



Mobile Connect SDK Integration Guide

Running the REST API sample integration

Technical Information Paper

Disclaimer

This document gives certain information about products and/or services provided by Gallagher Group Limited or its related companies (referred to as "Gallagher Group").

The information is indicative only and is subject to change without notice meaning it may be out of date at any given time. Although every commercially reasonable effort has been taken to ensure the quality and accuracy of the information, Gallagher Group makes no representation as to its accuracy or completeness and it should not be relied on as such. To the extent permitted by law, all express or implied, or other representations or warranties in relation to the information are expressly excluded.

Neither Gallagher Group nor any of its directors, employees or other representatives shall be responsible for any loss that you may incur, either directly or indirectly, arising from any use or decisions based on the information provided.

Except where stated otherwise, the information is subject to copyright owned by Gallagher Group and you may not sell it without permission. Gallagher Group is the owner of all trademarks reproduced in this information. All trademarks which are not the property of Gallagher Group, are acknowledged.

Copyright © Gallagher Group Ltd 2022. All rights reserved.

Contents

1	Background.....	3
2	External System Configuration	3
2.1	Create Test Items for sample integration	3
2.2	ASP.NET Core sample integration	4
2.3	Node.js sample integration.....	8

1 Background

This document will show you how to configure Command Centre and launch the NodeJS or ASP.NET Core sample integrations. These are designed to provide examples of things you might do on the server side to complement incorporating the Mobile Connect SDK into your mobile apps.

This document assumes you have the REST API Licensed, configured, and have your API key.

If you have not done this, please stop and follow the instructions in the **Command Centre Configuration** section of the document titled **Mobile Connect SDK – Getting started with the REST API** which accompanies this document.

For full documentation and reference regarding the Command Centre REST api, please refer to the attached REST api documentation in the CC_REST_<version> folder which accompanies the Mobile Connect SDK Integration package.

2 External System Configuration

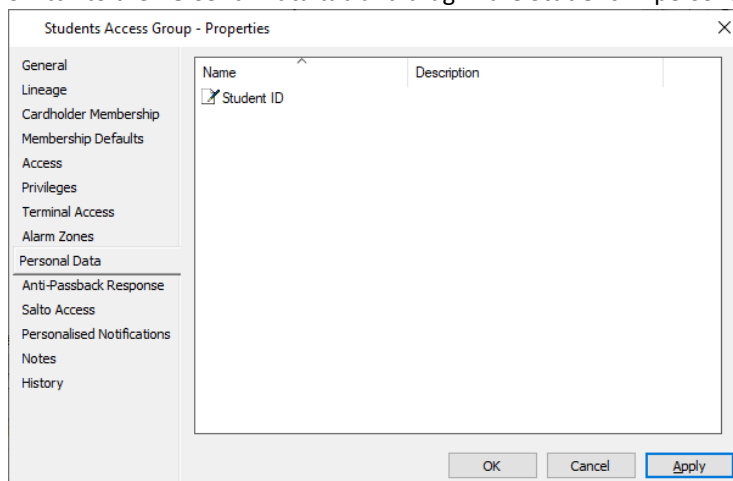
For the purposes of this guide, we will show how to load the API key into the sample integration websites.

The sample integration assumes that you have created some test items within your Command Centre server. These are documented in the following sections, and in the **README.md** file which accompanies the sample integration source code.

Note: These test items should not be needed for your own real integration. You will have your own personal data fields, access groups and other things which you should use in place of the test items

2.1 Create Test Items for sample integration

1. In the Command Centre Configuration Client, select the **Configure** menu, then **Personal Data Fields**.
 - a. Create a new Personal Data Field.
 - b. Enter **Student ID** in the name field. You can leave the **Type** and other settings at their default values
2. In the Command Centre Configuration Client, select the **Manage** menu, then **Access Groups**
 - a. Create a new Access Group
 - b. Enter **Students Access Group** in the name field.
 - c. Switch to the **Personal Data** tab and drag in the **Student ID** personal data field.



