-04-13 16:04:56.21
OK
Date submitted: 2018-04-13 16:00:27.0
Submission status: ACCEPTED
SHOW CERTIFICATE

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 can see the result of a processed application. The user receives a notification letting them know that their application has been accepted. When the user taps this, they are brought to the submission with the updated status and a link to the certificate that has been created. 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.

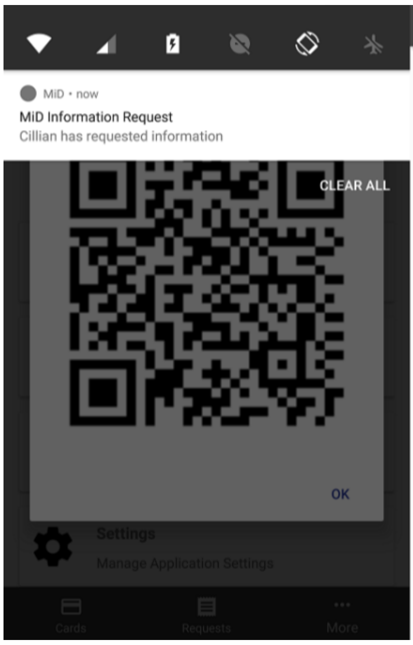
Information Request Creation

Users can send identity requests to one another. 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 select 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 user receiving the request will get a notification.

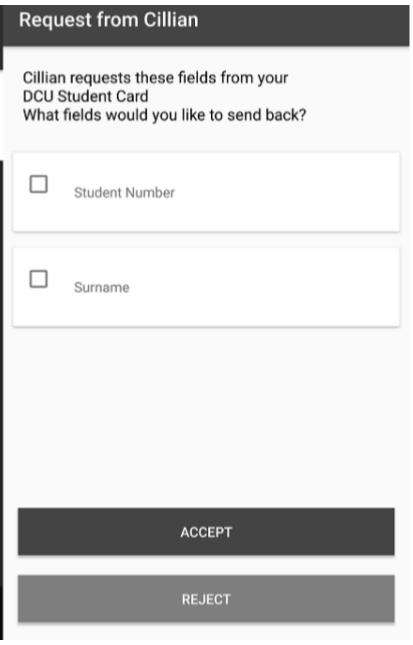


Information Review & Response

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 is important to note that a user does not have to send back all of the information that is being requested. This is to give the user finer control over what information is being given out to another user.



