Mobile Connect SDK Integration Guide

Getting Started with the Command Centre REST API

Technical Information Paper

Contents

1 Background

This document is designed to provide a quickstart for working with the Command Centre REST api. It does not cover detailed scenarios.

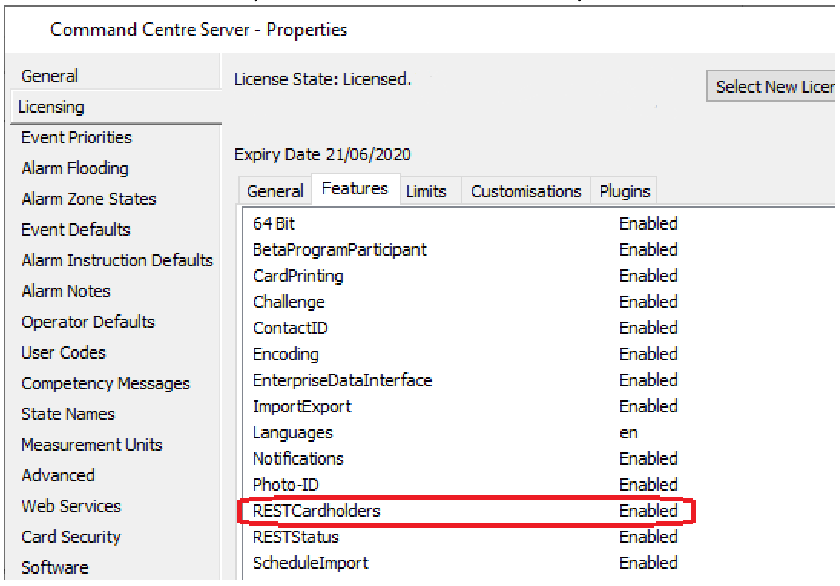
For full documentation and reference regarding the Command Centre REST api, please refer to the attached REST api documentation in the CC_REST_<version> folder which accompanies the Mobile Connect SDK Integration package.

2 Command Centre Configuration

2.1 Verify you are licensed to use the REST api

Your license file will need the RESTCardholders feature. You can check this as follows:

1. Launch the Command Centre Configuration Client and log in as a user with **Advanced User** privileges or the **System** operator
2. From the **File** menu, select **Server Properties**
3. Select the **Licensing** tab and then the **Features** tab within it.
   You should see an entry for **RESTCardholders** that says **Enabled**



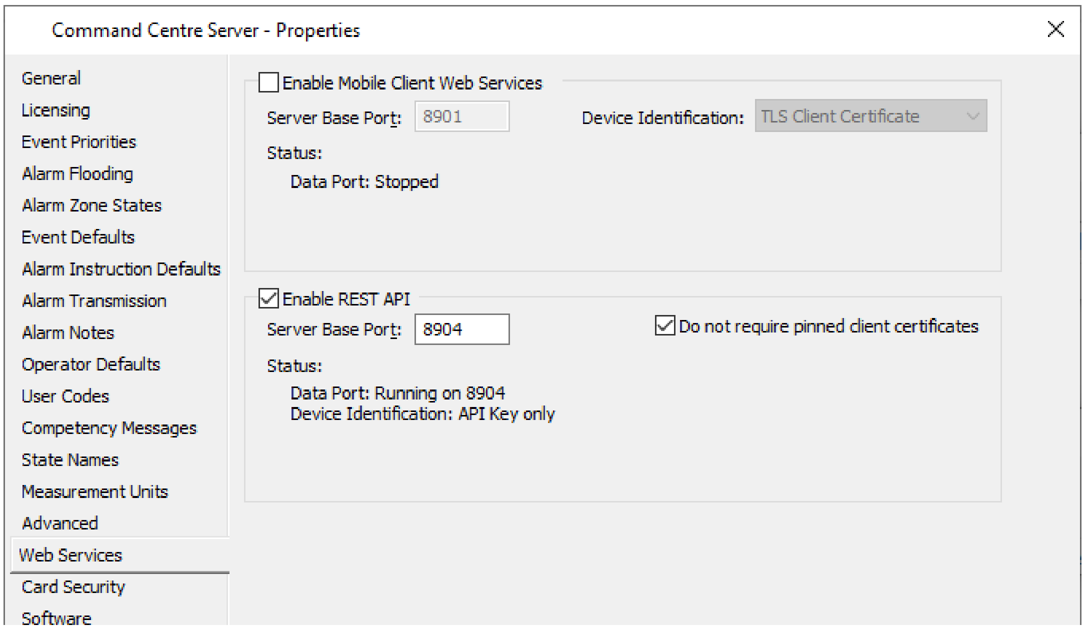
License State: Licensed.

Expiry Date 21/06/2020

64 Bit	Enabled
BetaProgramParticipant	Enabled
CardPrinting	Enabled
Challenge	Enabled
ContactID	Enabled
Encoding	Enabled
EnterpriseDataInterface	Enabled
ImportExport	Enabled
Languages	en
Notifications	Enabled
Photo-ID	Enabled
RESTCardholders	Enabled
RESTStatus	Enabled
ScheduleImport	Enabled

2.2 Enable the REST api in Command Centre

1. Still within the **Server Properties** dialog you opened above, switch to the **Web Services** tab and enable the REST API.
   If this is an isolated test server, or if you are sure you have a secure network, you may temporarily tick the **Do not require pinned client certificates** checkbox during development.


Command Centre Server - Properties

☐ Enable Mobile Client Web Services
Server Base Port: 8901
Device Identification: TLS Client Certificate
Status:
Data Port: Stopped

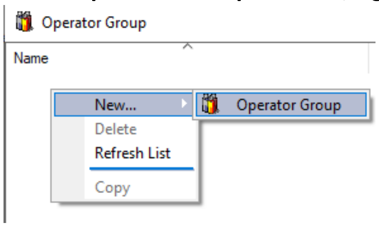
☑ Enable REST API
Server Base Port: 8904
☑ Do not require pinned client certificates
Status:
Data Port: Running on 8904
Device Identification: API Key only

NOTE: It is not recommended that you leave this checkbox ticked permanently in production environments as it reduces the security of your Command Centre Server