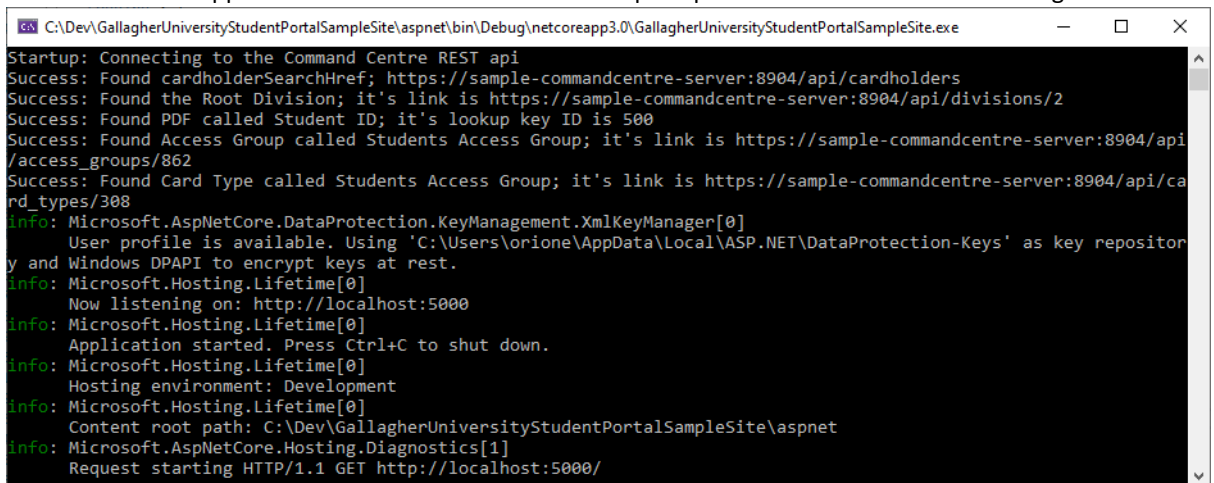
*Note: This will allow the SDK sample to function, however cardholders you create will not be granted access at any readers. You may wish to add some Access Zones under the **Access** tab as well.*

3 ASP.NET Core sample integration

1. Open the **SampleIntegration** folder that came packaged inside the zip file with this document, and then within that open the **aspnet** folder
2. Open the **GallagherUniversityStudentPortalSampleSite.sln** file using Visual Studio 2022.
Note: As this project is an ASP.NET Core 6 solution (targeting .NET 6 framework), you will need to have the latest updates applied to your Visual Studio 2022 installation. You will need at least version 17.3.3 of Visual Studio.
3. Open the **appsettings.json** file, and do the following
 - a. Replace the "host" value with the DNS name or IP address of your command centre server
 - b. Replace the "apiKey" value with the API key you created above.
 - c. As we have not required client certificates in command centre, you can leave the entries in place for "clientCertificatePfx" and "clientCertificatePfxPassword", they will not be active.

```
// Our configuration
"host": "https://sample-commandcentre-server:8904",
"apiKey": "02FB-970F-0898-6BCB-D562-4BA3-C471-BBF3",
```

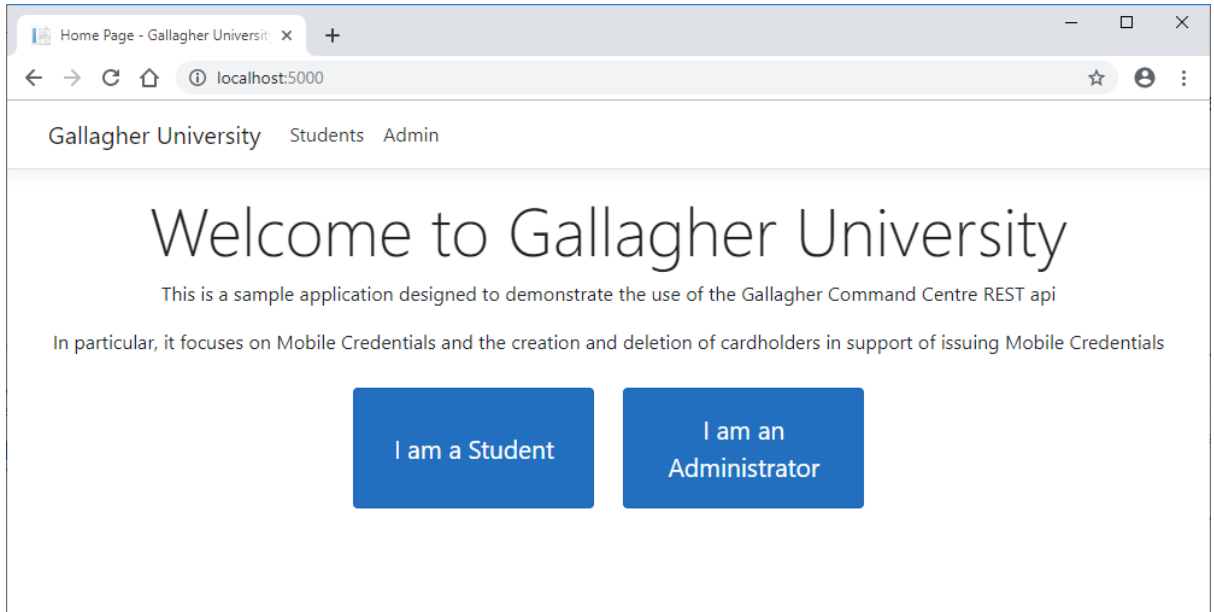
4. Build and run the application. You should see a command prompt window similar to the following:



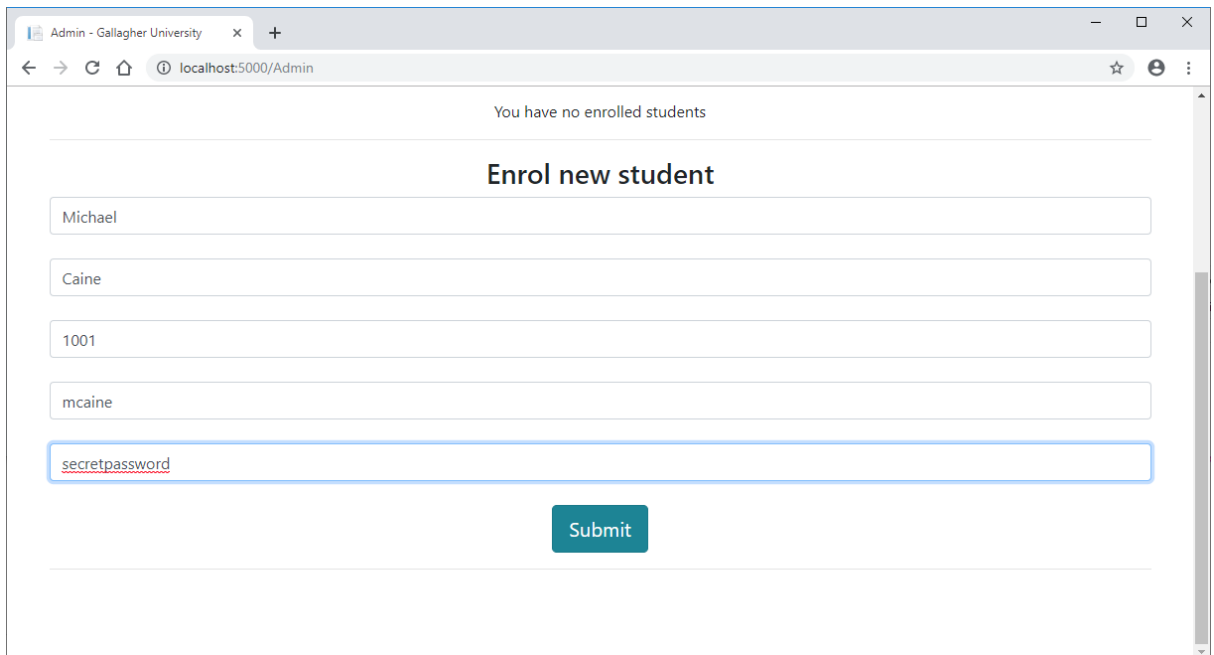
```
C:\Dev\GallagherUniversityStudentPortalSampleSite\aspnet\bin\Debug\netcoreapp3.0\GallagherUniversityStudentPortalSampleSite.exe
Startup: Connecting to the Command Centre REST api
Success: Found cardholderSearchHref; https://sample-commandcentre-server:8904/api/cardholders
Success: Found the Root Division; it's link is https://sample-commandcentre-server:8904/api/divisions/2
Success: Found PDF called Student ID; it's lookup key ID is 500
Success: Found Access Group called Students Access Group; it's link is https://sample-commandcentre-server:8904/api/access_groups/862
Success: Found Card Type called Students Access Group; it's link is https://sample-commandcentre-server:8904/api/card_types/308
info: Microsoft.AspNetCore.DataProtection.KeyManagement.XmlKeyManager[0]
      User profile is available. Using 'C:\Users\orione\AppData\Local\ASP.NET\DataProtection-Keys' as key repository and Windows DPAPI to encrypt keys at rest.
info: Microsoft.Hosting.Lifetime[0]
      Now listening on: http://localhost:5000
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: C:\Dev\GallagherUniversityStudentPortalSampleSite\aspnet
info: Microsoft.AspNetCore.Hosting.Diagnostics[1]
      Request starting HTTP/1.1 GET http://localhost:5000/
```