MiD • now
MiD Information Request
Cillian has requested information

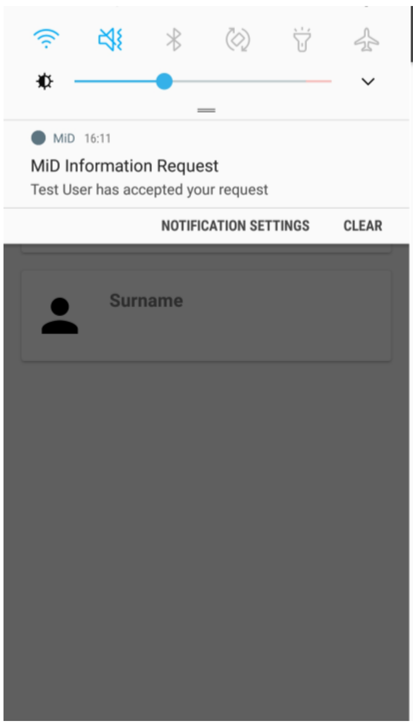


Request from Cillian

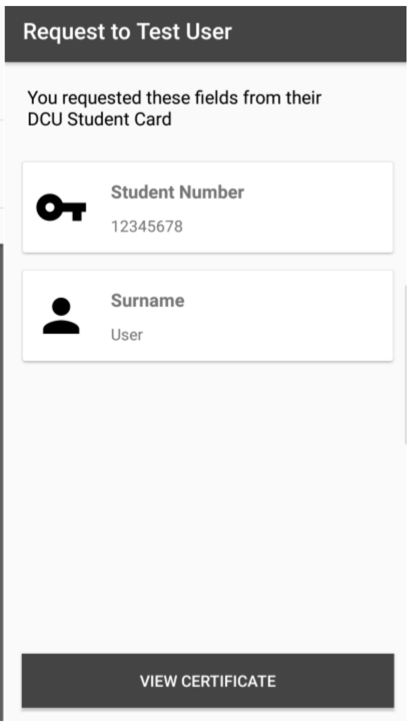
Cillian requests these fields from your DCU Student Card
What fields would you like to send back?

☐ Student Number

☐ Surname



MiD 16:11
MiD Information Request
Test User has accepted your request



Request to Test User

You requested these fields from their DCU Student Card

Student Number
12345678

Surname
User

Above is the response that the sender will get once the receiver has reviewed what has asked from them. It can be seen that the user has accepted the 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 was not present, then the request is not validated, and any information contained within could be incorrect or falsified.

Checks are made against the corresponding submissions to ensure falsification will not occur. This is done by matching hashed information stored in the certificate against the information being send back by the receiver. If the information does not match then thye have edited it and it cannot be trusted. If it does match then we can say that the information is the same as what was initially validated.

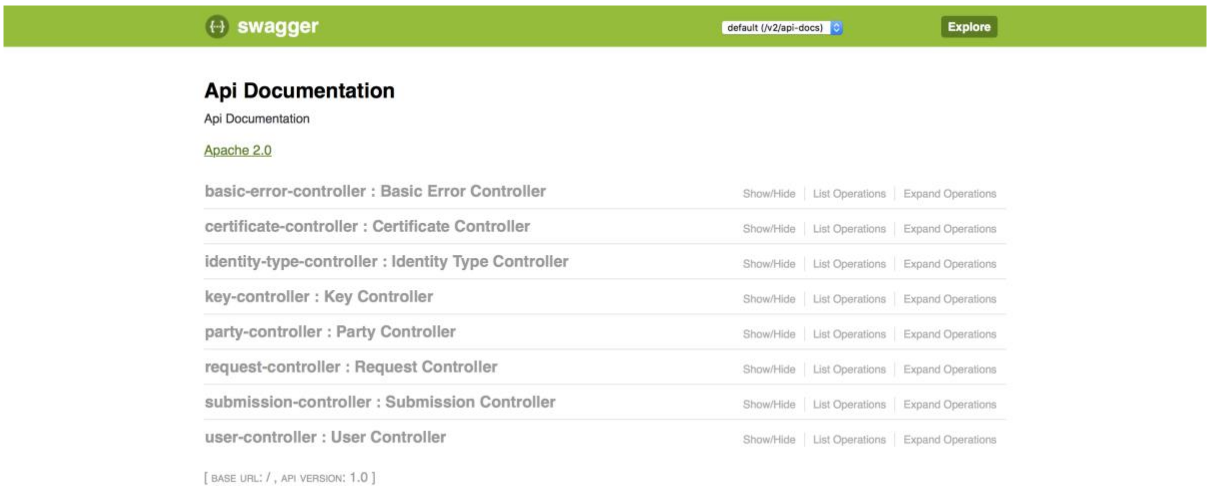
Backend Application

Overview

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

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 can view the most up to date version of the Swagger implementation.



Api Documentation

Api Documentation

Apache 2.0

basic-error-controller : Basic Error Controller — Show/Hide | List Operations | Expand Operations

certificate-controller : Certificate Controller — Show/Hide | List Operations | Expand Operations

identity-type-controller : Identity Type Controller — Show/Hide | List Operations | Expand Operations

key-controller : Key Controller — Show/Hide | List Operations | Expand Operations

party-controller : Party Controller — Show/Hide | List Operations | Expand Operations

