



Verizon Critical Asset Sensor Create AIC Apps User Guide

[Home](#) » [Verizon](#) » Verizon Critical Asset Sensor Create AIC Apps User Guide 



Contents

- [1 Critical Asset Sensor Create AIC Apps](#)
- [2 Introduction](#)
- [3 Create Azure IOT Central \(AIC\) Apps using the ARM template API](#)
- [4 Detailed Steps](#)
- [5 Differences between the AIC App directly from the Portal](#)
- [6 References](#)
- [7 Appendix](#)
- [8 Documents / Resources](#)
 - [8.1 References](#)

Critical Asset Sensor Create AIC Apps

Important—Please Read

Verizon Confidential & Proprietary.

© 2022 Verizon. All rights reserved.

Restricted and Controlled Distribution. Not to be used, copied, reproduced in whole or in part, nor its contents revealed in any manner to others without the express written permission of Verizon.

All information herein is subject to change without notice. The information provided was considered accurate at the time the document(s) were developed, and Verizon disclaims and makes no guarantee or warranty, express or implied, as to the accuracy or completeness of any information contained or referenced herein.

VERIZON DISCLAIMS ANY IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE.

Verizon does not guarantee or warrant the availability of the network nor the compatibility of a network with any device, service or product. Verizon disclaims liability for any damages or losses of any nature whatsoever whether direct, indirect, special or consequential resulting from the use of or reliance on any information contained or

referenced herein.

Technical data contained in this document may be subject to U.S. and international export, re-export, or transfer ("export") laws. Diversion contrary to U.S. and international law is strictly prohibited.

Verizon and Verizon logos are trademarks of Verizon. Other product and brand names may be trademarks or registered trademarks of their respective owners.

Introduction

This document is prepared specially for creating Azure IOT Central (AIC) Apps for using Verizon's Asset Tracker Devices using an Azure Resource Manager (ARM) Template. The usual definition of ARM template is scoped here with the Azure Subscription as a resource within the Azure Resource Group in Azure IOT Central, and a Verizon Subscription for Things pace services for the Verizon Asset Tracker devices. Using the ARM Template APIs, effectively creates a custom Azure IOT Central App meant to allow the customer to quickly get started with Verizon Things pace Cloud Connector especially designed to connect IOT devices like, VZ CAT/GAT devices for Azure IOT Central as a target.

Azure IOT Central (AIC) Application

You have several ways to create an Azure IoT Central Application.

You can use one of the GUI-based methods if you prefer a manual approach, or one of the CLI or programmatic methods if you want to automate the process, or a choice of built-in sample templates, including using Custom templates. In each of these, you will not be able to get the VZ CAT/GAT device templates, and the associated Rules, Device Groups, and Default Appl Dashboard, but when you create the AIC App using the VZ provided ARM Template API, you will be able to get all of these. In this context, we will be using ARM Template APIs to create an Azure IOT Central Application.

Whichever approach you choose, the Application configuration options are the same, and the process typically takes a few minutes to complete.

Azure AD App:

In this example/context, we will be creating an Azure Active Directory App, and registering it on Azure IOT Central Platform, and then use that App in turn to create the necessary Azure IOT Central (AIC) App and use the ARM Template API to work with VZ Asset Tracker devices. Please note that AD App plays a critical role in creating the Application ID (also known as Client ID), and then also creating the Directory ID (also known as Tenant ID). Please note that these 2 IDs (in addition to some other IDs that we will be creating) form some of the key elements of the ARM Template API.

AD App Roles:

The App roles are custom roles to assign permissions to Users/Groups and Applications. The Application defines and publishes the app roles and interprets them as Permissions during Authorization. And here in our context, in particular, the App roles for Applications, assign App roles with "Applications" allowed member types in API permission blade.