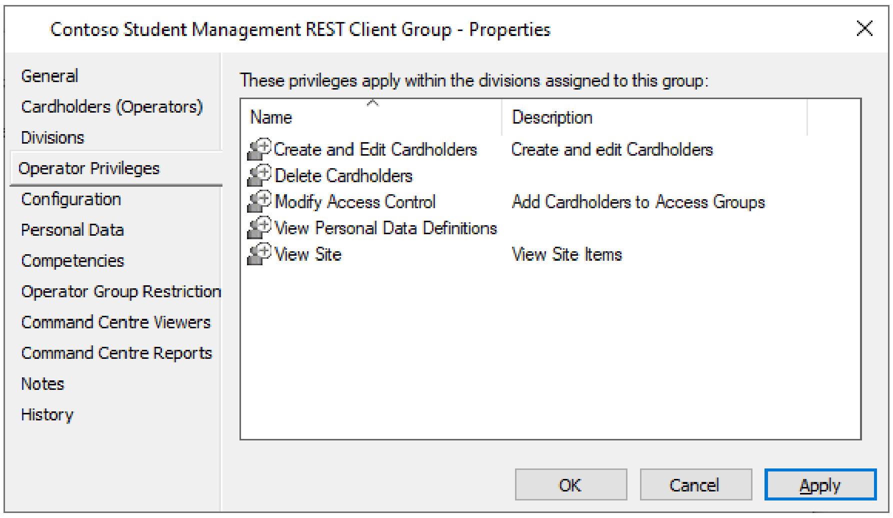
2.3 Create an operator group for your REST client

Your external system will need various privileges to read and write data using the REST api.
Privileges in Command Centre are managed using Operator Groups.

1. In the Command Centre Configuration Client, select the **Manage** menu, then **Operator Groups**
2. In the **Operator Group** window, right-click on the empty space and select **New... > Operator Group**



3. Give the operator group a descriptive name. We recommend adding "REST Client Group" somewhere in the description to help identify this later. For example, if your external system is the Contoso Student Management System, something like ***"Contoso Student Management REST Client Group"***
4. Select the **Operator Privileges** tab and add the privileges that you need. To run the Mobile Connect SDK sample integration website, you need:
   Create and Edit Cardholders
   Delete Cardholders
   View Personal Data Definitions
   View Site
   Modify Access Control


Contoso Student Management REST Client Group - Properties

These privileges apply within the divisions assigned to this group:

Name	Description
Create and Edit Cardholders	Create and edit Cardholders
Delete Cardholders	
Modify Access Control	Add Cardholders to Access Groups
View Personal Data Definitions	
View Site	View Site Items

I would recommend adding these privileges as above, and then tweaking them later for your desired security requirements. NOTE that by default the privileges are granted to your entire site. You may wish to use divisions for finer grained security.

2.4 Create a cardholder to use that operator group

1. In the Command Centre Configuration Client, select the **Manage** menu, then **Cardholders**
2. In the **Cardholder** window, right-click on the empty space and select **New… > Cardholder**
3. Give the cardholder a descriptive name to indicate the external system. We recommend using a similar name as the external system. Given the above example of the Contoso Student Management system, ***"Contoso Student Management System"*** would be an appropriate cardholder name.

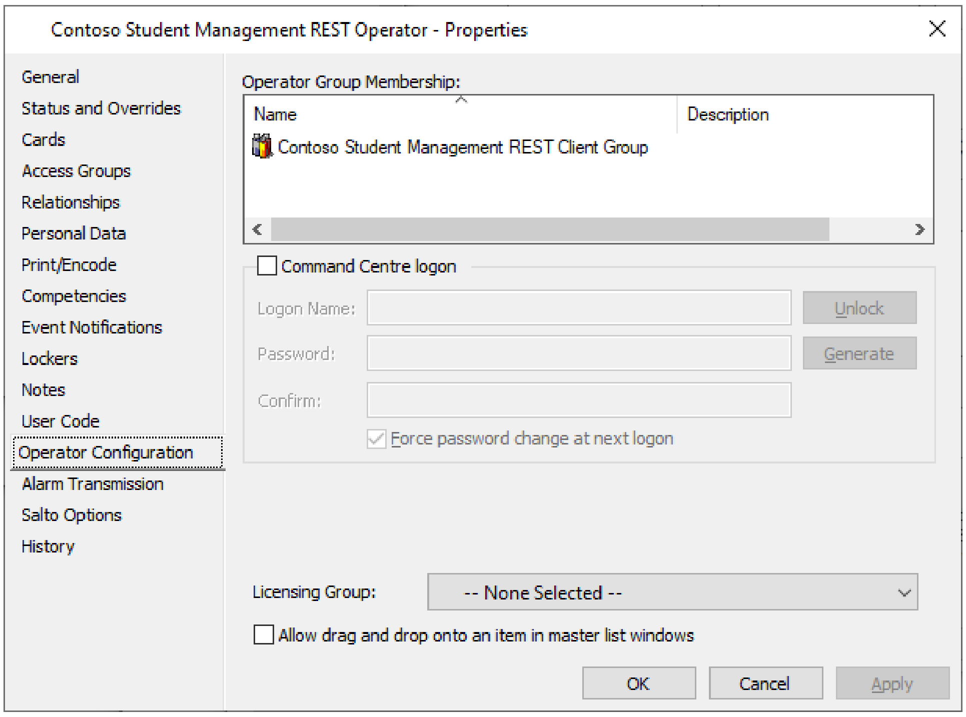
   NOTE: The names of the other items (REST Client, Operator Group, etc) are not as important, as they do not usually appear in audit events, however the name of this operator will appear against audit events for any actions they perform

   Example event for when a cardholder is created using the REST api:
   Operator "Contoso Student Management System" Added Cardholder "Brown, James"

   Please consider a name that will look acceptable in this context.

4. Select the Operator Configuration tab and drag in the operator group you created above.

5. **UN-TICK** the **Command Centre logon** checkbox, as this operator does not need to log on using any Command Centre GUI clients.


