Gallagher Command Centre

Mobile Connect SDK Integration Guide

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The information is indicative only and is subject to change without notice meaning it may be out of date at any given time.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.

Contents

1 Background

The Gallagher Mobile Connect App uses Bluetooth® Low Energy or NFC to exchange data with Gallagher T-Series readers, which in turn talk to Gallagher 6000 series controllers to grant or deny someone access.

The Mobile Connect SDK is a redistributable package which provides the functionality of the Mobile Connect App. Third party developers can incorporate this functionality into their own Mobile applications for branding, functionality or other purposes.

In addition to bundling Mobile Connect functionality into third party apps, the third party back-end systems must use the Command Centre REST API to provision mobile credentials and distribute them to their mobile apps.

This document primarily covers design and architectural patterns for incorporating Command Centre, third-party back-end systems, third-party mobile apps, and the Mobile Connect SDK together.

1.1 Target Audience for the Mobile Connect SDK and REST API

The Mobile Connect SDK is for use by other software developers, who will need to have the skills and tools to develop mobile applications and incorporate our SDK package according to our developer instructions.

Likewise, the Command Centre REST API is for use by other software developers, who will need to have the skills and tools to integrate with it.

This document is addressed to software developers working for or on behalf of other organizations which use Command Centre as their security system and wish to develop mobile applications to integrate with it. Henceforth the document will refer to "you" as the third-party software developer, and "your" things – meaning things associated with that organization.

1.2 Assumptions/Prior Knowledge

You should be least somewhat familiar with Command Centre and how to configure it. As such, this document does not go into detail of how to configure access zones, cardholders, etc.

You should be familiar with the Gallagher Mobile Connect application and the Mobile Credential registration process. More information can be found in the **Mobile Connect Cloud and App Security TIP** document which should accompany this document.

It is assumed that your mobile application will connect to a backend service of some kind. That backend service can then in turn connect to the Command Centre server via its REST API. Standalone mobile applications without a backend service are not supported; you should not attempt to have your mobile application connect directly to the Command Centre REST API.

1.3 Security

The Mobile Connect SDK has the same security properties as the Mobile Connect app itself; Please refer to the **Mobile Connect Cloud and App Security TIP** document

2 Mobile Connect SDK System Architecture

2.1 Mobile Credential Registration

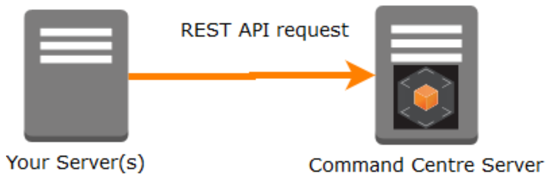
If you have not already, please read the **Mobile Connect Cloud and App Security TIP** document and familiarize yourself with the credential registration process. It explains how credentials are issued to mobile devices from Command Centre using Gallagher cloud services.

The SDK has the same behavior, and so also requires a connection to Gallagher cloud services in order to register mobile credentials, however the way that the invitation gets from Command Centre to the mobile device (Email and SMS for Mobile Connect itself) is the responsibility of your own systems.

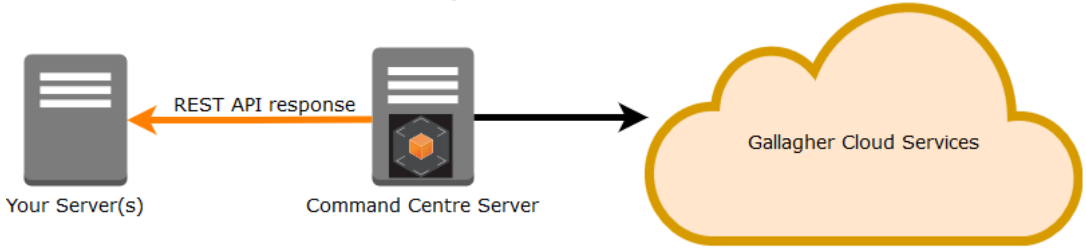
The SDK has been designed this way because the majority of third party use-cases we discovered want to take control of the credential registration process themselves, and not have emails coming from Gallagher.

