

CMID RESTful API Developer's Guide

Version 1.0, July 2019

Table of Contents

1. Overview	1
1.1. About This Guide	1
1.1.1. Conventions	1
1.2. About CMID DA Core Service	2
1.3. Introduction to CMID REST API	4
1.3.1. RESTful API	4
1.3.2. REST Client	4
2. Getting Prepared	5
2.1. Setup Overview	5
2.2. System Requirements	6
2.3. Installing DA Core	6
2.4. Setting Up DA Core	8
2.5. Running Sample Demo	9
3. Getting Started	11
3.1. Authentication	11
3.1.1. Using Sample Program	12
3.1.2. Using Other REST Clients	15
3.2. Making First Request	16
3.2.1. Using Sample Program	16
3.2.2. Using Other REST Clients	18
4. Tutorials	21
4.1. Managing Device	21
4.1.1. Adding Device	22
4.1.2. Reading Device	24
4.1.3. Updating Device	26
4.2. Registering User	27
4.2.1. Adding User	27
4.2.2. Creating User Permission	31
4.3. Getting Logs	33
4.3.1. References for Getting Logs	34

1. Overview

CMID Manager V2 is an integrated management solution that allows users to manage access control and time & attendance for employees under the biometric security environment. It has a client-server architecture that is composed of a central DA (device agent) Core server and clients that utilize the DA core service. It is originally designed and intended to work with biometric devices manufactured by CMITech Co. Ltd., such as EF-45.

1.1. About This Guide

This guide contains the conceptual information, the procedural information, and other related references about CMID Manager DA Core and CMID REST API. It is intended and written for software developers who are in charge of application development and system integration on various platforms.



If you are new to the CMID Manager V2 solution, we recommend that you familiarize yourself with [the CMID Manager V2 Admin Guide](https://app.box.com/s/8m38x7i6961ikbl4z0swfxt8cgjjg6p6) [https://app.box.com/s/8m38x7i6961ikbl4z0swfxt8cgjjg6p6] first that covers the comprehensive use cases and instructions for using the software.




- The figures and screenshots in this guide are given for illustration purposes only and may differ from the actual product.
- Due to continuous technological improvements, the guide may not contain the most updated information. For further information not covered in this guide, please contact us at service@cmi-tech.com [mailto:service@cmi-tech.com].

1.1.1. Conventions

The following symbols are used throughout this guide. Make sure that you fully understand each symbol and follow the instructions accompanied.

Symbol	Meaning	Description
	CAUTION	Indicates a situation that demands prompt and careful action, a specific remedy, or emergency attention.
	IMPORTANT	Indicates important information that let users know the possible cause of a potential problem in a specific situation and the solution.

Symbol	Meaning	Description
	NOTE	Indicates necessary or useful information that generally does not requires actions.

1.2. About CMID DA Core Service

The DA Core service works as a core system of the CMID Manager V2 and provides basic backend functions for main service, device, and database management. It incorporates three management modules—Device Manager, Database Manager and Network Manager.

Device Manager

The Device Manager acts as the main service framework of the DA Core and provides a set of components necessary for user and device management.

Network Manager

The Network Manager provides the network interface for the clients and devices, transparently handling connections over TCP/IP socket or REST service.

Database Manager

The Database Manager provides database service to clients based on Maria DB management system. It maintains a database connection pool to improve performance and synchronize database upon CRUD operations.

Each module has access to the database depending on its needs and retrieves and manipulates data when the relevant events occur.

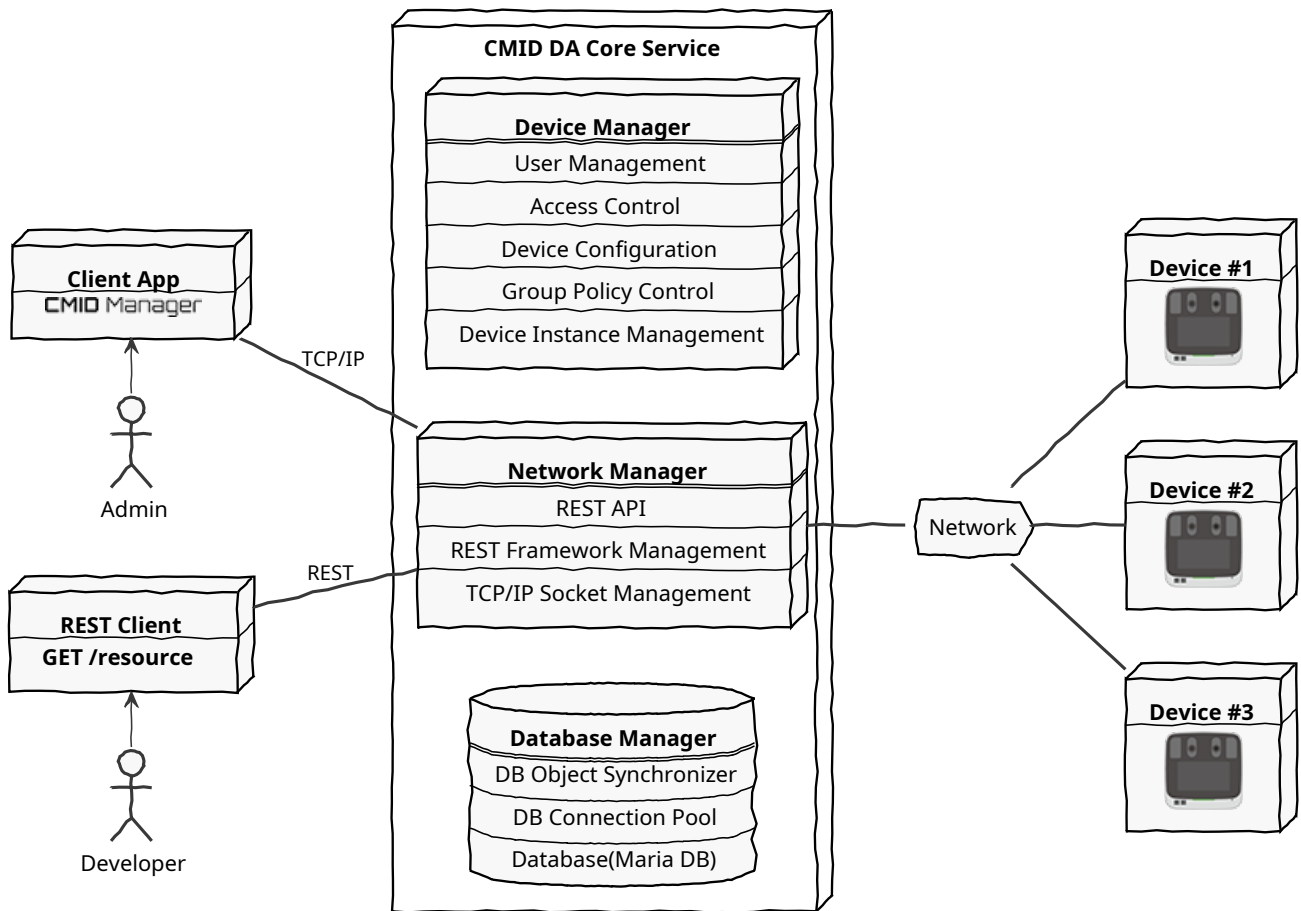


Figure 1. CMID Manager V2 Architecture

There are two types of clients that use the services provided by DA Core:

CMID Manager client

The CMID Manager client is an off-the-shelf program that the CMID Manager V2 software package comes with. It is intended for system administrators and poses a GUI-based application that allows them to connect to the DA core and perform their common tasks in the front-end.