Contoso Student Management REST Operator - Properties

Operator Group Membership:

Name	Description
Contoso Student Management REST Client Group	

☐ Command Centre logon

Logon Name:
Password:
Confirm:
✓ Force password change at next logon

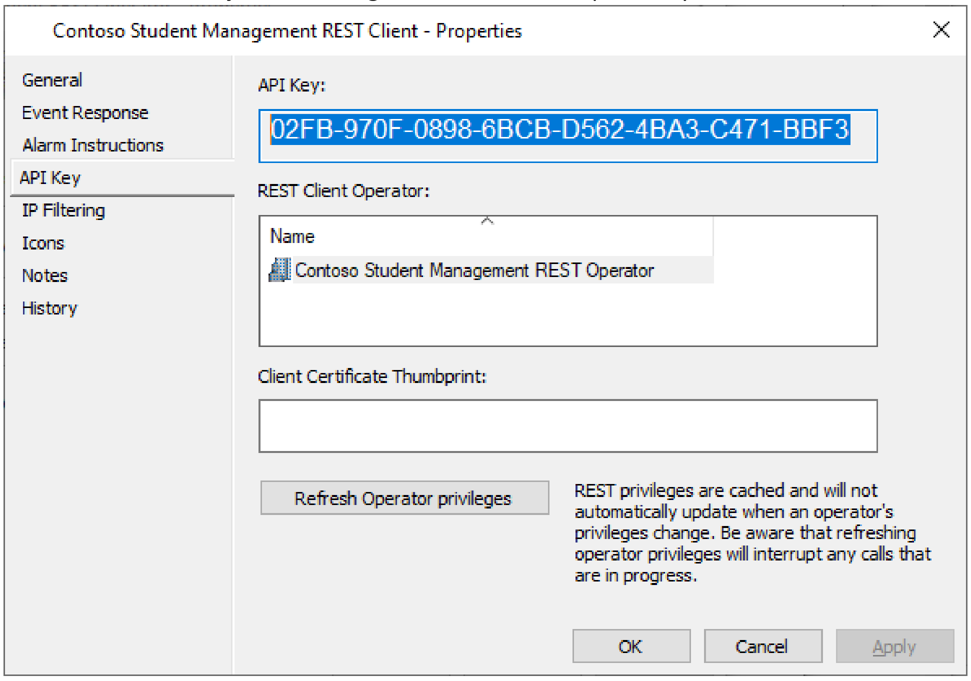
Licensing Group: -- None Selected --

☐ Allow drag and drop onto an item in master list windows

2.5 Create a REST Client item to control the external connection

1. In the Command Centre Configuration Client, select the **Configure** menu, then **Services & Workstations** (*at the bottom of the menu)*
2. In the **Services and Workstations** window, right-click on the empty space and select **New… > REST Client**
3. Give the item a descriptive name to indicate the external system. We recommend using a similar name as the operator group and adding the term "REST Client". Given the above example of the Contoso Student Management system, ***"Contoso Student Management REST Client"*** would be an appropriate cardholder name.

4. Switch to the **API Key** tab and drag in the cardholder operator you created above



Contoso Student Management REST Client - Properties

API Key: 02FB-970F-0898-6BCB-D562-4BA3-C471-BBF3

REST Client Operator:

Name
Contoso Student Management REST Operator

Client Certificate Thumbprint:

REST privileges are cached and will not automatically update when an operator's privileges change. Be aware that refreshing operator privileges will interrupt any calls that are in progress.

5. After saving the REST client item with the **Apply** button, please make note of the API key. You will need to copy this and load it into your external system.
   Please store it securely. Loss of an API key could cause other people to extract data and make changes to your Command Centre System.

You may optionally wish to secure the system by adding a Client Certificate Thumbprint and/or enabling IP filtering, however for now leave these on the default (blank) settings.

3 Walkthrough with Postman

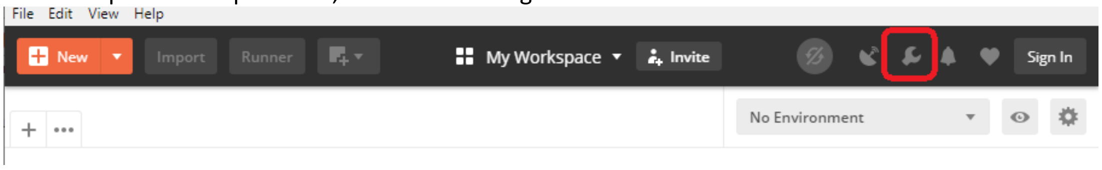
The remaining parts of this document demonstrate key parts of the Command Centre REST API using the Postman application from https://www.getpostman.com/

Note: This document references Postman version 7.10.0

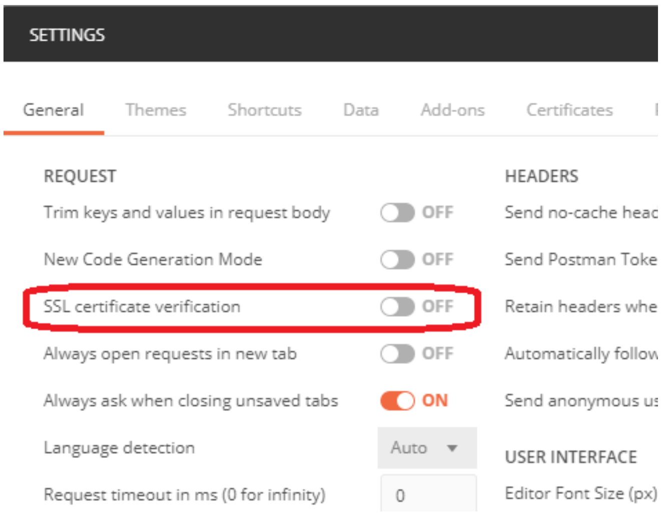
3.1 Turn off SSL certificate verification in the Postman Settings screen

Because Command Centre uses a self-signed certificate for its REST API, you need to turn off postman's certificate verification to connect to the REST API. You only need to do this once.

From the postman top toolbar, select the Settings Icon



In the settings screen, disable SSL certificate verification


SETTINGS

General | Themes | Shortcuts | Data | Add-ons | Certificates

REQUEST

Trim keys and values in request body	OFF
New Code Generation Mode	OFF
SSL certificate verification	OFF
Always open requests in new tab	OFF
Always ask when closing unsaved tabs	ON
Language detection	Auto
Request timeout in ms (0 for infinity)	0

HEADERS

Send no-cache heac

Send Postman Toke

Retain headers whe

Automatically follow

Send anonymous us

USER INTERFACE

Editor Font Size (px)

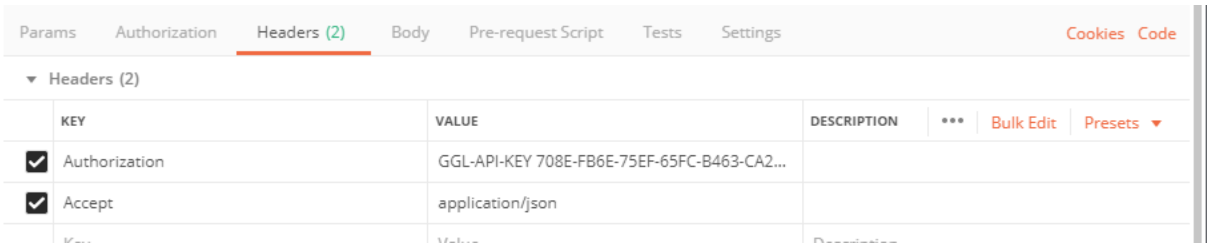
Save and close the settings screen

3.2 Create a new request and set up your API key and content types

Click the "+" button within postman main window to make a new request tab.