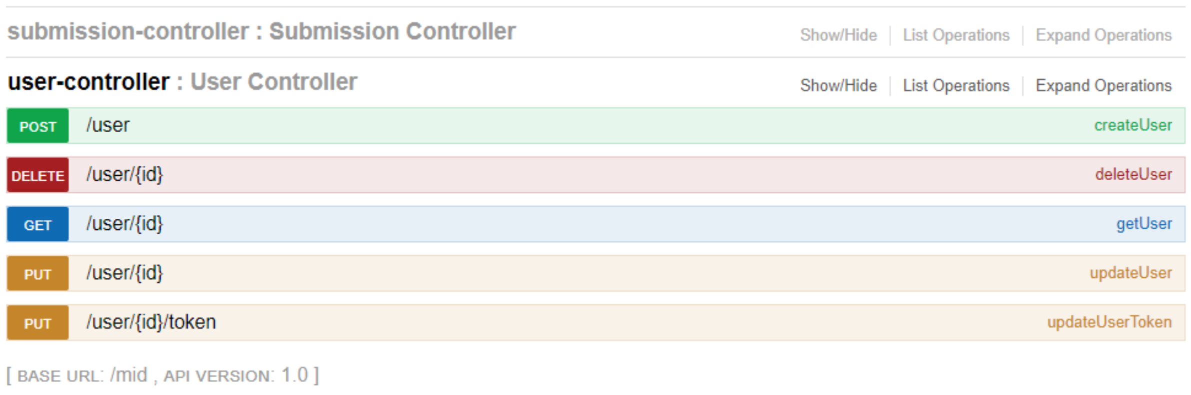
request-controller : Request Controller — Show/Hide | List Operations | Expand Operations

submission-controller : Submission Controller — Show/Hide | List Operations | Expand Operations

user-controller : User Controller — Show/Hide | List Operations | Expand Operations

[BASE URL: / , API VERSION: 1.0]

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



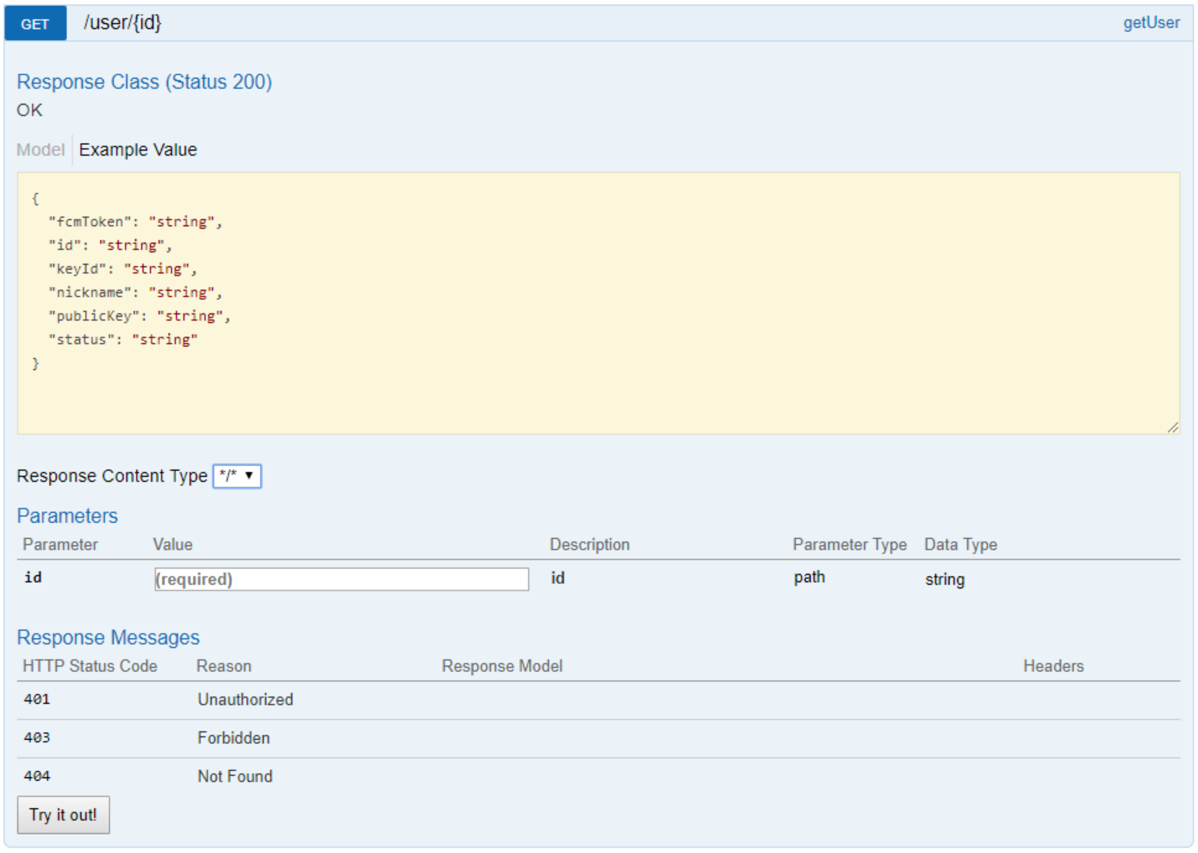
submission-controller : Submission Controller — Show/Hide | List Operations | Expand Operations