REST client

REST client is a testing tool that helps developers to create and test HTTP requests according to the relevant REST API specification.

1.3. Introduction to CMID REST API

The CMID Manager V2 also provides an API (application programming interface) to give developers the ability to create and build their own applications or custom integrations with CMID solution.

1.3.1. RESTful API

Because the CMID API is organized around REST (Representational State Transfer) architectural style, it can be accessed via HTTP protocol at a predefined set of endpoints or URLs. With predictable resource-oriented URLs, the CMID REST API accepts request bodies and returns responses in JSON format, and uses standard HTTP response codes, authentication, and methods like GET, POST, PUT, and DELETE.

The RESTful API is designed to be available from any programming languages. Thus, once you come to grasp the basic concepts of the API, you will be able to apply them to any platform by using your favorite HTTP/REST library for your programming language.

Now that the business logic and internal complexity are taken care of by CMID DA Core Service, you can focus on how to achieve your goals by identifying and consuming the available resources and actions properly.

1.3.2. REST Client

There are several test tools available that do the work of REST Client for calling REST APIs. For example, you can use GUI REST Clients in desktop application or web browser extension such as Postman or command-line tools like cURL.

We also provide a simple REST program along with a sample project written in C# in order to let you have hands-on practice on our API and to give you an aid for understanding how an application using REST API can be implemented.

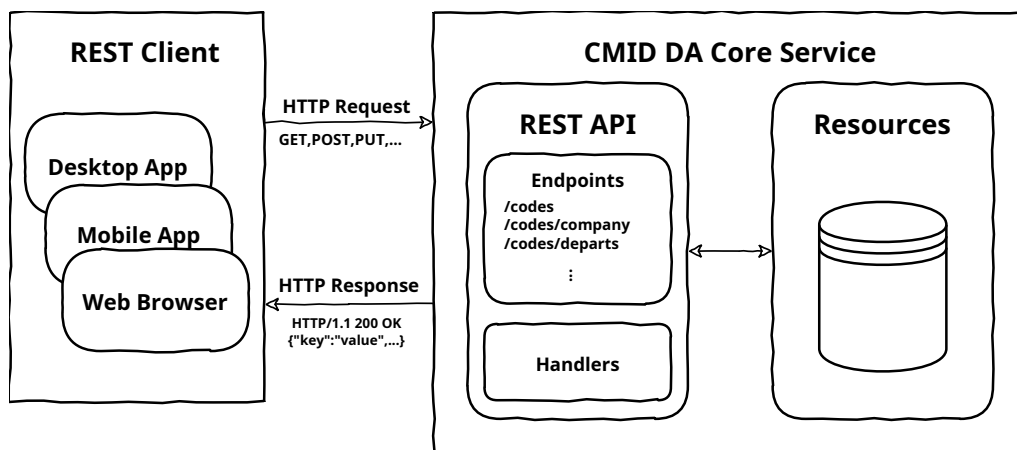


Figure 2. CMID REST API

2. Getting Prepared

This section gives the information about the system requirements, the installation procedures, and the environment setup procedures before using CMID REST API.

2.1. Setup Overview

You need to install the CMID DA Core service modules—DA Core installer and DB installer—that are included in the CMID Manager V2 installation package.

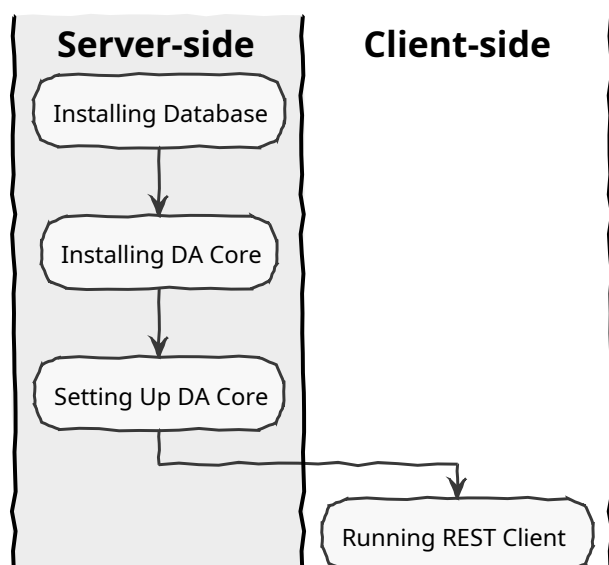


Figure 3. Setup Overview

Before installation, make sure that you check the following prerequisites:

- Prepare a Workstation or a PC where you install the DA core service modules.
- Make sure that the Workstation or the PC meets the system requirements.

2.2. System Requirements

The system requirements for the CMID DA Core service and the sample project are as follows:

	Items	Minimum	Recommended
DA Core Service	Operating System	Windows 7	Windows 10 (64bit)
	CPU	Intel Core i5 series processors or AMD equivalent	
	RAM	4GB	16GB or higher
	Disk Drive	100GB HDD	256GB SSD or higher
Sample Project	Database	<ul style="list-style-type: none"> • MariaDB 10.2.12 or higher • MySQL 5.6, 5.7 	
	Development Environment	<ul style="list-style-type: none"> • Visual Studio 2013 • .Net Framework 3.5 with C# 	

2.3. Installing DA Core

You must install the CMID DA Core in the Workstation or the PC that you choose as the server machine. You can install the DA Core modules by running two installers for Database and DA Core in a row.



- If you have a previous version of CMID DA Core installed, remove the old version before running the CMID DA Core installers.
- Do not remove or reinstall the Database Setup Program (**CMV_DB_Setup_x.x.x.x.exe**) if you want to keep the current database. Otherwise, you will lose all the data that are stored in the database.

1. Double-click **CMV2_DB_Setup_x.x.x.x.exe** to run the Database installer.



- If the User Account Control warning message window appears, click **OK** or **Yes** to continue.
- "x.x.x.x" stands for version identifiers and may vary depending on the software version

2. Click **Next**.

3. Click **Install** to start the installation.

4. Click **Finish** to exit the installer.
5. Double-click **CMV2_DA_Core_Setup_x.x.x.x.exe** to run DA Core installer.



If the DA Core installer version is lower than 2.1.2.2, the REST service does not work. Contact us at sales@cmi-tech.com [mailto:sales@cmi-tech.com] to get the latest version.

6. Click **Next > Install > Finish**.



Antivirus Exclusion Recommended

After installation, we recommend that you add DA Core program (for example, *C:\Program Files (x86)\CMITECH\CMV2\DA_Core\Bin\CMV2_DA_Core.exe*) to the exclusion list of your Antivirus software, if any, to avoid any potential issues. For how to configure antivirus exclusions, refer to your Antivirus software user guide or help.