5. Visual studio should automatically launch your default browser and load <http://localhost:5000/>. If for some reason it does not auto-launch, please load that URL manually. You should see the home page

of the sample integration site:



6. Click **I am an administrator** to be taken to the administrator area. This area allows you to create and delete cardholders.
7. By default you should see **You have no enrolled students**
Fill out some values under **Enrol new student** and click **Submit**



8. After clicking submit, the page should reload and you should see your new student in the list. Verify it has been created correctly using the Command Centre Client.
Open a **Cardholder Viewer**, then search for the name you entered. You should see the cardholder

record with the correct name and personal data field values:

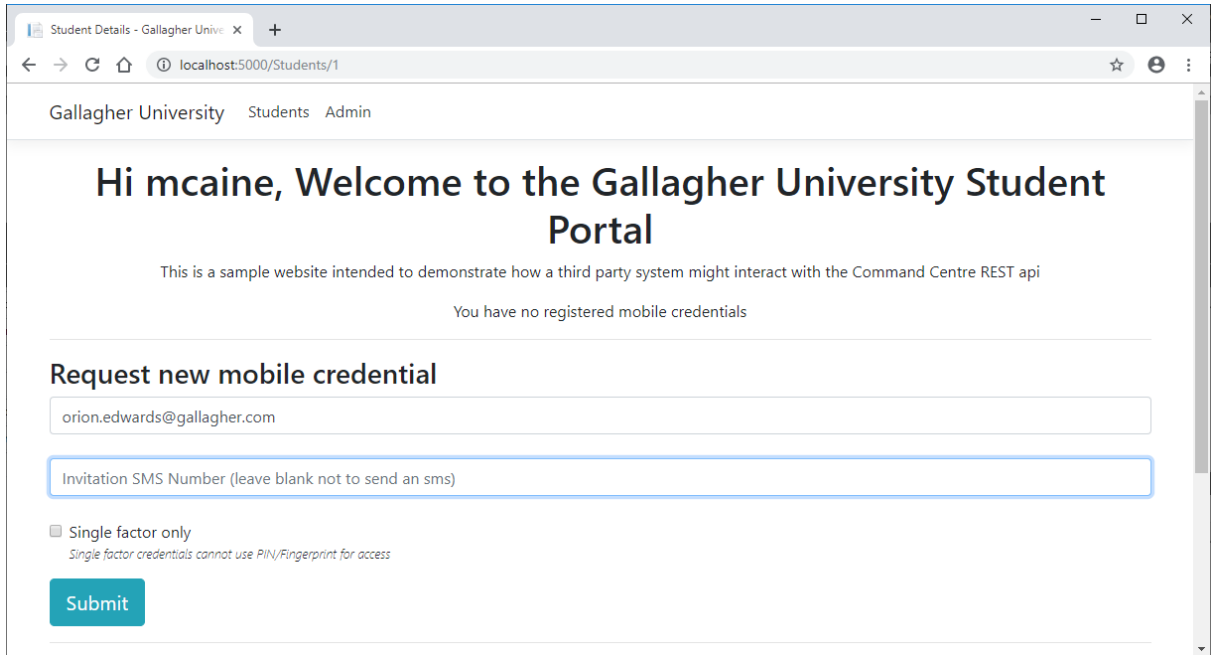
The screenshot shows a web application interface. At the top, there is a search bar with the text 'caine' and a dropdown menu set to 'Name'. Below the search bar is a table with columns: First Name, Last Name, Description, Division, and Card Numbe... The table contains one row with the values: Michael, Caine, Root Division, and Card Numbe... Below the table, there is a section for 'Cardholder Details' with fields for First Name (Michael), Last Name (Caine), Short Name, Description, Division (Root Division), Access Time (Standard), and Student ID (1001). To the right of the details is a section for 'Cardholder Access Groups' with a table with columns: Name, Description, From, Until, and Status. The table contains one row with the values: Students Access Group, Root Division, and Active. Below the access groups is a section for 'Cardholder Cards' with a checkbox for 'Cardholder Authorised' and a table with columns: Num..., Type, Issue, From, Until, and Status.