Switch to **Headers** and add the following

- Authorization: `GGL-API-KEY your-key`
- Accept: `application/json`


Params | Authorization | Headers (2) | Body | Pre-request Script | Tests | Settings | Cookies | Code

Headers (2)

	KEY	VALUE	DESCRIPTION
✓	Authorization	GGL-API-KEY 708E-FB6E-75EF-65FC-B463-CA2...	
✓	Accept	application/json	

All requests from this point forward will need a baseline of those two header values.

If you make a new tab in postman, it doesn't copy across headers from the previous tab, so you will need to re-enter them.

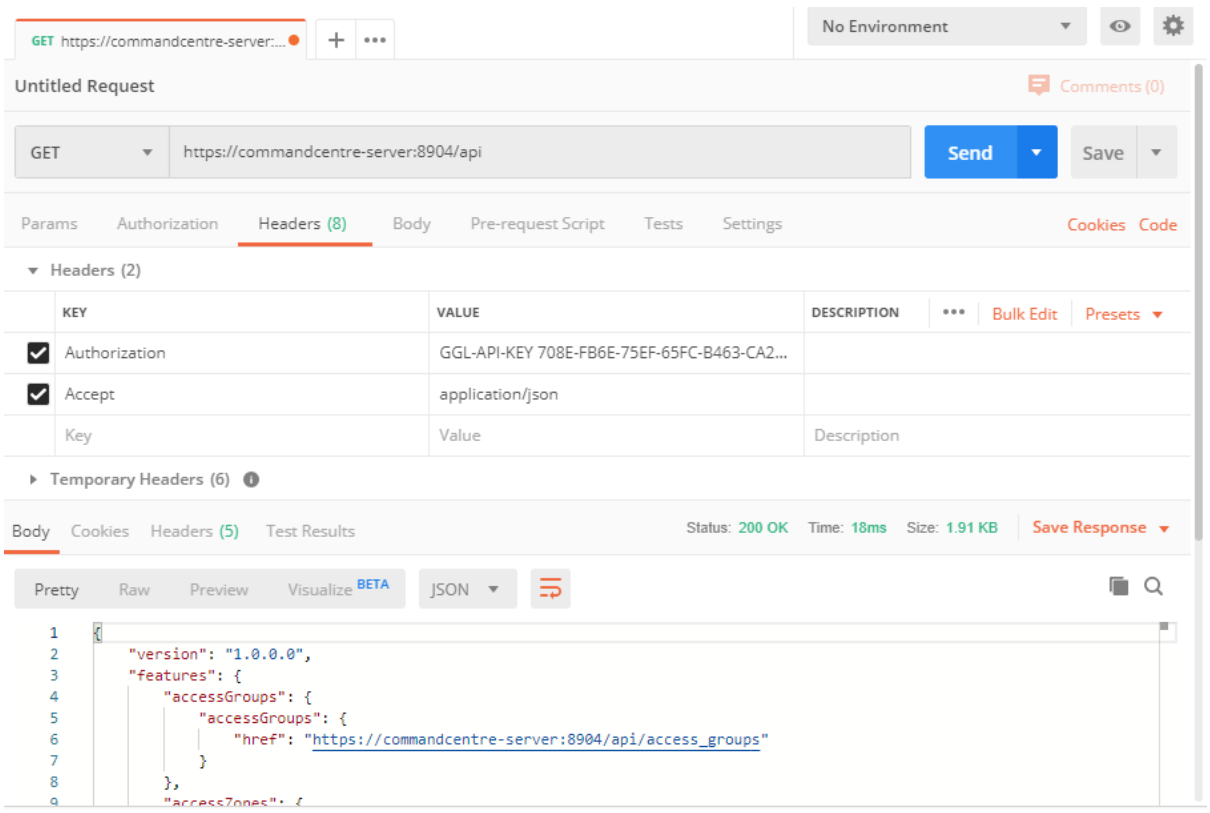
3.3 3. Perform your first request

In the request URL entry textbox, enter the DNS name or IP address of your server, and the port the REST api is configured to use.

For example: [`https://commandcentre-server:8904/api`](https://commandcentre-server:8904/api)

Replace "commandcentre-server" with the DNS name or IP address of your server, and if you have configured the REST api to use a different port from the standard 8904, change that as well

Click the **Send** button. You should see a response status of **200 OK** and you should see a JSON response showing your server version and some features



GET https://commandcentre-server:8904/api

Headers (2)

	KEY	VALUE	DESCRIPTION
✓	Authorization	GGL-API-KEY 708E-FB6E-75EF-65FC-B463-CA2...	
✓	Accept	application/json	
	Key	Value	Description

Status: 200 OK Time: 18ms Size: 1.91 KB

```
{
    "version": "1.0.0.0",
    "features": {
        "accessGroups": {
            "accessGroups": {
                "href": "https://commandcentre-server:8904/api/access_groups"
            }
        },
```

3.4 Troubleshooting

If you do not see a response like the above screenshot, please check the following

- Do you have the correct DNS name or is DNS not working? Does the request work with a direct IP address?
- Is there a firewall blocking your network requests to that port? Either at the network level or on the Command Centre server?
- Is the Command Centre REST API is enabled and licensed according to the **Getting Started with the REST API** document?
- Do you have the correct API key?
- Is Command Centre set to require client certificates?
- Do you have any other options set within postman? (for example, other headers)

4 Specific Task Walkthroughs

Once you have made a successful request to GET /api, we can proceed with specific tasks. You may not wish to try these in order, or you may wish to skip some. Pick the ones that interest you.

All examples need the API key header set as referenced above

4.1 Searching for Cardholders

If you perform a GET request on `https://commandcentre-server:8904/api/cardholders`, it will retrieve every single cardholder in the system which you are privileged to view, in pages of 100.