2.4. Setting Up DA Core

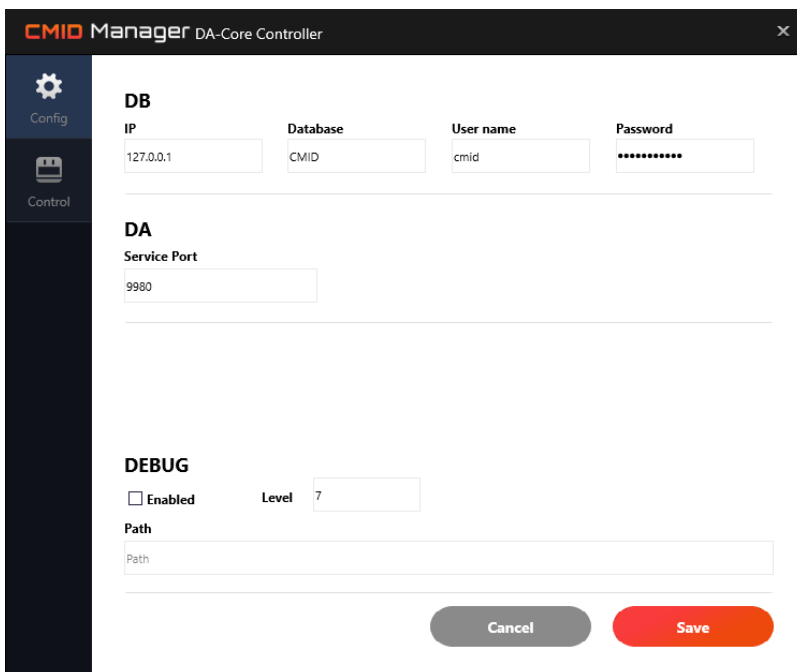
When the installation of the DA Core is completed, you should run CMV2 DA Core Controller first in the server machine to configure basic DA Core settings.

1. Open **Control Panel** in Windows.
2. Select **Large icon** or **Small icon** in **View by** drop-down menu.
3. Click **CMV2 DA Core Controller (32-bit)** to run the DA Core Controller.



If the User Account Control warning message window appears, click **OK** or **Yes** to continue.

4. Type the IP address of Device Agent in **IP** box under **DB**.




If you are to test REST API calls in the local PC where the DA Core is installed, you can use "127.0.0.1" as the IP address.

5. Click **Save** to save the changes and click **Close**.



Do not change the values in **Database**, **User name** and **Password** under **DB** unless you are advised to do so. Incorrect settings can cause problems in connecting the database.

To Start/Stop Service Manually

When the installation is completed, the Device Agent Core service should run

automatically in the background. But, in some cases, you need to start or stop the service manually. To start or stop the service manually, click **Start** or **Stop** on the **Control** tab.

2.5. Running Sample Demo

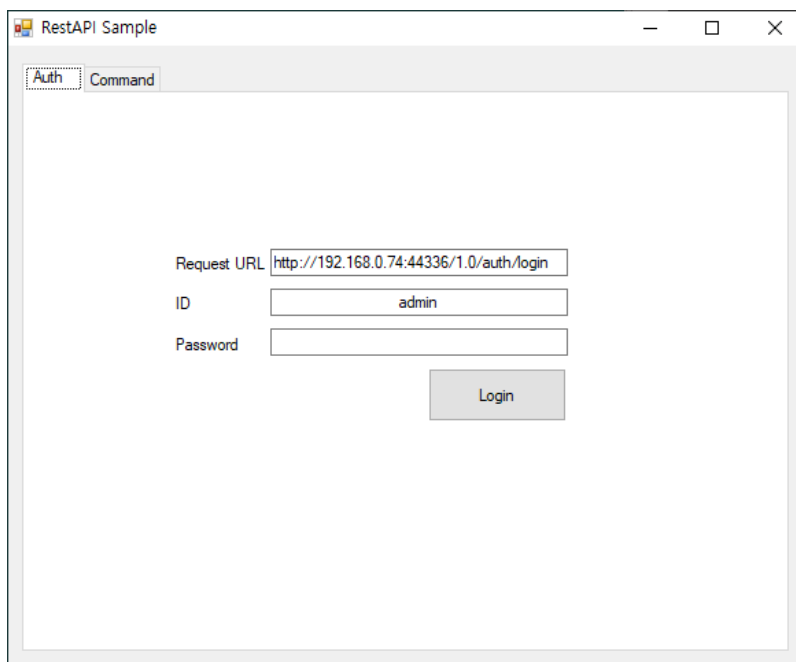
You can download the REST sample demo program and project package [here](https://app.box.com/s/9f1iy0umj8bml4g4m0gs1xwnahstf0a5) [https://app.box.com/s/9f1iy0umj8bml4g4m0gs1xwnahstf0a5].

1. Extract all the data from the .zip file.



- The sample project files can be found in the RestApiSample folder.
- The demo REST client program (RestApiSample.exe) is located in the RestApiSample\bin\Release folder.

2. Double-click **RestApiSample.exe** to run the demo program.



3. Getting Started

This section describes the first steps you need in order to start using our REST API. It aims to guide you to produce the simplest possible output—a successful response from the most basic request—with the API. As a result, you will be able to get an overall sense of how it works and be successful with other extensive API calls.

3.1. Authentication

Before you can make requests with the CMID REST API, you need to take a step for authentication and authorization first in order to access the resources that the API provides. The CMID REST API uses the Basic HTTP Authentication method to allow REST clients to authenticate themselves. It requires you to log in using credentials (user id and password). Upon successful authentication, a new session is established between two entities and you are authorized to consume the API.

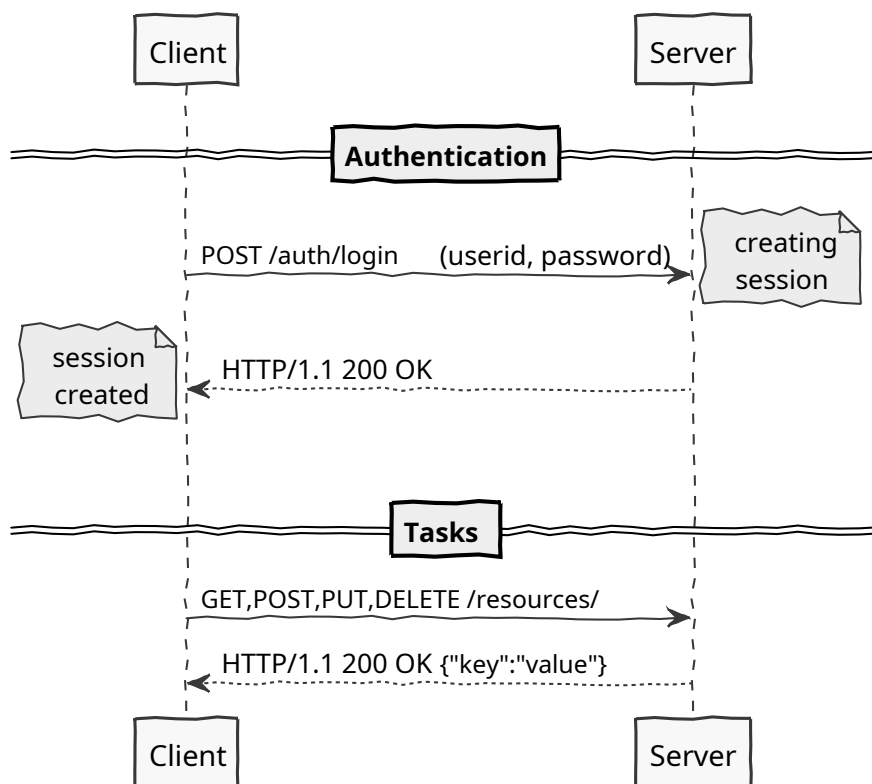


Figure 4. Basic Authentication



- The session timeout is 10 minutes.
- The default service port number is set to **44336**. The port number **44337** is reserved for secure http connection (`https://`) that is to be supported in the future.