The registration process as designed for the SDK as follows:

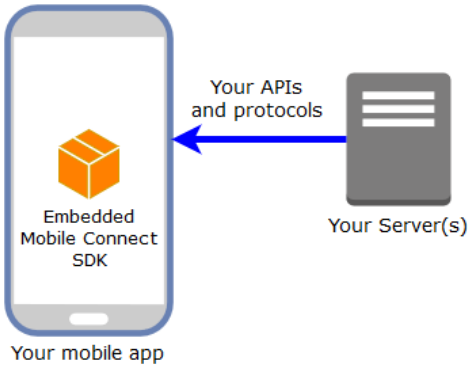
1. You use the REST api to create a mobile credential



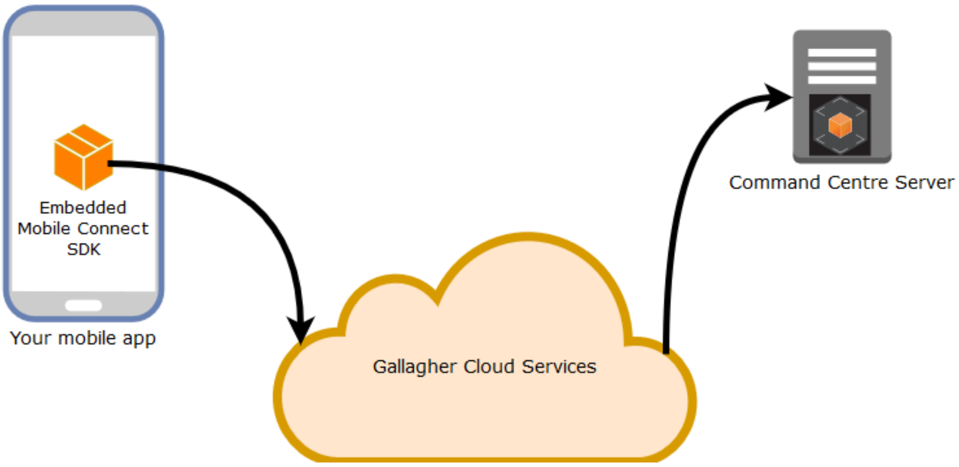
2. Command Centre sends the credential invitation to the Gallagher Cloud and returns the invitiation URL to you via the REST API



3. You send the invitation URL to your mobile device, and pass it to the registerCredential method in the SDK



4. The SDK registers the credential, and sends the internal details to the Gallagher Cloud, which in turn sends it to the Command Centre Server



2.1 Mobile Access

After a credential is registered, the Command Centre server will push that credential down to all appropriate Gallagher Controllers (depending on where the person should have access to).

At access time, the Mobile Device communicates directly with Gallagher Reader hardware using Bluetooth or NFC. The Reader relays commands back to a Gallagher Controller, which will grant or deny access based on the registered credential which it already has.

The access process takes place entirely offline and requires no connectivity other than the local Bluetooth or NFC connection. This process is entirely managed by the Mobile Connect SDK, and other than instructing the SDK to start and stop scanning for Readers, does not require any input from your mobile app.