9. Go back to the integration site, switch to the **Students** area, and log in with the student ID personal data field value, and associated password that you entered earlier:

The screenshot shows a web browser window with the URL 'localhost:5000/Students'. The page title is 'Gallagher University Student Portal'. The page content includes the text: 'This is a sample website intended to demonstrate how a third party system might interact with the Command Centre REST api'. Below this text, there is a login form with two input fields: one for the Student ID (containing '1001') and one for the password (containing '*****'). A green 'Log in' button is positioned below the password field. At the bottom of the page, there is a note: 'Use the Administration page to add students, or try the pre-created student John, with Student ID '12345' and password 'abc''.

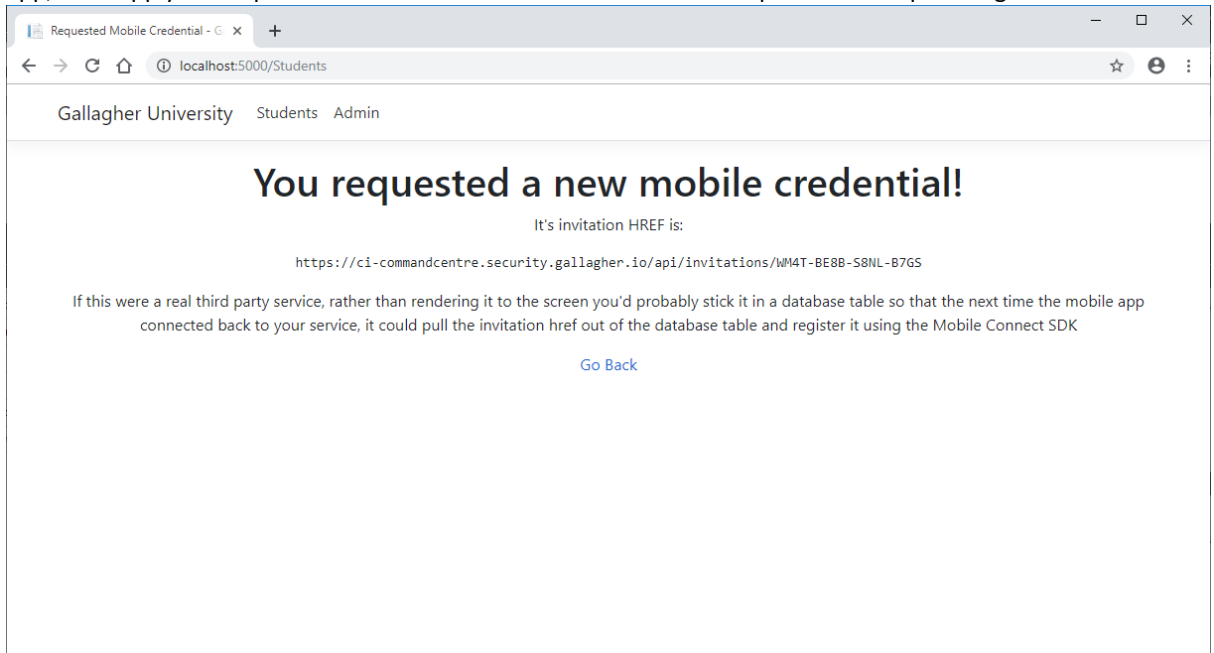
10. Fill out the **Request new mobile credential** form, then press **Submit**
We recommend for sample purposes that you enter an email address but leave the SMS number field

blank as it is optional.