AD App Credentialing:

Now the next step is creating the appropriate credentialing (either Certificates or Secrets) for the AD app. Credentials can enable confidential applications, like in our case AD app, to identify themselves to the authentication service when receiving Tokens at a web addressable location (using an HTTP scheme). For tighter security and a higher level of assurance, it is recommended to use a Certificate (instead of a Client Secret) as a Credential. A Certificate is more secure than Client Secret. But for simplicity, in our context/example, we are showing and creating a Client Secret (as it is both cheaper and faster). From a Client Secret point of view, a Client Secret string that the AD App uses to prove its identity when requesting a Token, is also referred to as Application Password.

PS: one thing to note, when you create a new Client Secret, you are shown the details of the Client Secret, along with the description, its expiration date, and the most important part – the Value (typically a long 37 digits, alpha-numeric + special characters, and this is visible only at the time of creation, and is required to capture this value to be used in our ARM Template APIs, as the Application password.

Permissions and Consent:

Now that we have the appropriate AD App credentialing done, we need to configure the API permissions on this. Applications are authorized to call APIs, when they are granted permissions by users/admins as part of the consent process. The list of configured permissions should include all the permissions the application needs. In our context/example, when we create/try to add API permissions/request API permissions, we get a host of APIs to select from. We need to select the group "APIs my organization uses", because these are some of the Apps in our AD that expose APIs, and we need to add the Azure IOT Central Application (accidentally? named as Microsoft IOT Central), as that is what is our end goal here.

Now that we select the specific type of API permissions for our Azure IOT Central App, we need to ask the question – What type of permissions does my AD App require? And the answer to that is “Delegated permissions”, and what this means is that our application needs to access the APIs as the signed-in user. The specific permissions are for both “user-impersonation” and “application read Write all”. The User-impersonation permission allows access to Azure IOT Central REST APIs as Signed-In User, where Admin consent is not required. And the Application permission is for Application to do all Read and Write functionality, but with Admin consent required.

There are more details on the concept of the Authorization model, including scopes, permissions and consent at the following referenced document (https://docs.microsoft.com/en-us/azure/active-directory/develop/v2-permissions-andconsent?WT.mc_id=Portal-Microsoft_AAD_RegisteredApps)

Create Azure IOT Central (AIC) Apps using the ARM template API

In our example, we will be creating an Azure Active Directory App and registering it on Azure IOT Central Platform, and then use that in turn to create the necessary Azure IOT Central App using the ARM Template API to work with VZ Asset Tracker devices.

Azure Resource Manager is the deployment and management service for Azure. It provides a management layer that enables you to create, update, and delete resources in your Azure account.

When you create an Azure IoT Central (AIC) application, you have a choice of built-in sample AIC templates. You can also create your own application/ARM templates from existing IoT Central applications. You can then use your own application/ARM templates when you create new AIC Applications.

When you create an AIC Application using ARM template APIs, it includes the following items from your existing application:

- The default application dashboard, including the dashboard layout and all the tiles you’ve defined.
- Device templates, including measurements, settings, properties, commands, and dashboard.
- Rules. All rule definitions are included. However actions, except for email actions, aren’t included.
- Device groups, including their queries.

NOTE: When you create an AIC Application using the ARM template, it doesn’t include the following items:

- Devices
- Users
- Continuous data export definitions

The user needs to add these items manually to any applications created from an application/ARM template APIs.

Prerequisites

The following things are prerequisites for our work in context.

- An active Azure subscription. If you don’t have an Azure subscription, create a free account before you begin.
- You should have at least Contributor access in your Azure subscription. If you created the subscription yourself, you’re automatically an administrator with sufficient access.
- Log in to the Azure Portal [<https://portal.azure.com/#home>]

Some additional concepts and definitions

In this section, some additional concepts are described and definitions are added, that might be needed to understand the overall process.

What is a resource group?

In Azure’s terminology, a resource group is a container that holds related resources for an Azure solution. The resource group can include all the resources for the solution, or only those resources that you want to manage as a group. You decide how you want to allocate resources to resource groups based on what makes the most sense for your organization. Generally, add resources that share the same lifecycle to the same resource group so you can easily deploy, update, and delete them as a group.