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 users a 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 us the above overview of what a user will enter as a parameter and what they will get back. It also allows anyone 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 functions.

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 the user's individual token. The token must be encrypted with their private key. The server will validate the token by unencrypting it with the public key. It will ensure the user 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.

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.

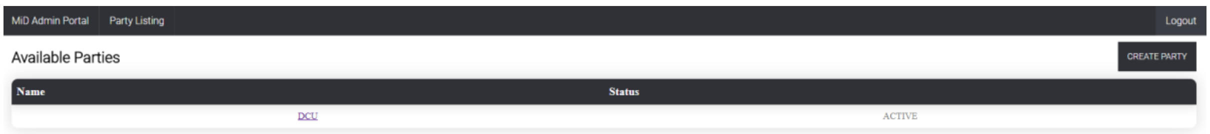
Admin Console

Overview

The admin console is a testing platform to demonstrate the functions that a party can carry out. It is important to note that the admin interface is not 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 the dummy web backend must be implemented 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. A party may implement their own but if testing is required on a working example of it, you can find it in the source code.

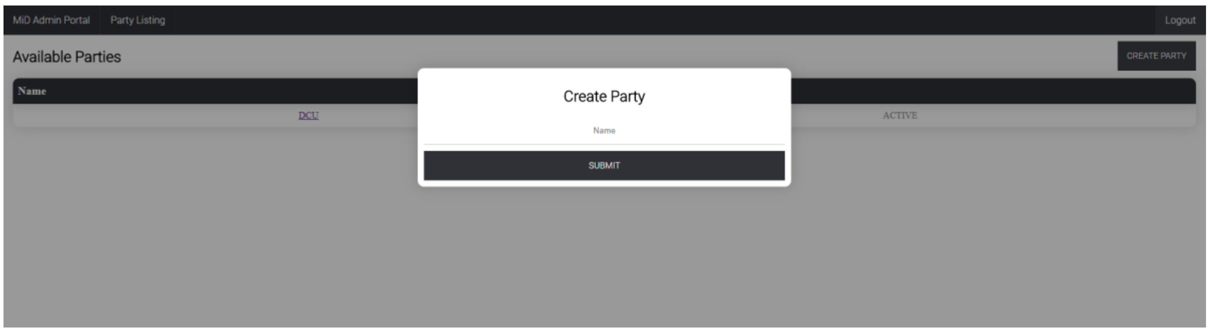