Base URI Scheme

- Host: {DA Core's IP address}:44336
- BasePath: /1.0
- Schemes: HTTP

URI Example

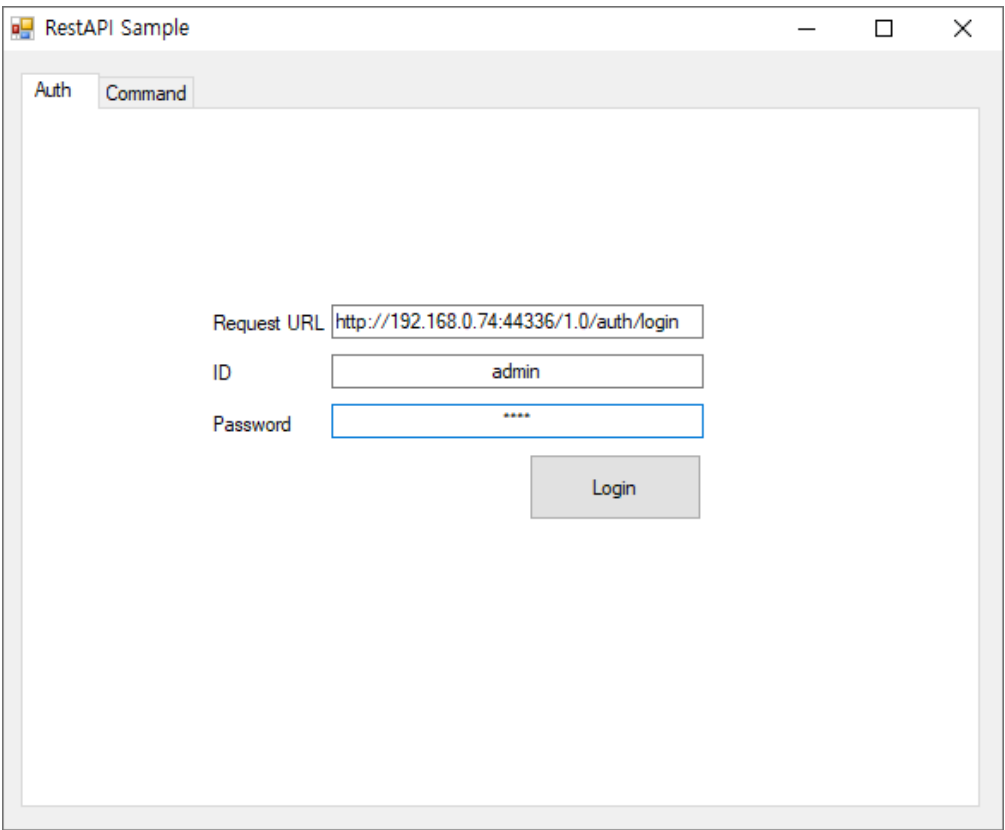
```
http://192.168.0.100:44336/1.0
```

3.1.1. Using Sample Program

This procedure assumes that you use using our REST client demo program (RestApiSample.exe).

1. Type the URL in the **Request URL** box, user id and password, and click **Login**.

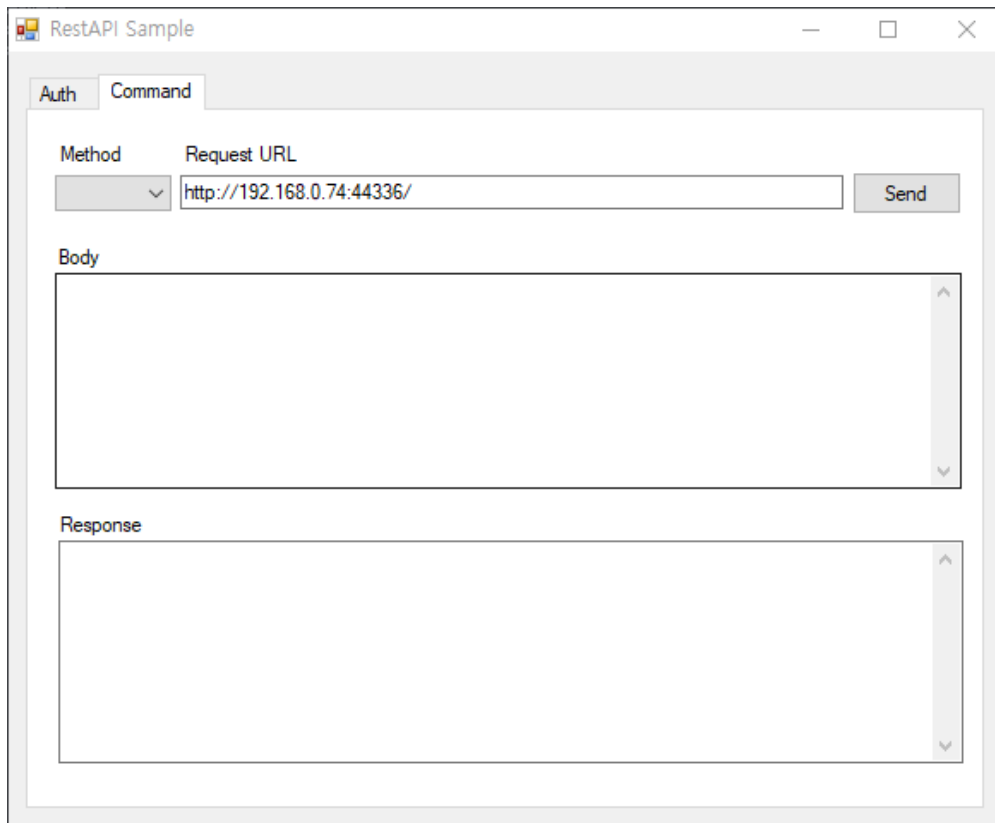
```
http://{host ip address}:44336/1.0/auth/login
```





- The default admin credential is ID: **admin**, Password: **0000**.
- We recommend that you should change the admin password and add admin accounts by installing and using CMID Manager client. For more information, see *2.5. Installing Client* and *3.1.3. Updating Administrator Account* in the CMID Manager V2 admin guide.

2. When the authorization is completed, you are ready to start making requests.



Sample Code Snippet for Authentication

```

AuthData auth = new AuthData();
auth.user_id = txtID.Text;
auth.password = txtPwd.Text;

try
{
    string json = JsonConvert.SerializeObject(auth);
    string retStr = _webRequst.Post(txtAuthURL.Text, json);
    AuthResultData authResult =
    JsonConvert.DeserializeObject<AuthResultData>(retStr);
    if (authResult.admin && authResult.logged)
    {
        lblAuthResult.Text = "Success!";
        pnlCommand.Enabled = true;
    }
    else
    {
        lblAuthResult.Text = "Fail!";
        pnlCommand.Enabled = false;
    }
}
catch { lblAuthResult.Text = "Fail!"; }

```



All the code snippets illustrated in this guide come from *Mainform.cs* in the RestApiSample folder.

3.1.2. Using Other REST Clients

If you want to use other REST testing tools, refer to the following examples in raw format:

Authentication Request Example

```
POST /1.0/auth/login HTTP/1.1
HOST: cmid-server
Content-Type:application/json
```

```
{
  "user_id": "admin",
  "password": "0000",
}
```

Authentication Response Example

```
HTTP/1.1 200 OK
Content-Type:application/json
```

```
{
  "user_id": "admin",
  "admin": true,
  "logged": true,
}
```

3.2. Making First Request

The next step is to send a basic request to the server and receive the response from it. Let us suppose, for instance, that you want to retrieve user information stored in CMID DA Core. The following procedure demonstrates how to achieve your goal.