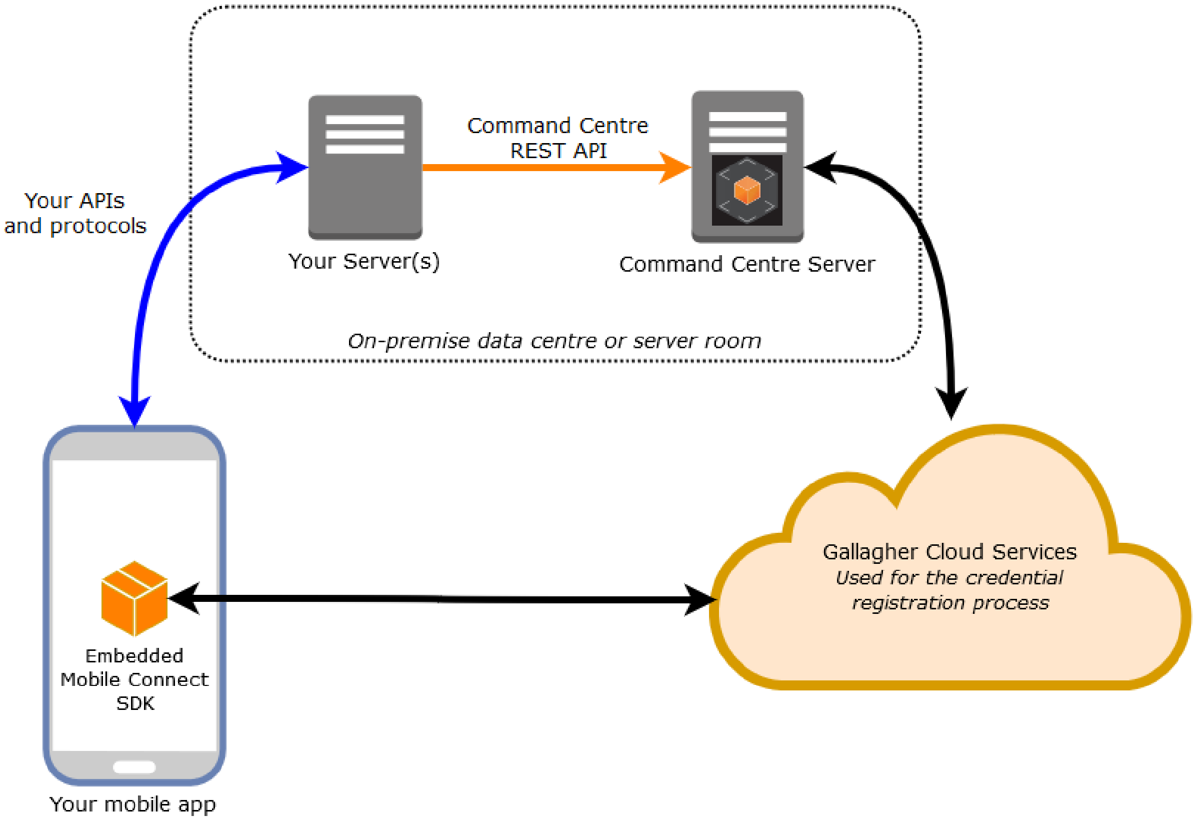
3 Integration Architecture

There are multiple ways in which you could integrate Command Centre with your systems. The best option depends on where your services are located relative to Command Centre, and your network requirements.

3.1 Your Mobile Application connects to on-premise services

Command Centre is generally an on-premise system; it is conventional that your organization will have some sort of server room or company-controlled hosting facility where the Command Centre server will reside, along with other servers and IT resources owned by your organization.

If your mobile application's backend services are hosted on a server which is also on-premise, then those backend services can connect directly to Command Centre's REST API, as follows:



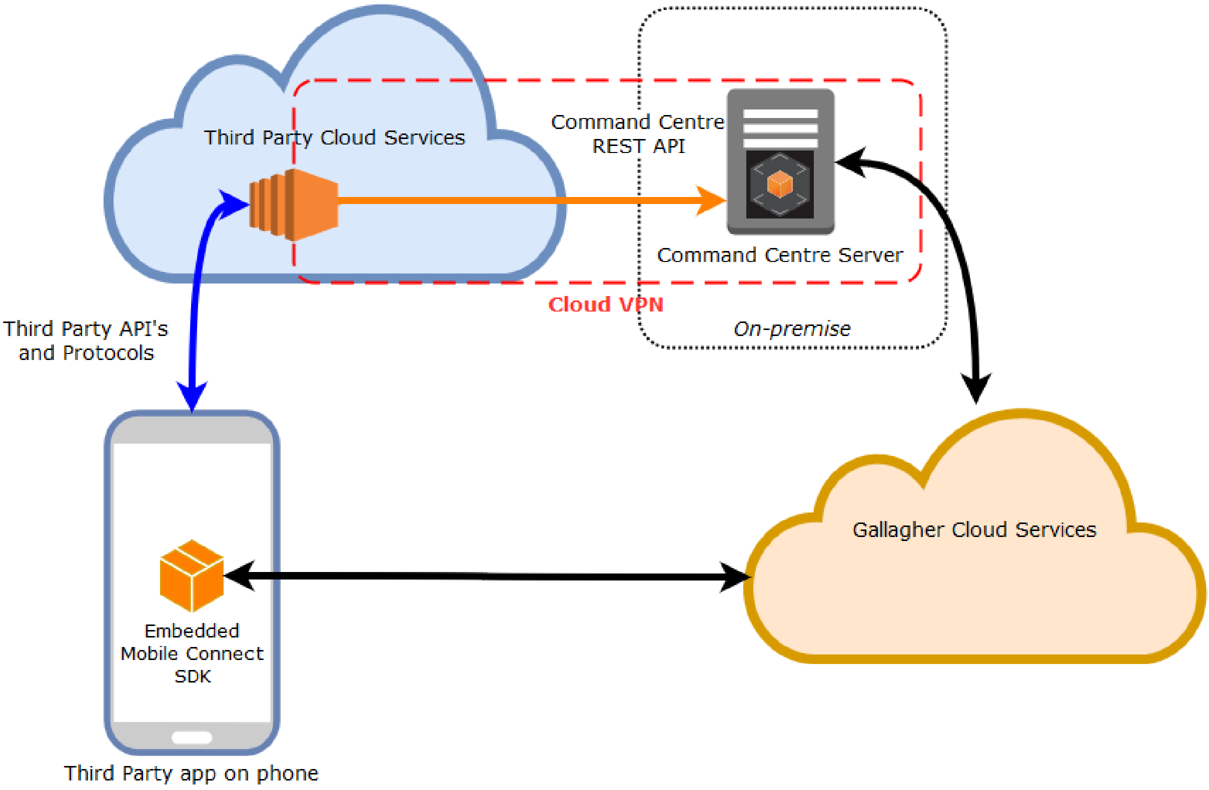
3.2 Your Mobile Application connects to cloud services

Generally speaking, allowing inbound connections from the cloud to internal services is considered a security risk, so if your mobile application's backend services are hosted in the cloud then it is unlikely they can connect directly to the Command Centre REST API.

Depending on your cloud provider, security requirements and other organizational decisions, you may have different options for establishing a connection.