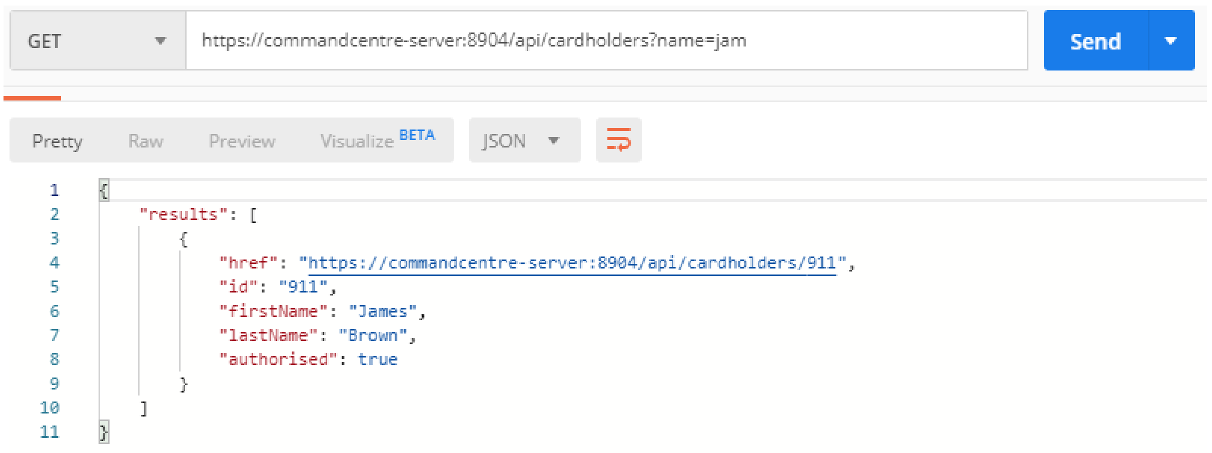
Usually you will want to search for a cardholder by their name or by a personal data value such as an Email Address.

4.1.1 Search by name

Append the query string **`?name=value`** to search for all cardholders with either a first or last name matching the substring xyz, such as

GET `https://commandcentre-server:8904/api/cardholders?name=jam`

To match all cardholders with "jam" in their first or last name



```
{
    "results": [
        {
            "href": "https://commandcentre-server:8904/api/cardholders/911",
            "id": "911",
            "firstName": "James",
            "lastName": "Brown",
            "authorised": true
        }
    ]
}
```

If you wish to match a name exactly, surround it in quotes, for example

GET `https://commandcentre-server:8904/api/cardholders?name="james"`

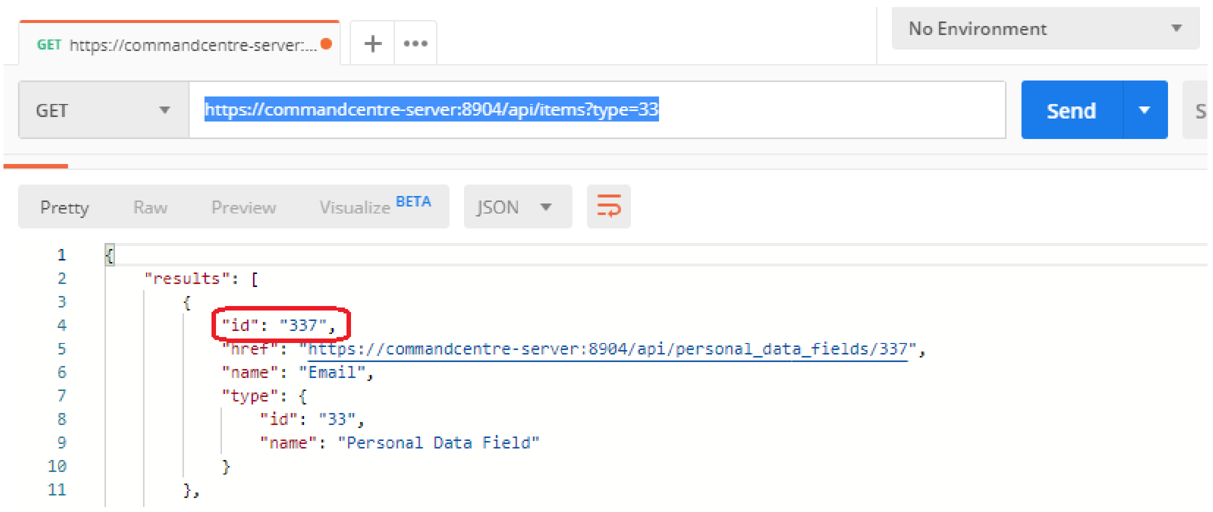
Please refer to the full REST api documentation for reference on searching, filtering, paging and other related topics.

4.1.2 Search by personal data value (such as Email)

In order to search by a personal data value, you must first find out what that personal data value's internal ID is. To find all your personal data field internal ID's, issue the request

GET `https://commandcentre-server:8904/api/items?type=33`

In the response JSON you should see all your personal data field types. Find the one you want and note down its ID field. In the example below, based on my test system we would note down the id of **337** (your value will be different)



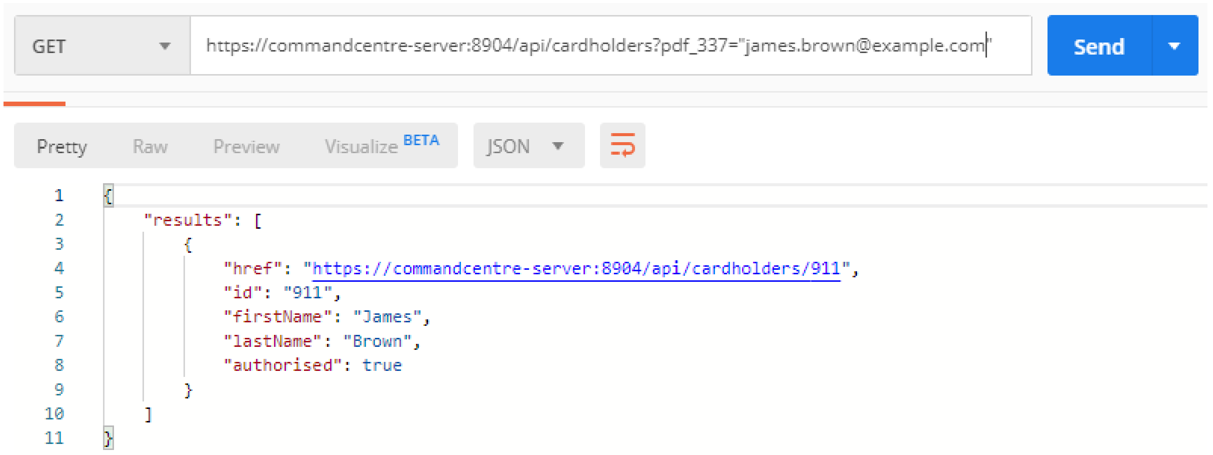
GET https://commandcentre-server:8904/api/items?type=33

```
{
    "results": [
        {
            "id": "337",
            "href": "https://commandcentre-server:8904/api/personal_data_fields/337",
            "name": "Email",
            "type": {
                "id": "33",
                "name": "Personal Data Field"
            }
        },
```

Next, we perform a cardholder search for **`?pdf_`*`ID`*`=value`**.