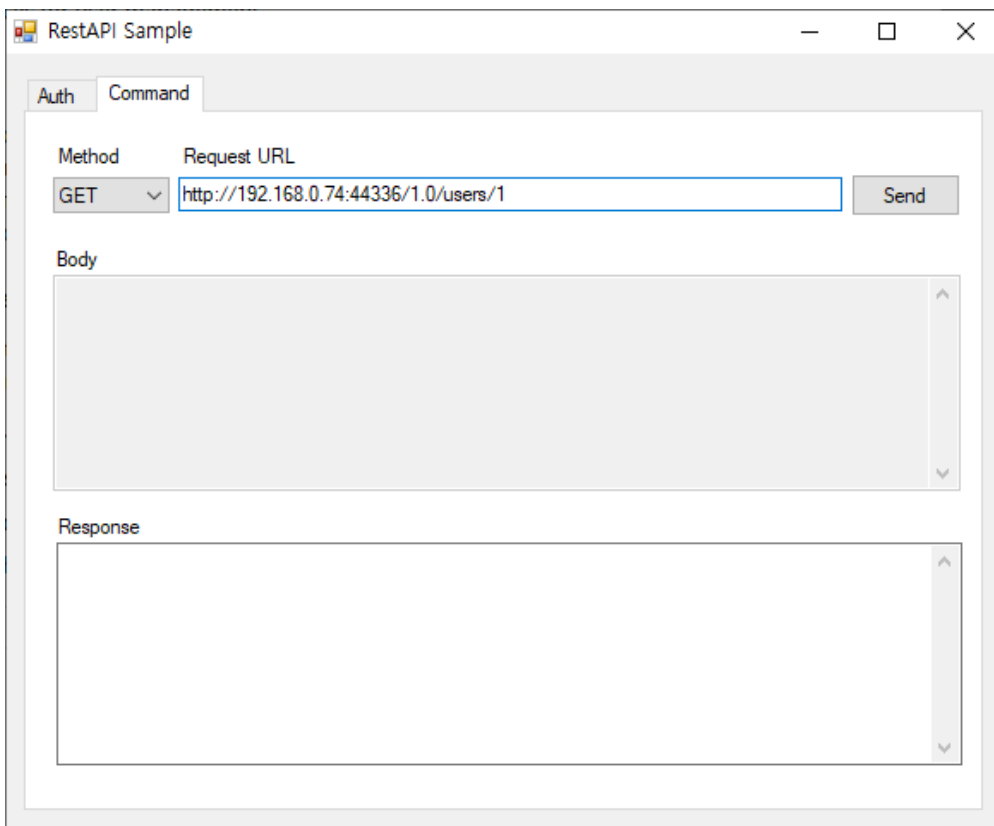
3.2.1. Using Sample Program

The following example request assumes that a user (user id = 1) is registered already in CMID DA Core database using the CMID Manager client. If the user with ID number 1 does not exist, try a different user ID or add a new user using the CMID Manager client (See the **3.3.1. Registering user** section in [the CMID Manager V2 Admin Guide](https://app.box.com/s/8m38x7i6961ikbl4z0swfxt8cgjjg6p6) [https://app.box.com/s/8m38x7i6961ikbl4z0swfxt8cgjjg6p6])

1. Type the following URL in the **Request URL** box and select **GET** in **Method**.

```
http://{host ip address}:44336/1.0/users/1
```

2. Click **Send**.



Request Scheme

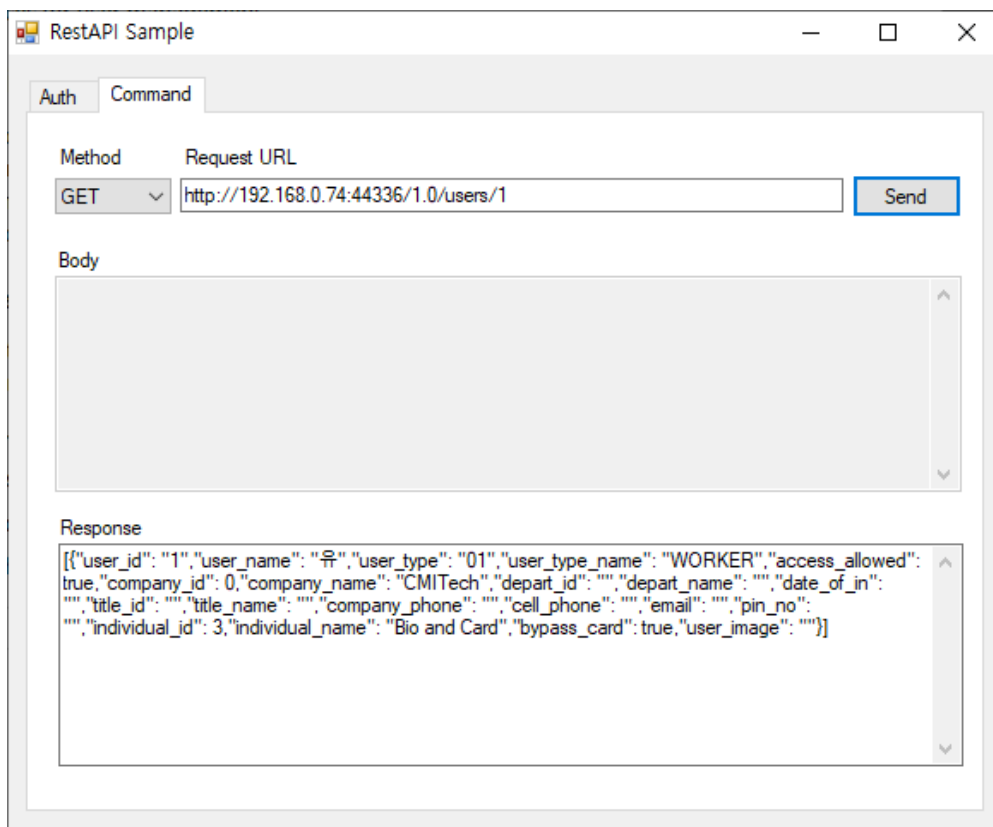


- Method: **GET**
- Path: **/1.0/users,/1.0/users/{user_id}**
- Query parameters (optional):
page={page}&per_page={per_page}&user_id={user_id}&user_name={user_name} (For more information, see [Getting Users Request Query Content](#))

URI Example

```
http://192.168.0.100:44336/1.0/users&page=1&per_page=3&user_id=1&user_name=david
```

3. Evaluate the response.



Sample Code Snippet for GET Method

```
txtResponse.Text = _webRequest.Get(txtURL.Text);
```

3.2.2. Using Other REST Clients

For using other REST testing tools, refer to the following examples in raw format:

Getting User Request Example 1

```
POST /1.0/users/{user_id} HTTP/1.1
HOST: cmid-server
Content-Type:application/json
```

Getting User Request Example 2

```
POST
/1.0/users&page={page}&per_page={per_page}&user_id={user_id}&user_name={user_name}
HTTP/1.1
HOST: cmid-server
Content-Type:application/json
```

Table 1. Getting Users Request Query Content

Name	Description	Type	Scheme	Note
user_id	Specifies the search criteria by a user ID	string	Query or Path	
page	The start page of the user list to read	integer	Query	Default: 1
per_page	The last page of the user list to read	integer	Query	Default: 13
user_name	Specifies the search criteria by a user name	string	Query	