3.2.1 VPN to bridge your cloud and on-premise servers

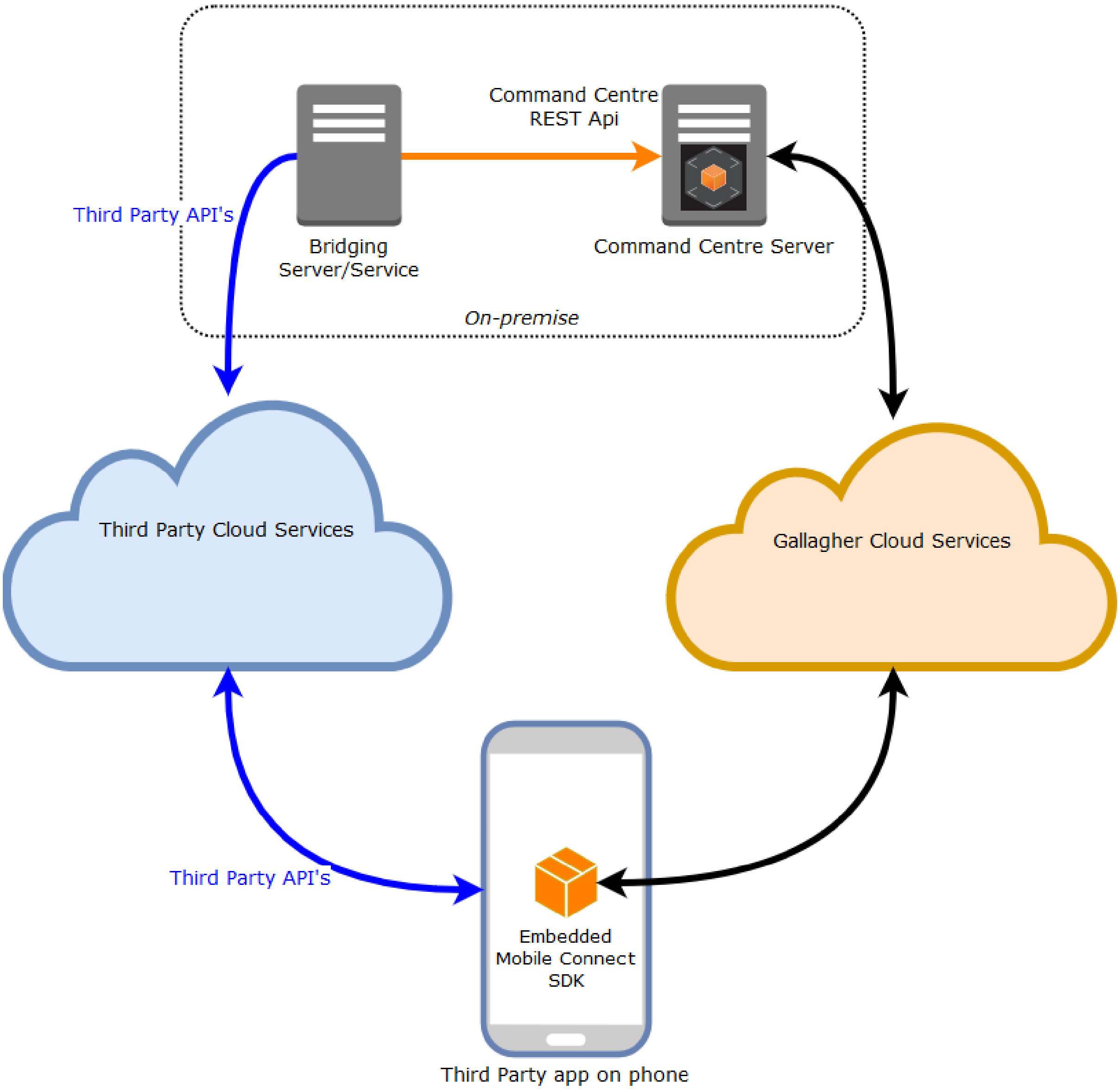
Using a technology such as an Amazon Web Services Managed VPN (https://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/vpn-connections.html) or Microsoft Azure VPN Gateway (https://azure.microsoft.com/en-us/services/vpn-gateway/), you can create a secure link between your cloud resources and your on-premise Command Centre server. Once this secure link is in place, your cloud services can connect to Command Centre's REST API in the same way as they would for an on-premise deployment



3.2.2 Manually bridge your cloud and on-premise servers with an extra service

Given that your cloud services can't connect *in* to Command Centre, and Command Centre can't connect *out* to your cloud services, you can create a separate piece of software which will connect to both your cloud services and Command Centre, and relay commands between the two. This software could be run on a separate physical or virtual server or could also run on the Command Centre server itself.

While this requires additional work, it may be simpler to set up and manage than a Cloud VPN and can serve a useful role in translating between Command Centre's REST API and your own.



4 Key design consideration: Mapping users across systems

You must determine a way to cross-reference people (Command Centre refers to them as cardholders) between Command Centre and your own systems. Each person must have a unique primary identifier which is present in both systems.

4.1 Primary Identifier defined by your system, Command Centre stores a copy

Your system is likely to have its own concept of how to uniquely define a person or user account. Examples include:

- Student ID
- Employee Number
- Internal Database ID from your database

You can create a Personal Data field in Command Centre and use it to record this identifier against each cardholder.

This approach requires no change to your existing data structures. It does require a separate process to attach your external identifier to cardholders within Command Centre and keep things up to date over time.

This option is recommended for situations such as:

- You have an existing system and are already importing cardholders into Command Centre via one of its other non-REST API import features (such as XML import or Enterprise Data Interface).
- You have a new system but have another requirement to import cardholders into Command Centre via one of its other non-REST API import features.
- It is not acceptable to add any more data fields to your system

Once Command Centre has this identifier loaded by some import mechanism, you can use it to locate cardholder records using the REST API by doing a personal data field search, then following the search results to the cardholder records.

For details please refer to the REST API documentation and the Gallagher University Student Portal Sample integration.

4.2 Primary Identifier defined by Command Centre, you store a copy

The primary identifier for a cardholder in the Command Centre REST API is its URL, or HREF, for example http://servername/api/cardholders/19238.