The screenshot shows a web browser window with the URL `localhost:5000/Students/1`. The page header includes "Gallagher University" and navigation links for "Students" and "Admin". The main heading is "Hi mcaine, Welcome to the Gallagher University Student Portal". Below this, a message states: "This is a sample website intended to demonstrate how a third party system might interact with the Command Centre REST api" and "You have no registered mobile credentials". A section titled "Request new mobile credential" contains a form with two input fields: the first contains the email `orion.edwards@gallagher.com`, and the second is empty with the placeholder text "Invitation SMS Number (leave blank not to send an sms)". There is a checkbox for "Single factor only" with a note: "Single factor credentials cannot use PIN/Fingerprint for access". A blue "Submit" button is at the bottom of the form.

11. You should be taken to the success page, which will show the invitation HREF containing the unique invitation code. In your external system, this is the URL that you need to send down to your mobile app, and supply to the phone-side Mobile Connect SDK to enable the phone to complete registration.



The screenshot shows a web browser window with the URL `localhost:5000/Students`. The page header includes "Gallagher University" and navigation links for "Students" and "Admin". The main heading is "You requested a new mobile credential!". Below this, it says "It's invitation HREF is:" followed by the URL `https://ci-commandcentre.security.gallagher.io/api/invitations/WM4T-BE8B-58NL-B76S`. A note explains: "If this were a real third party service, rather than rendering it to the screen you'd probably stick it in a database table so that the next time the mobile app connected back to your service, it could pull the invitation href out of the database table and register it using the Mobile Connect SDK". A blue "Go Back" link is at the bottom.

Note: Creating mobile credentials requires that your Command Centre server has an active connection to the Command Centre Cloud

You can verify the credential by switching back to the Command Centre Client. Note the added Mobile Credential under the Cardholder Cards tile:

Cardholder Details		Cardholder Access Groups					
First Name:	Michael	Name	Description	From	Until	Status	
Last Name:	Caine	Students Access Group					
Short Name:						Active	
Description:		<input type="button" value="Assign Access"/> <input type="button" value="Copy Access"/> <input type="button" value="Remove"/>					
Division:	Root Division	Cardholder Cards					
Access Time:	Standard	<input checked="" type="checkbox"/> Cardholder Authorised					
Student ID:	1001	Num...	Type	Issue	From	Until	Status
		c94714...	Mobile Credential	1			Waiting for registration

4 Node.js sample integration

1. Make sure you have node.js version 16 or greater installed.
2. Open the **SampleIntegration** folder that came packaged inside the zip file with this document, and then within that open the **nodejs** folder
3. Launch a Command Prompt window within that folder.
4. Enter the command **npm install --save**

NPM should run for a while and should succeed with similar output to the below:

```

PS C:\Dev\GallagherUniversityStudentPortalSampleSite\nodejs> npm install --save
> sqlite3@4.0.6 install C:\Dev\GallagherUniversityStudentPortalSampleSite\nodejs\node_modules\sqlite3
> node-pre-gyp install --fallback-to-build
node-pre-gyp WARN Using request for node-pre-gyp https download
[sqlite3] Success: "C:\Dev\GallagherUniversityStudentPortalSampleSite\nodejs\node_modules\sqlite3\lib\binding\node-v64-win32-x64\node_sqlite3.node" is installed via remote
npm WARN gallagheruniversitystudentportalsamplesite@1.0.0 No repository field.

added 157 packages from 128 contributors and audited 376 packages in 7.574s
found 0 vulnerabilities

PS C:\Dev\GallagherUniversityStudentPortalSampleSite\nodejs>