The resource group stores metadata about the resources. Therefore, when you specify a location for the resource

group, you are specifying where that metadata is stored. For compliance reasons, you may need to ensure that your data is stored in a particular region.

Detailed Steps

Create an Azure Active Directory Application

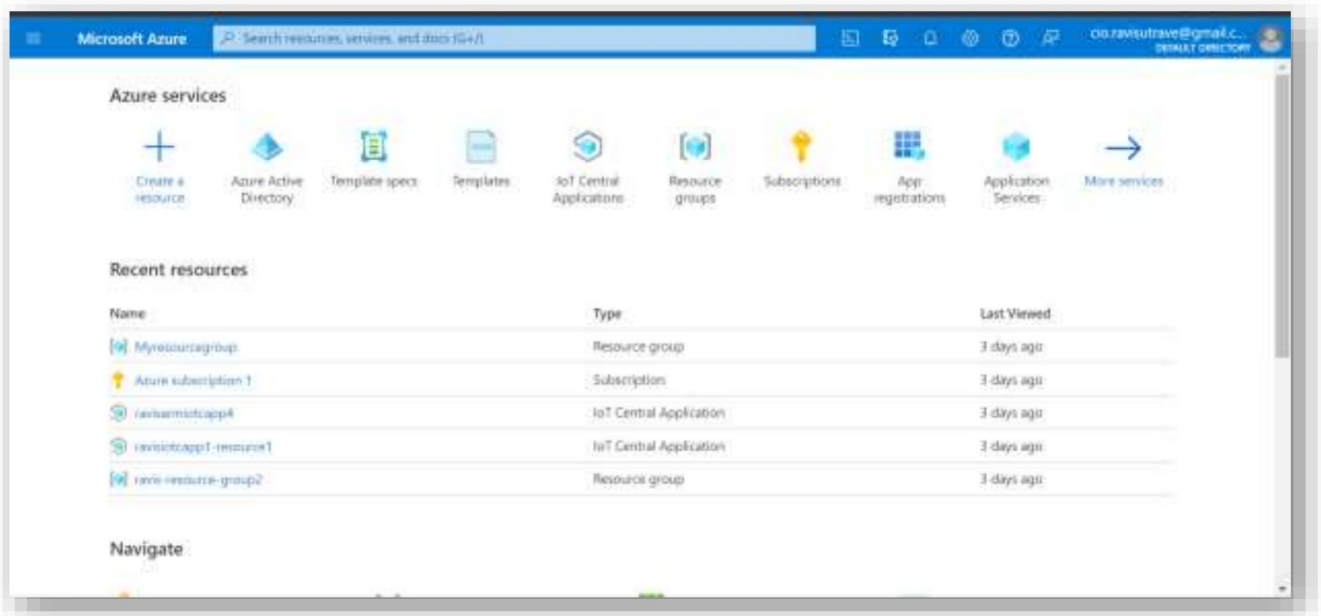
Navigate to the Microsoft Azure portal site. [<https://portal.azure.com/#home>]. Then sign in with the (personal/free-tier) account associated with your Azure subscription.

NOTE: If you are in an enterprise environment, using the Incognito mode for all the browser activities is preferred so that there are less constraints from the corporate firewall settings.

NOTE: The detailed reasoning why we start with an Active Directory App is mentioned in the Appendix

1. Register the Azure Active Directory App to the Azure IOT Central Platform

Log in to the Azure Portal:

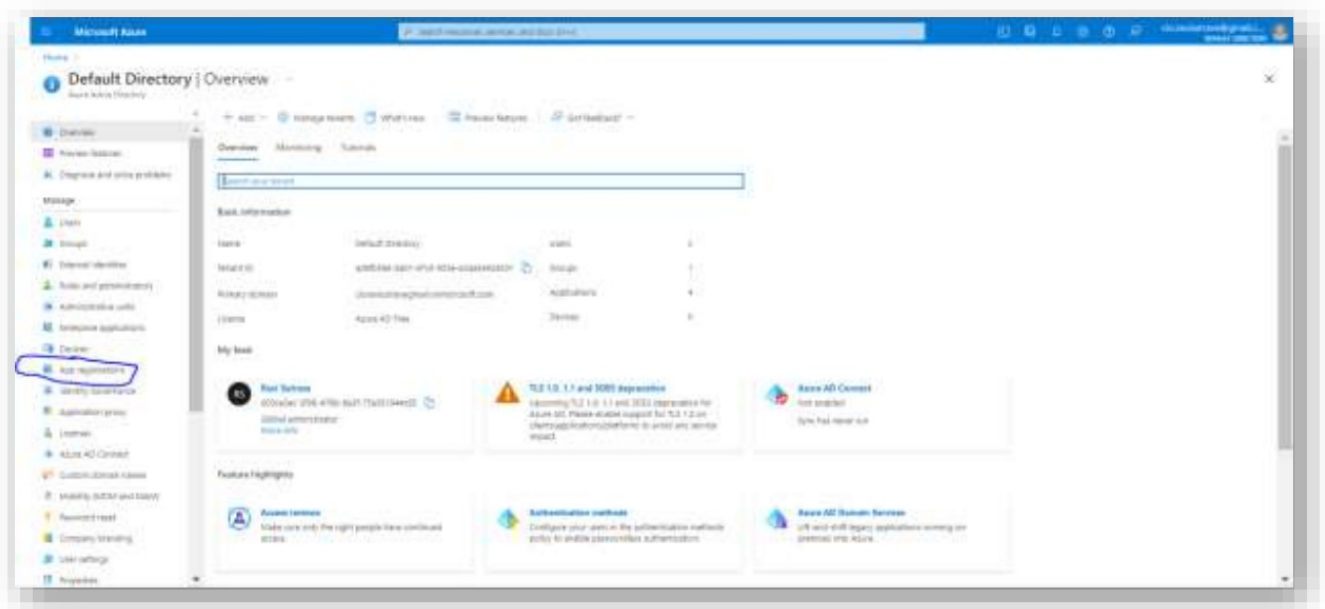


Search for the Azure Active Directory on the search bar,

Once you select Azure AD link/button, create an AD App.

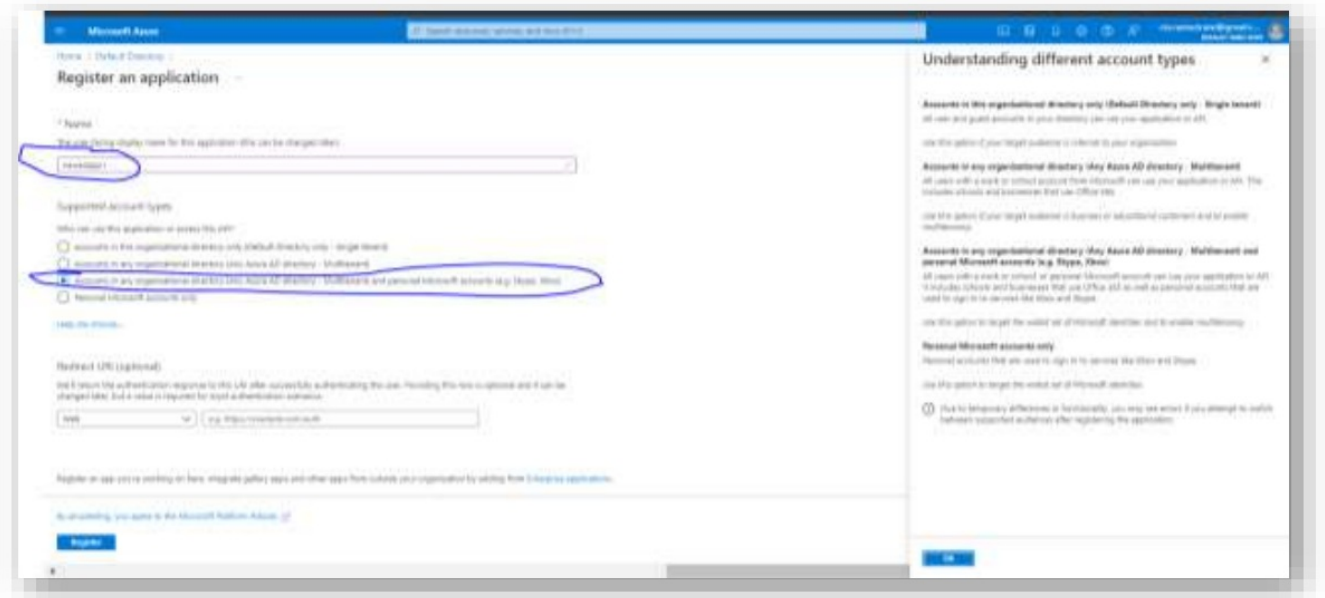
In terms of creating an AD App and then registering it in Azure IOT Central Platform Services in your environment, Microsoft chooses to achieve it from a single command, – App Registration.