If you can alter your system to store this URL against each user, then you can use it to directly modify or delete the cardholder record without first having to do personal data field searches. This is more performant and requires less development effort.

This option is recommended for situations when:

- You will be using the REST API to create or import cardholders into Command Centre.
  (When you create the cardholder via the REST API, the URL will be returned; You can store it and use it thereafter)

For details please refer to the REST API documentation and the sample integration.

4.3 Both systems store a copy of each-other's Primary Identifier

Command Centre can have your external identifier stored in a personal data field, and you can also store Command Centre cardholder URLs in your system.

Under this design, you would use the stored Command Centre cardholder URL if known, but if it is not present then you would fall back to using a personal data field search, assuming that your primary identifier had been loaded into a personal data field in Command Centre already. Once you do the personal data field search you will be told the cardholder URL which you can re-use in future.

This may be required if you are working in a complex environment, such as when some cardholders are added to Command Centre via the REST API, but other cardholders are added via another import mechanism, or perhaps manually via a person using a Workstation.

You may also use it as an optimization in situations where you can speed things up by storing the URL the first time you encounter it (after a personal data field search) and use this to avoid future searches. This is effectively doing a PDF search + a cache.

This option is the one taken by the sample Integration.

5 Setting up Command Centre

Command Centre will need to be configured as follows

- The REST API must be enabled.
- An Operator account must be created in Command Centre to represent the privileges and access your third-party service needs.
- A REST client API key must be generated and associated with this Operator account.
- A client certificate should be created and used to authenticate against the Command Centre REST API.

Please refer to the REST API documentation in the Configuration Client’s online help for details on how to do this.

6 Creating Cardholders

If you have determined that cardholders are going to be added to Command Centre by some other import mechanism, you can ignore this; Note that as mentioned above, you will need to use a personal data field search to look up cardholders via an identifier that your system defines (such as Student ID).

Note: The sample integration uses an external "Student ID" personal data field to look up existing cardholders, it does not demonstrate how to create cardholders

If you have determined that your integration is going to use the REST API to add cardholders to Command Centre, you can achieve this by submitting an HTTP POST with appropriate JSON content to the create cardholder URL. See the Cardholders section of the REST API documentation for more information.

If you submit valid data, then Command Centre will create the cardholder and return HTTP 201 (created) with an HTTP "Location" header referencing the URL of the cardholder; you can save this in your system to easily modify or delete the cardholder record in future.
Note that if you take this approach, there is no need to store any specific external identifier (such as Student ID) in command centre; thus avoiding the work of setting up and managing personal data fields.

For details on how to do this please refer to the REST API documentation.

7 Managing Mobile Credentials

7.1 Listing credentials

When you retrieve a cardholder record via its URL, you will be given all of its Cards.

Command Centre considers mobile credentials to be a special form of access card, hence their inclusion under the "cards" section of the cardholder data structure. You can distinguish a Mobile Credential from other kinds of card either by looking at the Card Type name string, or by testing for the presence of an "invitation" sub-object.

Example JSON – part of a cardholder record:

```
"cards": [
    {
        "href": "https://server:8904/api/cardholders/501/cards/...",
        "number": "615fe35c-cfbe-459b-998a-e898e4c687c9",
        "status": {
            "value": "Active",
            "type": "active"
        },
        "type": {
            "name": "Mobile Credential", // Clue: a mobile credential
            "href": "https://server:8904/api/card_types/..."
        },
        "invitation": { // Only mobile credentials have "invitation"
            "href": "https://...security.gallagher.cloud/api/invitations/...",
            "status": "sent"
        }
    }
],
```

For details please refer to the REST api documentation and the sample integration.

7.1.1 Card Status and Invitation Status

All Cards and Credentials in Command Centre have a status, which is visible as part of the Card JSON object above.
The status will be "Active", "Disabled Manually", "Expired", or another custom value such as "Lost". This status controls whether the credential can be used for access. Only "Active" cards and credentials will be granted access.