Getting User Response Example

```
HTTP/1.1 200 OK
Content-Type:application/json
```

```
[
  {
    "user_id": "1",
    "user_name": "David",
    "user_type": "01",
    "user_type_name": "WORKER",
    "access_allowed": true,
    "company_id": 0,
    "company_name": "CMITech",
    "depart_id": "",
    "depart_name": "",
    "date_of_in": "",
    "title_id": "",
    "title_name": "",
    "company_phone": "",
    "cell_phone": "",
    "email": "",
    "pin_no": "",
    "individual_id": 3,
    "individual_name": "Bio and Card",
    "bypass_card": true,
    "user_image": ""
  }
]
```

Table 2. Getting User Response Body Content

Name	Description	Type	Notes
user_id	The ID of the user	string	
user_name	The name of the user	string	
user_type	Which type the user belongs to in number	string	
user_type_name	Which type the user belongs to in name	string	
access_allowed	True if the user is allowed to access devices	boolean	
company_id	The ID number of the user's company	integer	
company_name	The name of the user's company	boolean	
depart_id	The ID number of the department that the user belongs to	integer	
depart_name	The department name that the user belongs to	string	
date_of_in	The employment date of the user	string	
<i>Array</i> title_id	The ID number of the user's title	string	
title_name	The title name of the user	string	
company_phone	The phone number of the user's company	string	
cell_phone	The mobile phone number of the user	string	
email	The email address of the user	string	
pin_no	The PIN that the user can present as a credential	string	
individual_id	The ID number of individual authentication type	integer	
individual_name	The name of individual authentication type	string	
bypass_card	True if the user can access using card only	boolean	
user_image	The preview face image of the user	string	Encoded in BASE64

4. Tutorials

This section gives step-by-step instructions for others aspects of using the CMID API after making your first request. It covers a few of common topics or use cases that you are likely to encounter during the implementation. Each topic is intended to describe a use case that exposes how various goals of the API can be combined in order to achieve something.

4.1. Managing Device

The device management is one of the common tasks that need to be performed in an early stage mostly. It may include registering device, reading device, and updating device.

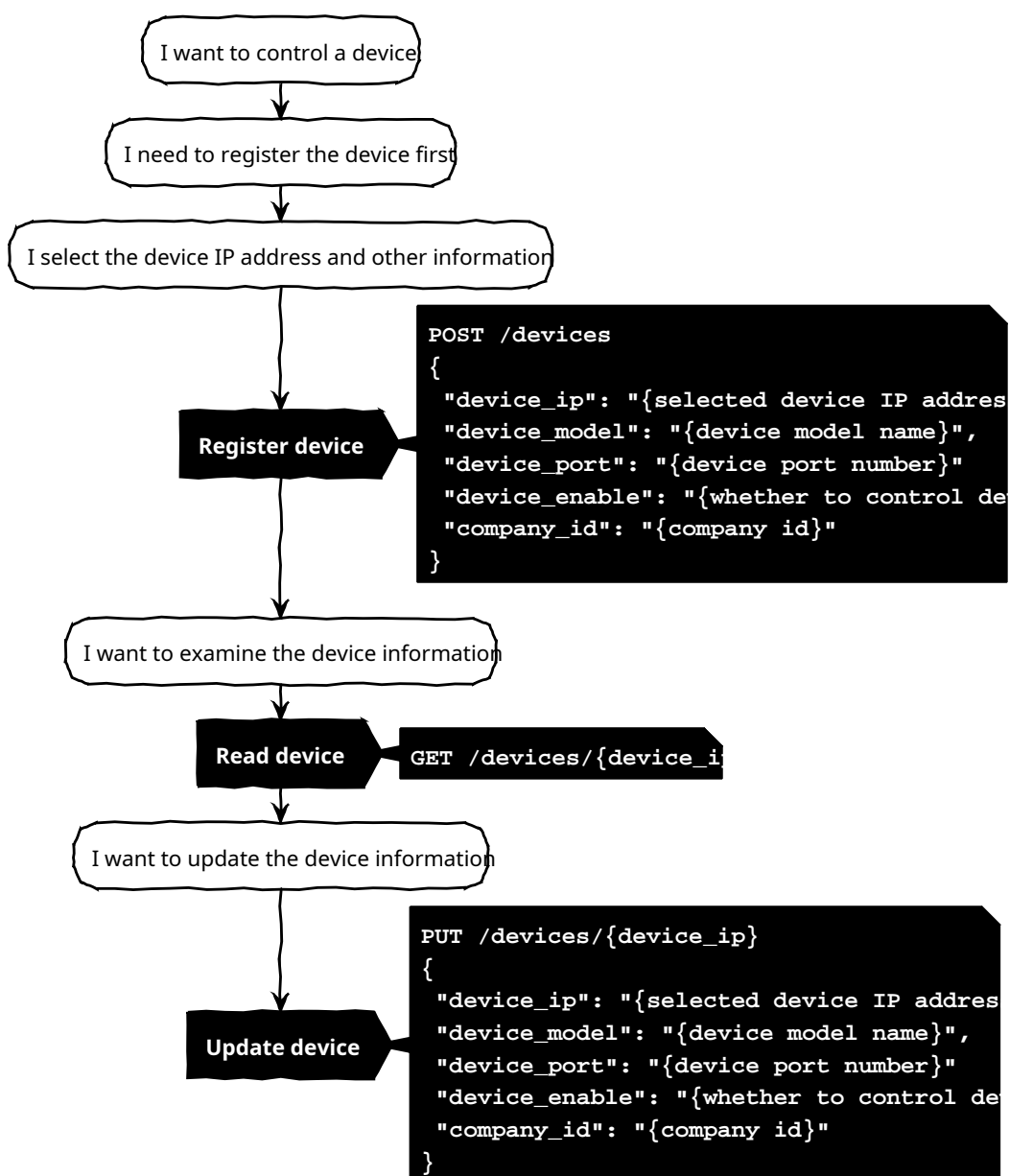


Figure 5. Device Management Workflow Example

4.1.1. Adding Device

Registering a new device to CMID DA Core server is the first step for device management. It leads to connect the device to the server. Once you make a connection between them, you will be able to take a control of the device.



If you make requests to the existing CMID DA Core that has been operating all the while and serves the API calls from other clients, you should check whether the device IP address is registered already in the server by sending a "GET device" (see [Reading Device](#) later in this section) request before trying to register a new device.