Choose the “App Registration” button on the left tab or on +Add – pulldown menu. Next click on “New Registration” on the top with the plus icon:



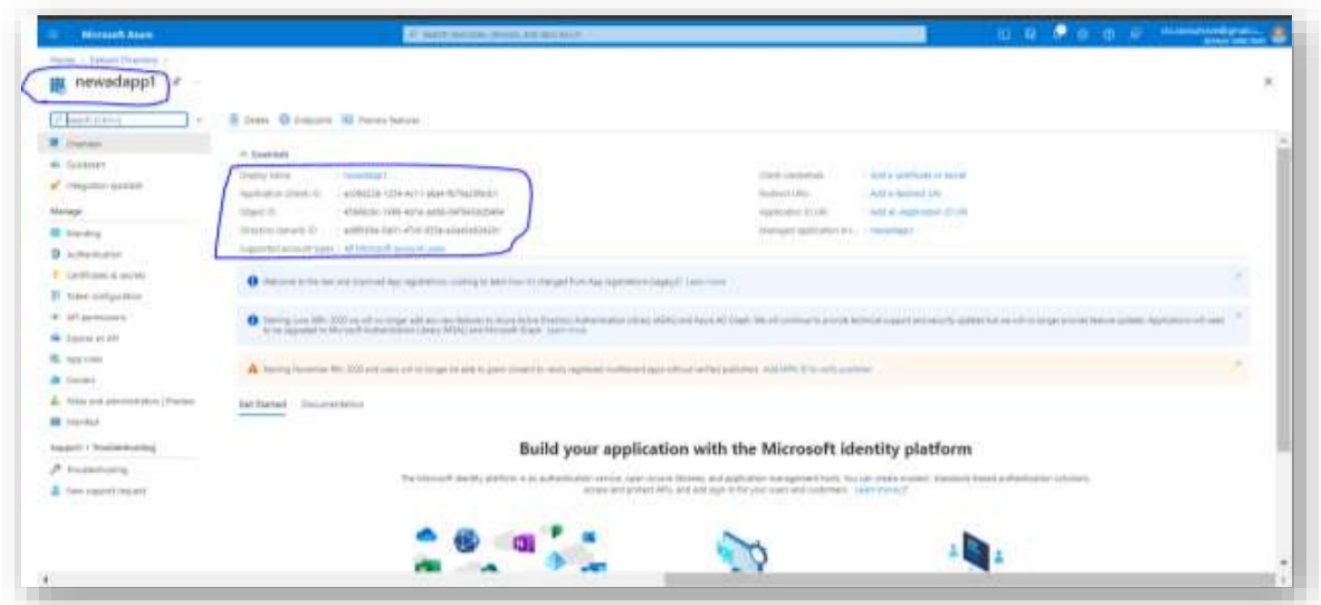
2. Name your AD App and complete the registration