Mobile credentials specifically also have a second "invitation status", which reflects the invitation/registration process for mobile credentials. It will take one of the following values:

notSent	The credential invitation has been created and is queued in Command Centre, but it has not yet been sent to the Gallagher Cloud. Unless there is a network issue, it is likely that credentials will only be in the "notSent" state for less than one second. This credential should be considered pending. In this state, the invitation URL/HREF is not yet available.
sent	The credential invitation has been sent to the Gallagher Cloud. We are now waiting for recipient to open the mobile app on their phone and follow through and register the credential. This credential should be considered pending. In this state, the invitation URL/HREF is available. This is the key piece of data required in the mobile app to register the credential. It is likely you will want to take the invitation URL and send it through to your mobile application via your existing services/accounts/secure channels. Once it arrives on the phone, your mobile application should supply it to the registerCredential method in the Mobile Connect SDK.
expired	The credential invitation was sent to the Gallagher Cloud, but the recipient did not follow through and register the credential before the expiry period elapsed. This credential should be considered invalid. It should either be re-issued or deleted. The credential invitation expiry period defaults to 7 days and is configurable in Command Centre.
accepted	The credential invitation was sent to the Gallagher Cloud, the recipient has registered it on their phone, and the result has been relayed back to Command Centre. This credential should be considered valid and may be used to gain access so long as the root credential status is "Active".

7.2 Adding Credentials

HTTP PATCH to the cardholder's URL (or include as part of the creation of the cardholder) an instruction to create a card. You will need to specify the URL of the Card Type, which you must obtain ahead of time.

Example JSON – part of PATCH to the cardholder record:

```
"cards": {
    "add": [
        {
            "type": {
                "href": "https://server:8904/api/card_types/..."
            }
        }
    ]
}
```

7.2.1 Options when adding credentials

As above, when you submit a request to add a card, the minimum "card object" contains the "type" href to identify which Card Type to use in Command Centre.

You can control credential invitation behavior by populating an "invitation" object under the "card" object:

```
{
    "type": {
        "href": "https://server:8904/api/card_types/..."
    },
    "invitation": {
        "email": "user@something.com", // not recommended
        "mobile": "+6401234567", // not recommended
        "singleFactorOnly": true // not recommended
    },
}
```

The invitation object is optional, and all its properties are also optional.

email: If you set this property, the Gallagher Cloud will send an email containing the invitation code to the specified email address. Note that for integrations, it is expected that you would send the invitation code via your own systems and would not populate the email value here – however, it may be useful for testing and development

mobile: If you set this property, the Gallagher Cloud will send an SMS message with a 6-digit one-time-code as part of the credential registration process. This is only supported for use with the Gallagher Mobile Connect application, and not for third-party mobile applications.

singleFactorOnly: If you set this property, then the credential will NOT register a second factor on the phone. Second factor means a Fingerprint/FaceID, PIN or Passcode, and is required to gain access to areas which are configured to be in PINs mode in Command Centre.
You may wish to set **singleFactorOnly** if you are sure that your solution does not need second-factor access, as it will bypass the PIN/Fingerprint user-interface part of the registration workflow on the phone. Note that if you later decide that you do want second-factor access, you must remove and re-issue new credentials to all affected mobile phones.

For further details including setting card activation dates, expiries, and other common properties please refer to the REST api documentation.

7.3 Deleting Credentials

When you get a cardholder record, as part of listing cards/credentials, each card/credential will have its own URL/HREF. From the example above:

```
"cards": [
    {
        "href": "https://server:8904/api/cardholders/501/cards/...",
         ...
```

To remove an individual credential, issue an HTTP DELETE to that specific URL.

You may also issue an HTTP PATCH to the cardholder URL, with an instruction to remove the credential. This may be used to remove multiple credentials in a single request.

Example JSON – part of a PATCH cardholder request:

```
"cards": {
    "remove": [
        {
            "href": "https://server:8904/api/cardholders/501/cards/..."
        }
    ]
}
```

Note: Once a credential is removed, it cannot be restored without repeating the registration process involving the end user and their phone.

Note: Once a credential is removed, it will no longer be trusted by Command Centre and will be denied access and will no longer count towards license limits.
You may opt to also delete it from the mobile app on your phone for housekeeping purposes, but there is no need from Command Centre's perspective to do this.

For details please refer to the REST API documentation and the sample integration.