1. Type the following URL in the **Request URL** box and select **POST** in **Method**.

```
http://{host ip address}:44336/1.0/devices
```

2. Type the request body content in the **Body** field in JSON format and click **Send**.

Body Example

```
{"device_ip":"192.168.0.100","device_model":"EF-45","device_port":"9980","device_name":"","device_enabled":true,"company_id":0,"door_id":0,"door_type":0}
```

Request Scheme

- Method: **POST**
- Path: **/1.0/devices**
- Body: see [Adding Device Request Body Content](#)



URI Example

```
http://192.168.0.100:44336/1.0/devices
```

3. Evaluate the response.

Sample Code Snippet for POST Method

```
txtResponse.Text = _webRequest.Post(txtURL.Text, txtBody.Text);
```

References for Adding Device

Adding Device Request URI Example

```
POST /1.0/devices HTTP/1.1
HOST: cmid-server
Content-Type: application/json
```

Adding Device Request Body Example

```
{
  "device_ip": "192.168.0.254",
  "device_model": "EF45",
  "device_port": "9980",
  "device_name": "",
  "device_enabled": true,
  "company_id": "0",
  "door_id": "0",
  "door_type": "0"
}
```

Table 3. Adding Device Request Body Content

Name	Description	Type	Required	Notes
device_ip	The IP address of the device	string	Required	
device_model	The device model name	string	Required	For example, EF-45
device_port	The port number of the device	string	Required	Default is 9980
device_name	The device name	string	Optional	
device_enabled	True if the device is enabled	boolean	Required	
company_id	The ID number of the user's company	integer	Required	
door_id	The ID number of the door	integer	Optional	
door_type	The type of the door that the device controls	integer	Optional	Valid values: 0 (entrance), 1 (exit)

Adding Device Response Example

```
HTTP/1.1 200 OK
Content-Type:application/json
```

```
{
  "device_ip": "192.168.0.254",
  "device_id": 19,
  "result": true,
  "result_code": 0
}
```

Table 4. Adding Device Response Body Content

Name	Description	Type	Notes
device_ip	The IP address of the registered device	string	
device_id	The ID number of the device	integer	
result	True if the device registration is successful	boolean	
result_code	The code number of the result	string	

4.1.2. Reading Device

Getting the registered device information enables you to examine whether the data is added correctly. Sometimes, it can be a preliminary step prior to device registration to check whether the IP address that you will enlist is already in use.

1. Type the following URL in the **Request URL** box and select **GET** in **Method**.

```
http://{host ip address}:44336/1.0/devices/{device ip address}
```

2. Click **Send**.



Request Scheme

- Method: **GET**
- Path: **/1.0/devices/{device_ip}**

URI Example

```
http://192.168.0.100:44336/1.0/devices/192.168.0.254
```

3. Evaluate the response.

References for Reading Device

Reading Device Request URI Example

```
POST /1.0/devices/{device_ip} HTTP/1.1
HOST: cmid-server
Content-Type:application/json
```

Reading Device Response Example

```
HTTP/1.1 200 OK
Content-Type:application/json
```

```
{
  "device_ip": "192.168.0.254",
  "device_model": "EF-45",
  "device_port": "9980",
  "device_sn": "",
  "device_name": "",
  "device_enable": true,
  "device_connected": false,
  "door_name": "",
  "door_direction": ""
}
```

Table 5. Reading Device Response Body Content

Name	Description	Type	Notes
device_ip	The IP address of the device	string	
device_model	The device model name	string	
device_port	The port number of the device	integer	
device_sn	The serial number of the device	string	
device_name	The name of the device	string	
device_enable	True if the device is enabled	boolean	
device_connected	True if the device is connected to server	boolean	
door_name	The name of the door that the device controls	string	
door_direction	On which side of the door the device is installed	string	

4.1.3. Updating Device

If you want to add more information or modify the present data, you can use UPDATE method. The instructions and the references are identical to those of [Adding Device](#) except the request method and the URI scheme.



Request Scheme

- Method: **PUT**
- Path: **/1.0/devices/{device_ip}**
- Body: see [Adding Device Request Body Content](#)

URI Example

```
http://192.168.0.100:44336/1.0/devices/192.168.0.254
```

Sample Code Snippet for PUT Method

```
txtResponse.Text = _webRequest.Put(txtURL.Text, txtBody.Text);
```

4.2. Registering User

This tutorial give the information about user registration as one of the common tasks. Registering user may consist of two or three sub tasks (for example, adding user and granting permission to user) depending on your goals.

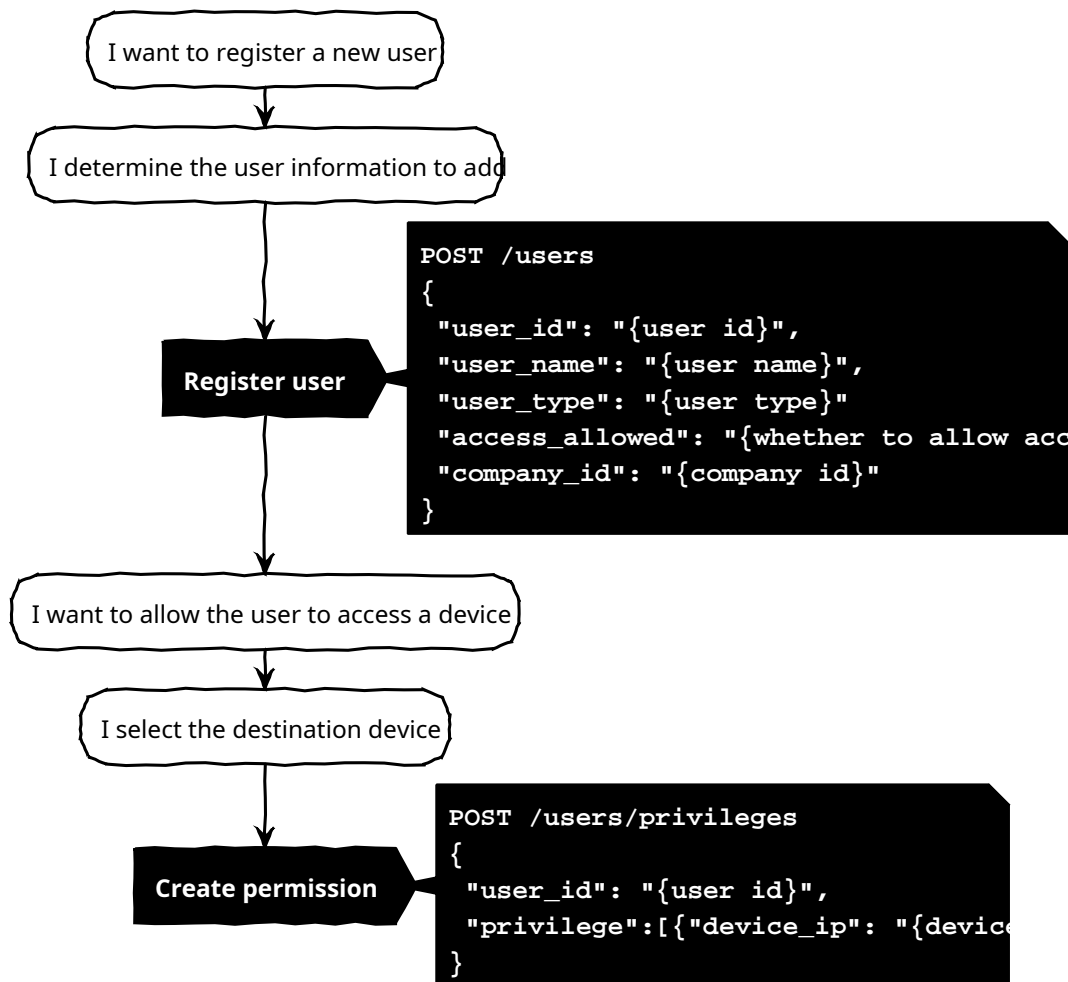


Figure 6. User Registration Workflow Example

4.2.1. Adding User

You can create a user in the CMID DA Core with the user profile including various personal information by making the following request.