The following request will match all cardholders whose email address exactly equals "james.brown@example.com"

GET `https://commandcentre-server:8904/api/cardholders?pdf_337="james.brown@example.com`"



GET https://commandcentre-server:8904/api/cardholders?pdf_337="james.brown@example.com"

```
{
    "results": [
        {
            "href": "https://commandcentre-server:8904/api/cardholders/911",
            "id": "911",
            "firstName": "James",
            "lastName": "Brown",
            "authorised": true
        }
    ]
}
```

Note: Your system is unlikely to have 337 as its internal id. Remember to change that

Important: If you are searching for someone by email address, you should always use quotes for an exact match. Partial match on an email address is not recommended as you may find the wrong person by accident.

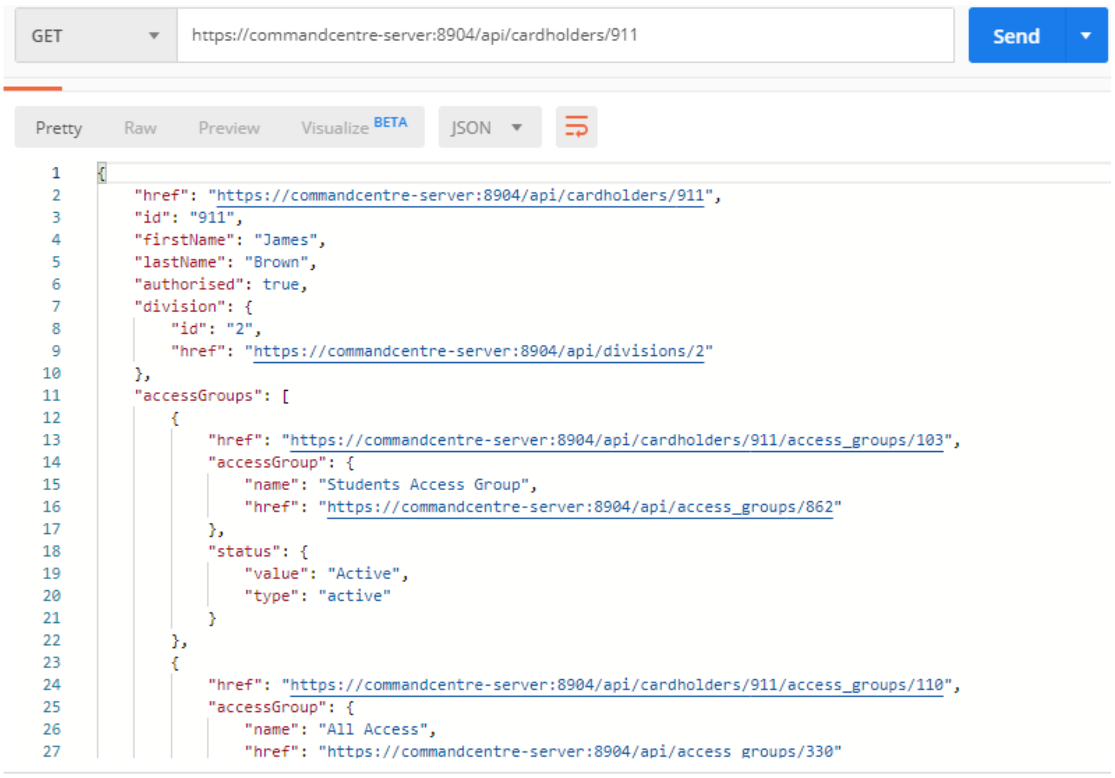
4.2 Getting Cardholder Details

Notice in the above examples, we searched for the cardholder James Brown. The search results only include basic information. Detailed fields such as cards, access groups, personal data values are not returned.

Notice the **href**. This is the key url for that cardholder. If you wish to read or modify that cardholder, you will use this key url. From the example above, the key url is https://commandcentre-server:8904/api/cardholders/911

Perform a GET request on the key url and you will retrieve all the information for that cardholder.

GET https://commandcentre-server:8904/api/cardholders/911



```
{
    "href": "https://commandcentre-server:8904/api/cardholders/911",
    "id": "911",
    "firstName": "James",
    "lastName": "Brown",
    "authorised": true,
    "division": {
        "id": "2",
        "href": "https://commandcentre-server:8904/api/divisions/2"
    },
    "accessGroups": [
        {
            "href": "https://commandcentre-server:8904/api/cardholders/911/access_groups/103",
            "accessGroup": {
                "name": "Students Access Group",
                "href": "https://commandcentre-server:8904/api/access_groups/862"
            },
            "status": {
                "value": "Active",
                "type": "active"
            }
        },
        {
            "href": "https://commandcentre-server:8904/api/cardholders/911/access_groups/110",
            "accessGroup": {
                "name": "All Access",
                "href": "https://commandcentre-server:8904/api/access_groups/330"
```