```

5. Open the **config.json** file, and do the following
 - a. Replace the "host" value with the DNS name or IP address of your command centre server
 - b. Replace the "apiKey" value with the API key you created above.
 - c. As we have not required client certificates in command centre, you can leave the entries in place for "clientCertificatePfx" and "clientCertificatePfxPassword", they will not be active.

```

{
  "host": "https://sample-commandcentre-server:8904",
  "apiKey": "02FB-970F-0898-6BCB-D562-4BA3-C471-BBF3",

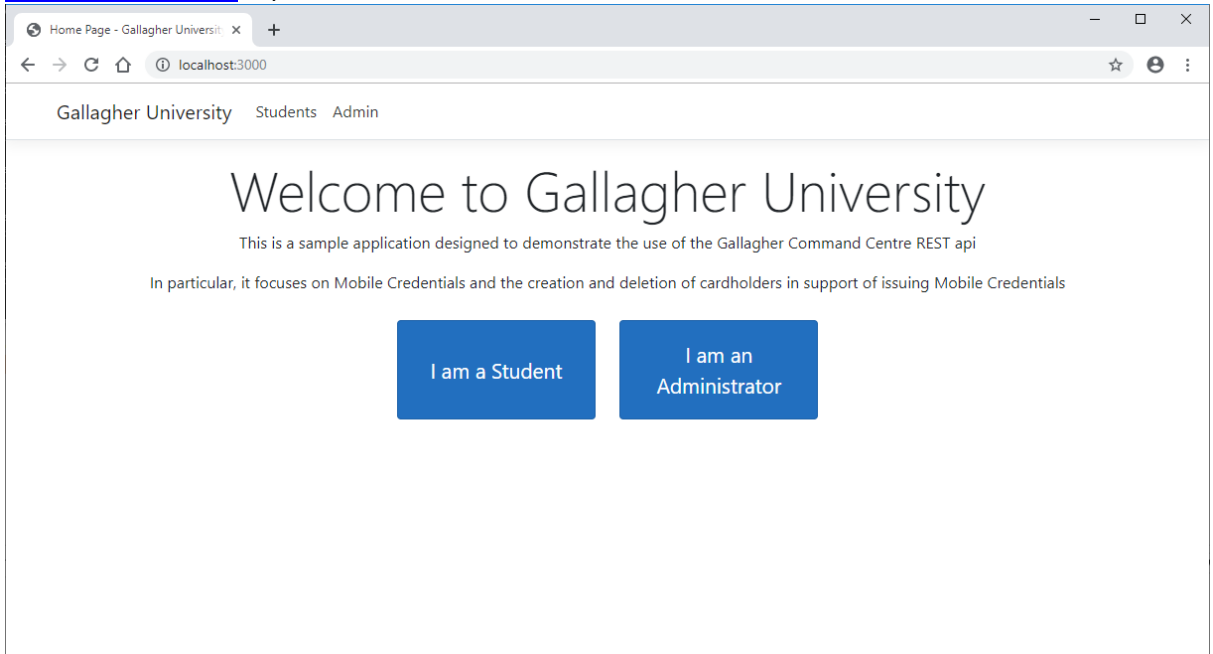
```


6. Run **node server.js**

You should see output like the following

```
PS C:\Dev\GallagherUniversityStudentPortalSampleSite\nodejs> node server.js
creating 'data' directory
GET https://sample-commandcentre-server:8904/api
undefined
Example app listening on port 3000!
Created database and students
Success: Found cardholderSearchHref; https://sample-commandcentre-server:8904/api/cardholders
GET https://sample-commandcentre-server:8904/api/items?type=15&name=%22Root%20Division%22
undefined
Success: Found the Root Division; it's link is https://sample-commandcentre-server:8904/api/divisions/2
GET https://sample-commandcentre-server:8904/api/items?type=33&name=%22Student%20ID%22
undefined
Success: Found PDF called Student ID; it's lookup key ID is 922
GET https://sample-commandcentre-server:8904/api/items?type=2&name=%22Students%20Access%20Group%22
undefined
Success: Found Access Group called Students Access Group; it's link is https://sample-commandcentre-server:8904/api/access_groups/923
GET https://sample-commandcentre-server:8904/api/card_types?name=%22Mobile%20Credential%22
undefined
Success: Found card type called Mobile Credential; it's href is https://sample-commandcentre-server:8904/api/card_types/308
```

7. Node will not auto-launch a web browser like Visual Studio does, so manually open <http://localhost:3000> in your default web browser.



8. The NodeJS sample site is functionally and visually identical to the ASP.NET site. Click **I am an administrator** to be taken to the administrator area, and then proceed with the same instructions as above from the ASP.NET sample integration

Note: If for some reason you chose to run both the NodeJS and ASP.NET integrations on the same system, they will connect to the Command Centre server, however they each maintain their own local database of usernames and passwords. If you add a cardholder in the NodeJS integration, it will not be visible in the ASP.NET integration and vice versa.