Create a name for your AD App and choose the option “Accounts in any organizational directory (Any Azure AD Directory – Multitenant) and Persona MSFT Account” as shown below, click on “Register”:



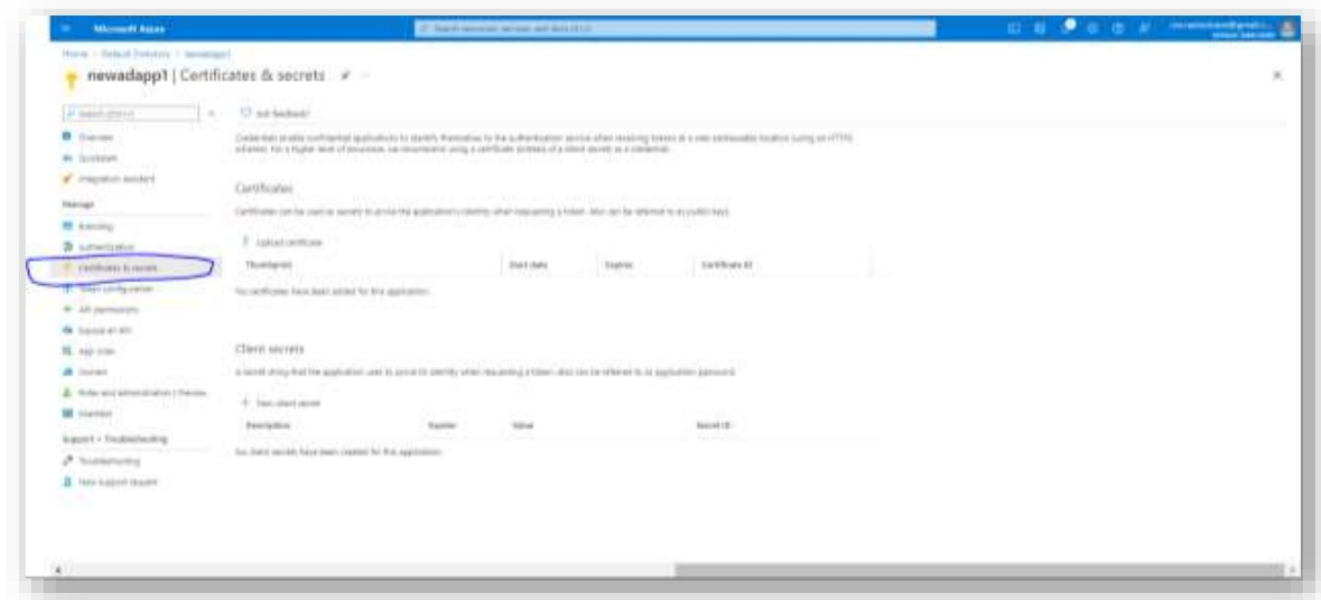
After this step, you should see the newly created AD App.

Inside the newly created app, we can see some essential IDs. Please save the Application client ID. This will be passed from the Verizon's Things Space Cloud Connector for Azure API as the header with the key client ID:



3. Create Secrets

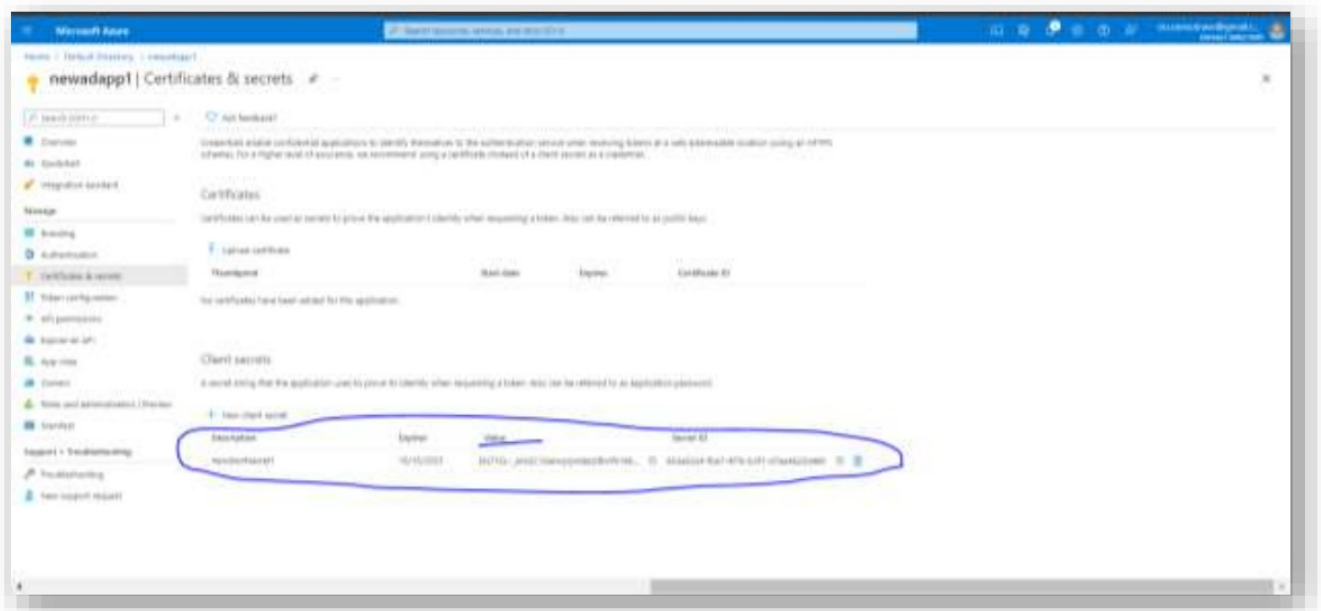
There should be a button that reads “certificates and secrets” under the newly created AD App. Click on that:



As a next step, the client secret has to be generated. This is a one-time process so please store the secret once created. This Secret gets hidden after this step.

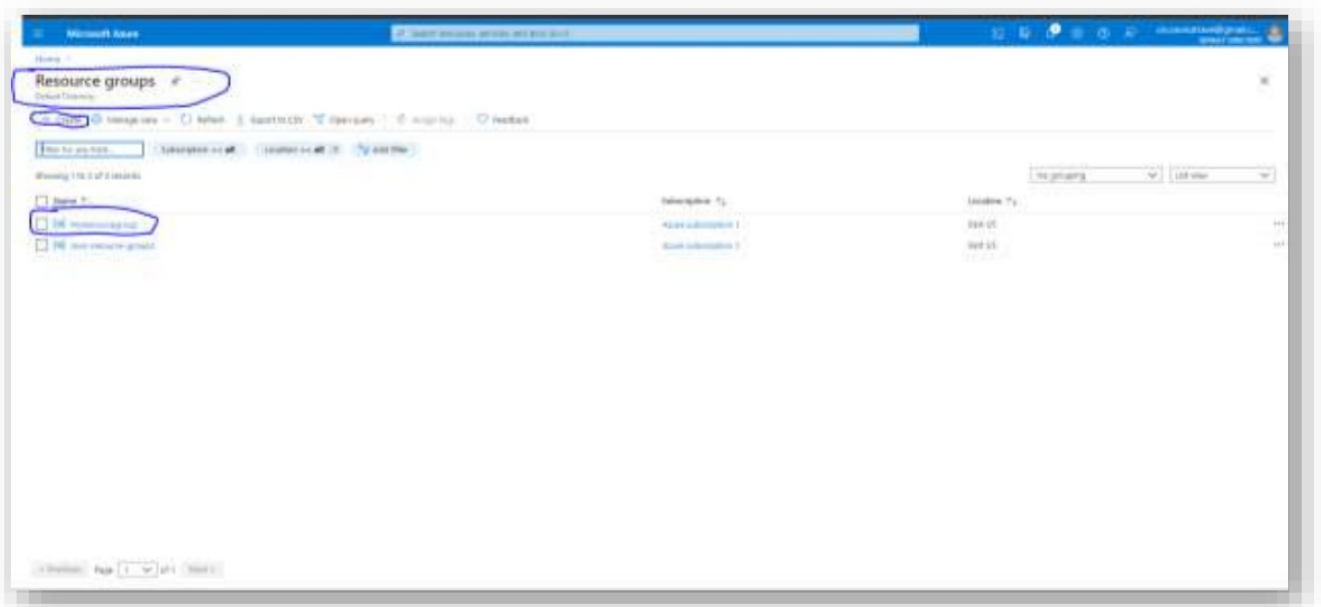
NOTE: In this example/context, the Secret Value shown/visible was: EN77Q~_ANGC1GeHqVjH08zGfbVfR199mHJm1w