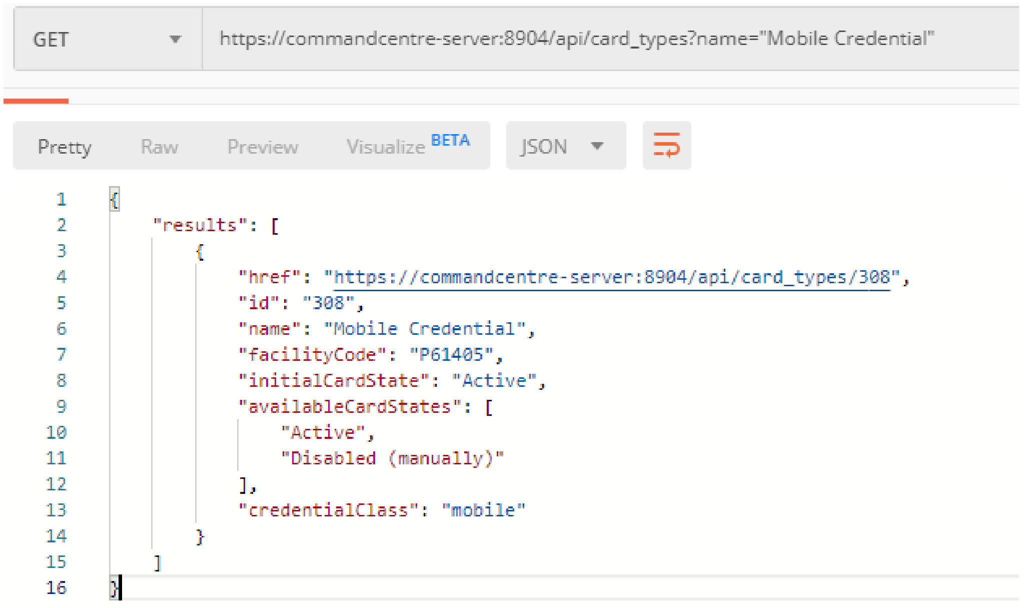
Please refer to the full REST api documentation for reference on all the different attributes a cardholder can have

4.3 Adding a Mobile Credential to a cardholder

Mobile credentials in command centre are represented in the same way as cards. You add a mobile credential by adding a "card" to a cardholder and specifying to use the "Mobile Credential" type.

First you must find the key URL for the "Mobile Credential" card type.

```
GET https://commandcentre-server:8904/api/card_types?name="Mobile Credential"
```


GET https://commandcentre-server:8904/api/card_types?name="Mobile Credential"

```
{
    "results": [
        {
            "href": "https://commandcentre-server:8904/api/card_types/308",
            "id": "308",
            "name": "Mobile Credential",
            "facilityCode": "P61405",
            "initialCardState": "Active",
            "availableCardStates": [
                "Active",
                "Disabled (manually)"
            ],
            "credentialClass": "mobile"
        }
    ]
}
```

Note down the key url from the **href** field in the response.

In the above example, the key url for the mobile credential card type is `https://commandcentre-server:8904/api/card_types/308`

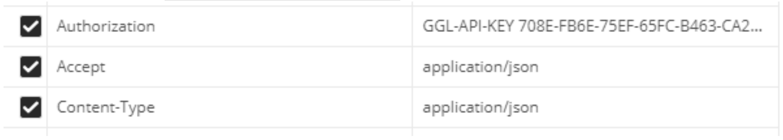
Now, we must edit a cardholder and ask Command Centre to add a card for them, specifying that type.

Input the key URL of the **cardholder** into the URL box in postman. Remember from above our cardholder James Brown had the key URL of https://commandcentre-server:8904/api/cardholders/911

Change the request type to PATCH

Switch to the **Headers** tab (if you cannot see this you may need to scroll up) and add a new header

- Content-Type: `application/json`

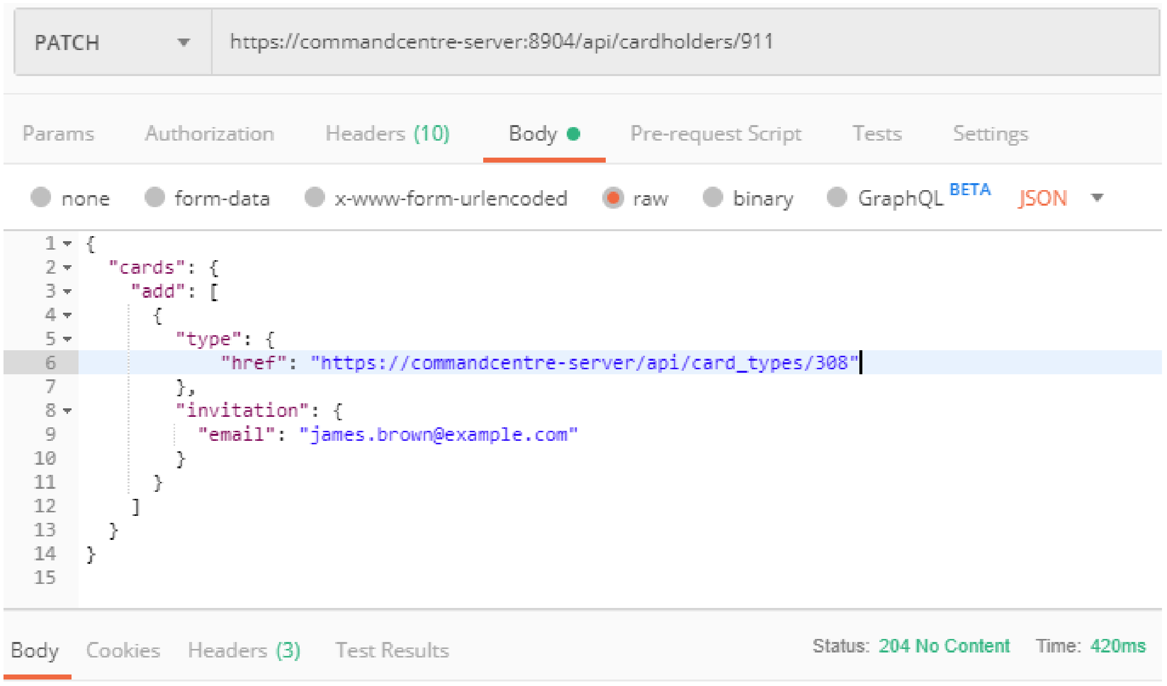


✓	Authorization	GGL-API-KEY 708E-FB6E-75EF-65FC-B463-CA2...
✓	Accept	application/json
✓	Content-Type	application/json

Switch to the **Body** tab, select the **raw** radio button and input the following in the request body textbox

```
{
  "cards": {
    "add": [
      {
        "type": { "href": "<mobile credential card type key url>" },
        "invitation": { "email": "<target email address>" }
      }
    ]
  }
}
```

Remember to replace the mobile credential card type key URL and the email address you would like to send the credential to. You should see a **204 No Content** response which indicates success



https://commandcentre-server:8904/api/cardholders/911

```
{
  "cards": {
    "add": [
      {
        "type": {
          "href": "https://commandcentre-server/api/card_types/308"
        },
        "invitation": {
          "email": "james.brown@example.com"
        }
      }
    ]
  }
}
```