Party Creation

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



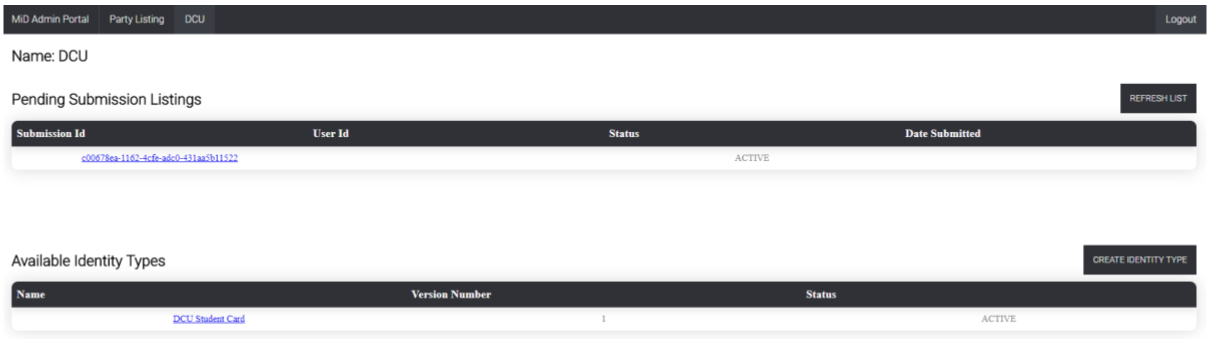
Name	Status
DCU	ACTIVE

These created parties are what is used to create identity types which users will create submissions for. An example of this is a party called “DCU” creating an identity type called 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 the 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 “Submit” is clicked, a 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, they must take care of this key generation and token storage.



View Party

With the party successfully created it can be viewed on the party's 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.



Name: DCU

Pending Submission Listings

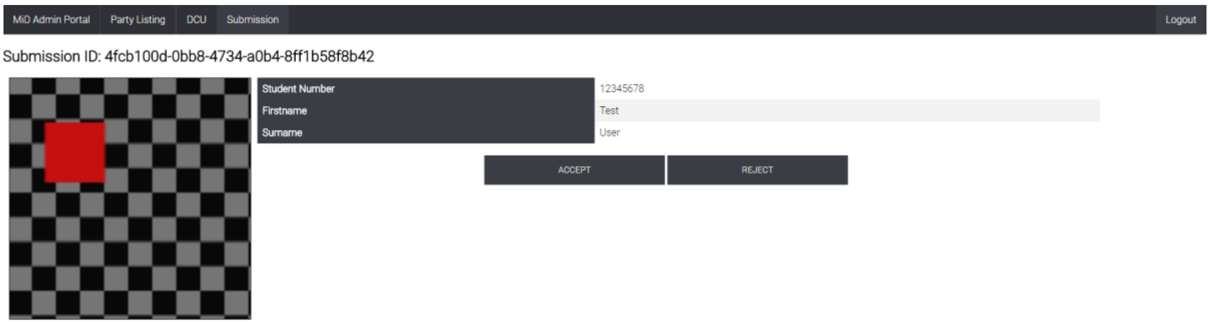
Submission Id	User Id	Status	Date Submitted
c00678ea-1162-4cfe-adc0-431aa5b11522		ACTIVE	

Available Identity Types

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 they have reviewed the submission they can click "Accept" or "Reject". Clicking "Accept" will mark the identity as valid and a proof of the party's verification is created and returned to the creator of the submission.