1. Type the following URL in the **Request URL** box and select **POST** in **Method**.

```
http://{host ip address}:44336/1.0/users
```

2. Type the request body content in the **Body** field in JSON format and click **Send**.

Body Example

```
{"user_id": "1234", "user_name": "Paul", "user_type": "0", "access_allowed": true, "company_id": 0, "depart_id": "1", "company_phone": "", "cell_phone": "", "date_of_in": "", "date_of_expired": "", "title_id": "", "email": "abc@abc.com", "password": "", "pin_no": "", "individual_id": 0, "bypass_card": false}
```

Request Scheme

- Method: **POST**
- Path: **/1.0/users**
- Body: see [Adding User Request Body Content](#)



URI Example

```
http://192.168.0.100:44336/1.0/users
```

3. Evaluate the response.

References for Adding User

Adding User Request URI Example

```
POST /1.0/users HTTP/1.1
HOST: cmid-server
Content-Type: application/json
```


Adding User Request Body Example

```

{
  "user_id": "1234",
  "user_name": "Paul",
  "user_type": "01",
  "access_allowed": true,
  "company_id": 0,
  "depart_id": "1",
  "company_phone": "",
  "cell_phone": "",
  "date_of_in": "",
  "date_of_expired": "",
  "title_id": "",
  "email": "abc@abc.com",
  "password": "",
  "pin_no": "",
  "individual_id": 0,
  "bypass_card": false
}

```

Table 6. Adding User Request Body Content

Name	Description	Type	Required	Notes
user_id	The ID of the user	string	Required	
user_name	The name of the user	string	Required	
user_type	Which type the user belongs to in number	string	Required	
access_allowed	True if the user is allowed to access devices	boolean	Required	
company_id	The ID number of the user's company	integer	Required	
depart_id	The ID number of the department that the user belongs to	string	Optional	
company_phone	The phone number of the user's company	string	Optional	
cell_phone	The mobile phone number of the user	string	Optional	

Name	Description	Type	Required	Notes
date_of_in	The employment date of the user	string	Optional	
date_of_expired	The date when the user's permission is expired	string	Optional	
title_id	The ID number of the user's title	string	Optional	
email	The email address of the user	string	Optional	
password	The password of the user	string	Optional	
pin_no	The PIN that the user can present as a credential	string	Optional	
individual_id	The ID of individual authentication type	integer	Optional	
bypass_card	True if the user can access using card only	boolean	Optional	

Adding User Response Example

```
HTTP/1.1 200 OK
Content-Type:application/json
```

```
{
  "user_id":"1234",
  "result":true,
}
```

Table 7. Adding User Response Body Content

Name	Description	Type	Notes
user_id	The ID of the registered user	string	
result	True if the user registration is successful	boolean	

4.2.2. Creating User Permission

Adding users does not make them to obtain the privilege for accessing devices that are connected to the server. In order to gain access to devices, you have to give the users the right to access for each device by creating permission.

1. Type the following URL in the **Request URL** box and select **POST** in **Method**.

```
http://{host ip address}:44336/1.0/users/privileges
```

2. Type the user ID and the IP address of destination devices in the **Body** field in JSON format and click **Send**.

Body Example

```
{"user_id":"1234","privileges":[{"device_ip":"192.168.0.200"}]}
```



You can put the multiple IP addresses in the `privileges` JSON array. For example,

```
{"user_id":"1234","privileges":[{"device_ip":"192.168.0.200"}, {"device_ip":"192.168.0.201"}, {"device_ip":"192.168.0.203"}]}
```

Request Scheme

- Method: `POST`
- Path: `/1.0/users/privileges`
- Body: see [Creating Permission Request Body Content](#)



URI Example

```
http://192.168.0.100:44336/1.0/users/privileges
```

3. Evaluate the response.

References for Creating Permission

Creating Permission Request URI Example

```
POST /1.0/users/privileges HTTP/1.1
HOST: cmid-server
Content-Type:application/json
```

Creating Permission Request Body Example

```
{
  "user_id": "1234",
  "privilege": [
    {
      "device_ip": "192.168.0.200"
    },
    {
      "device_ip": "192.168.0.201"
    },
    {
      "device_ip": "192.168.0.202"
    }
  ]
}
```

Table 8. Creating Permission Request Body Content

Name	Description	Type	Required	Notes
user_id	The ID of the user	string	Required	
privileges (array)	device_ip The IP address of the device	string	Required	

Creating Permission Response Example

```
HTTP/1.1 200 OK
Content-Type: application/json
```

```
{
  "user_id": "1234",
  "result": true,
  "result_code": "",
}
```

Table 9. Creating Permission Response Body Content

Name	Description	Type	Notes
user_id	The ID of the user	string	

Name	Description	Type	Notes
result	True if the user permission is written successfully	boolean	
result_code	The code number of the result	string	

4.3. Getting Logs

The event logs contain a variety of useful information that can be consumed in access control and time & attendance management. The time-based information collected from devices let you monitor various access and attendance events and manage the working time of the employees. You can also get the device history such as device restart records or change logs in device settings or parameters. This section covers how to get event logs by making a REST API call.

1. Type the following URL in the **Request URL** box and select **GET** in **Method**.

```
http://{host ip address}:44336/1.0/logs
```

2. Click **Send**.

Request Scheme

- Method: **GET**
- Path: **/1.0/logs**
- Query parameters (optional):
`page={page}&per_page={per_page}&user_id={user_id}&from_date={from_date}&to_date={to_date}&device_id={device_id}&granted_type={granted_type}`

URI Example

```
http://192.168.0.100:44336/1.0/logs
```

3. Evaluate the response.

4.3.1. References for Getting Logs

Getting Logs Request URI Example 1

```
POST /1.0/logs HTTP/1.1
HOST: cmid-server
Content-Type:application/json
```

Getting Logs Request URI Example 2

```
POST
/1.0/logs&page={page}&per_page={per_page}&user_id={user_id}&from_date={from_date}&to_date={to_date}&device_id={device_id}&granted={granted_type} HTTP/1.1
HOST: cmid-server
Content-Type:application/json
```

Table 10. Getting Logs Request Query Content

Name	Description	Type	Scheme	Note
page	The start page of the logs to read	integer	Query	
per_page	The last page of the logs to read	integer	Query	
user_id	Specifies the search criteria by a user ID	string	Query	
from_date	The start date of the logs to read	string	Query	Format: yyyyymmdd
to_date	The last date of the logs to read	string	Query	Format: yyyyymmdd
device_id	The ID number of the device	integer	Query	
granted	Specifies the search criteria by an event result type	string	Query or Path	Valid values: Success, Fail, Allowed, Denied, Denied(Door schedule), Denied(User schedule), Denied(Device schedule)

Getting Logs Response Example

```
HTTP/1.1 200 OK  
Content-Type:application/json
```

```
[  
  {  
    "event_index":1201,  
    "event_date":"2019-07-01",  
    "event_time":"14:08:04",  
    "event_result":"Timeout",  
    "user_id":"","  
    "user_name":"","  
    "door_name":"","  
    "direction":"","  
    "device_ip":"","  
    "device_name":"","  
    "device_sn":"","  
    "auth_type":"","  
    "event_img":""  
  }  
]
```

Table 11. Getting Logs Response Body Content

Name	Description	Type	Notes
event_index	The incremental index number of the log	integer	
event_date	The date when the event happened	string	Format: yyyy-mm-dd
event_time	The time when the event happened	string	Format: hh:mm:ss
event_result	The result message of the event handling	string	
user_id	The ID of the user	string	
user_name	The name of the user	string	
<i>Array</i> door_name	The name of the door that the device controls	string	
direction	On which side of the door the device is installed	string	For example, Entrance, Exit
device_ip	The IP address of the device	string	
device_name	The name of the device	string	
device_sn	The serial number of the device	string	
auth_type	In what type the authentication is made	string	
user_image	The preview face image of the user	string	Encoded in BASE64