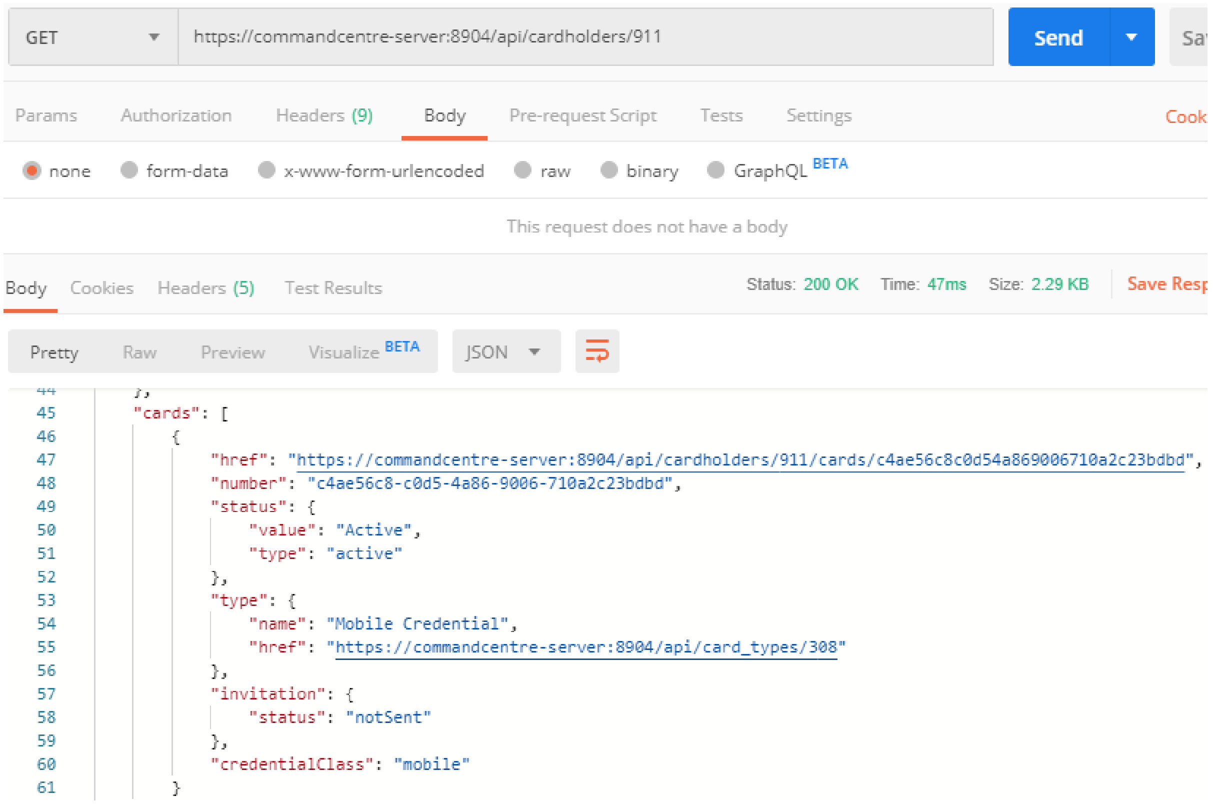
Status: 204 No Content   Time: 420ms

Check your email. If your server is connected to the Command Centre cloud, you should see one within a minute or so.

4.4 Removing a Mobile Credential from a cardholder

To remove a mobile credential, we must first figure out which credential we want to remove as a cardholder could have more than one.

Start by doing a GET on the key URL for your cardholder, then scroll down until you find the "cards" section



```
    },
    "cards": [
        {
            "href": "https://commandcentre-server:8904/api/cardholders/911/cards/c4ae56c8c0d54a869006710a2c23bdbd",
            "number": "c4ae56c8-c0d5-4a86-9006-710a2c23bdbd",
            "status": {
                "value": "Active",
                "type": "active"
            },
            "type": {
                "name": "Mobile Credential",
                "href": "https://commandcentre-server:8904/api/card_types/308"
            },
            "invitation": {
                "status": "notSent"
            },
            "credentialClass": "mobile"
        }
```

Tip: If you want to reduce the amount of data to search through, you can ask Command Centre only to send you certain fields

GET https://commandcentre-server:8904/api/cardholders/911?fields=href,cards

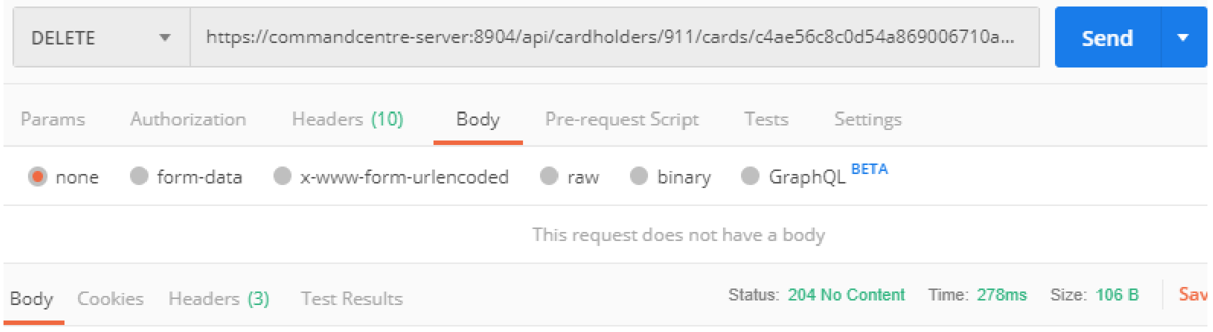
Please refer to the full REST api documentation for reference.

In the screenshot above, We can see the mobile credential we added above. In this instance there is only a single mobile credential and no other cards. If there were multiple credentials or cards we would need to work out which one we wanted.

Note: The credential also has a key URL. This is how we will remove the credential. Make note of the key URL, then issue a DELETE request to it

DELETE https://commandcentre-server:8904/api/cardholders/911/cards/c4ae56c8c0d54a869006710a2c23bdbd

If successful, you will see a response of **204 No Content**



To verify this, you can either look in command centre, or repeat the GET on the cardholder's key URL.

```
GET https://commandcentre-server:8904/api/cardholders/911
```

And you will find that the credential has been removed from the "cards" array. If there are zero cards remaining, then the "cards" array will not be returned at all.