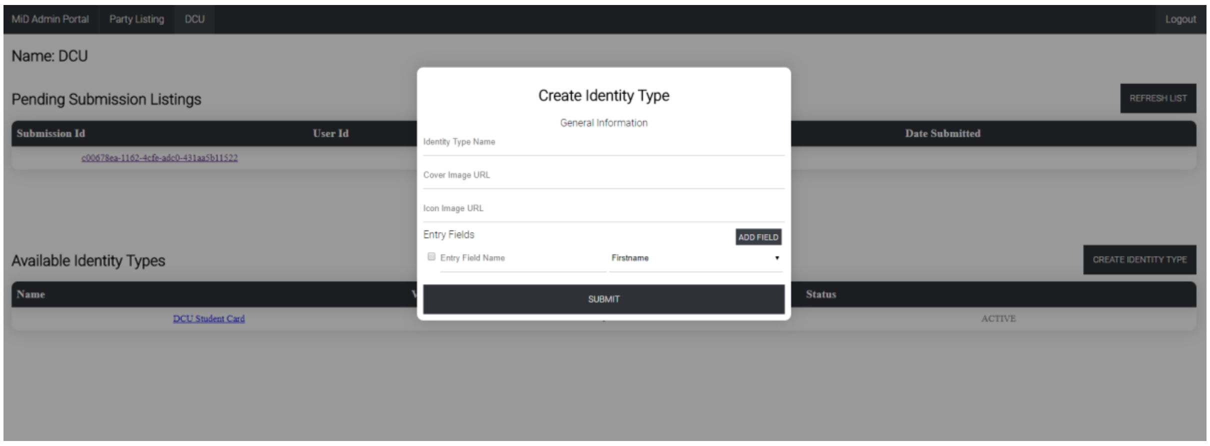


Submission ID: 4fcb100d-0bb8-4734-a0b4-8ff1b58f8b42

Student Number	12345678
Firstname	Test
Surname	User

Create Identity Type

If a party wishes to accept submissions of their type of identity, it must first be created 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.



Name: DCU

Pending Submission Listings

Submission Id	User Id	Date Submitted
c00678ea-1162-4cfe-adc0-431aa5b11522		

Available Identity Types

Name	Status
DCU Student Card	ACTIVE

Create Identity Type

General Information

Identity Type Name

Cover Image URL

Icon Image URL

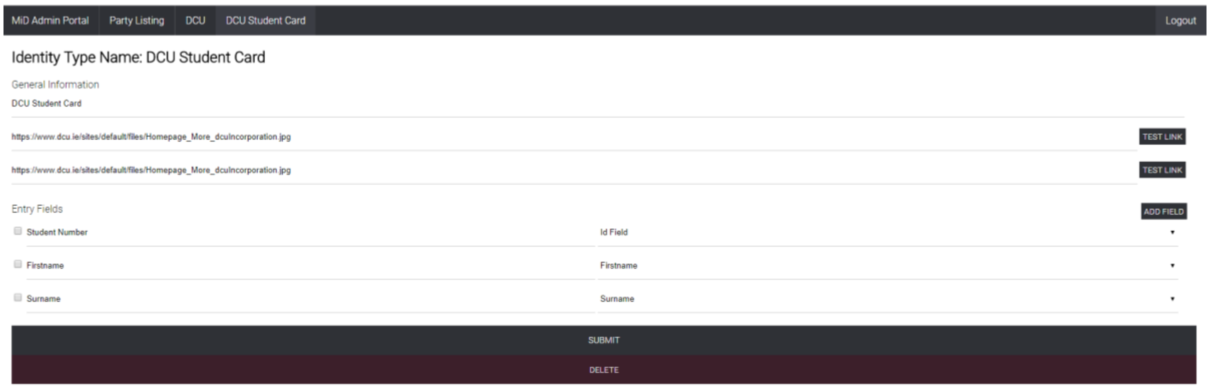
Entry Fields

Entry Field Name — Firstname

An identity type contains a title, a cover and icon image (for the mobile application's UI) and the fields that make up the identity type. These fields can be one of several types, first name, surname, birthday, key field (e.g. 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 any other information. It is very important to note that updates to the title or fields will invalidate all successful submissions to an identity type, so they should only be updated if it is absolutely necessary.



Identity Type Name: DCU Student Card

General Information

DCU Student Card

https://www.dcu.ie/sites/default/files/Homepage_More_dcuIncorporation.jpg

https://www.dcu.ie/sites/default/files/Homepage_More_dcuIncorporation.jpg

Entry Fields

Student Number	Id Field
Firstname	Firstname
Surname	Surname