Click on the “+ new secret” button and create the client secret. This client Secret will be sent by Verizon’s Thing Space Cloud Connector for Azure API as part of the header with the key “client Secret”:

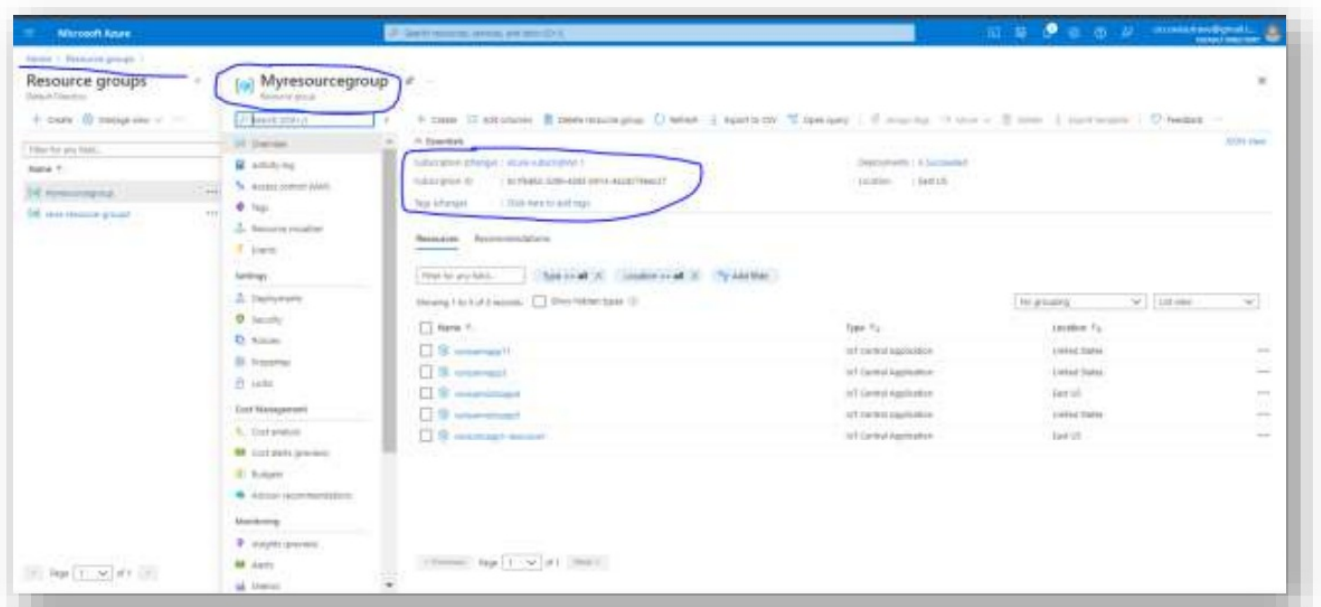


4. Create a Resource Group

One the search bar, search for the resource group. Create a resource group of your choice. Use that for the request parameter “**resourcegroup**”:



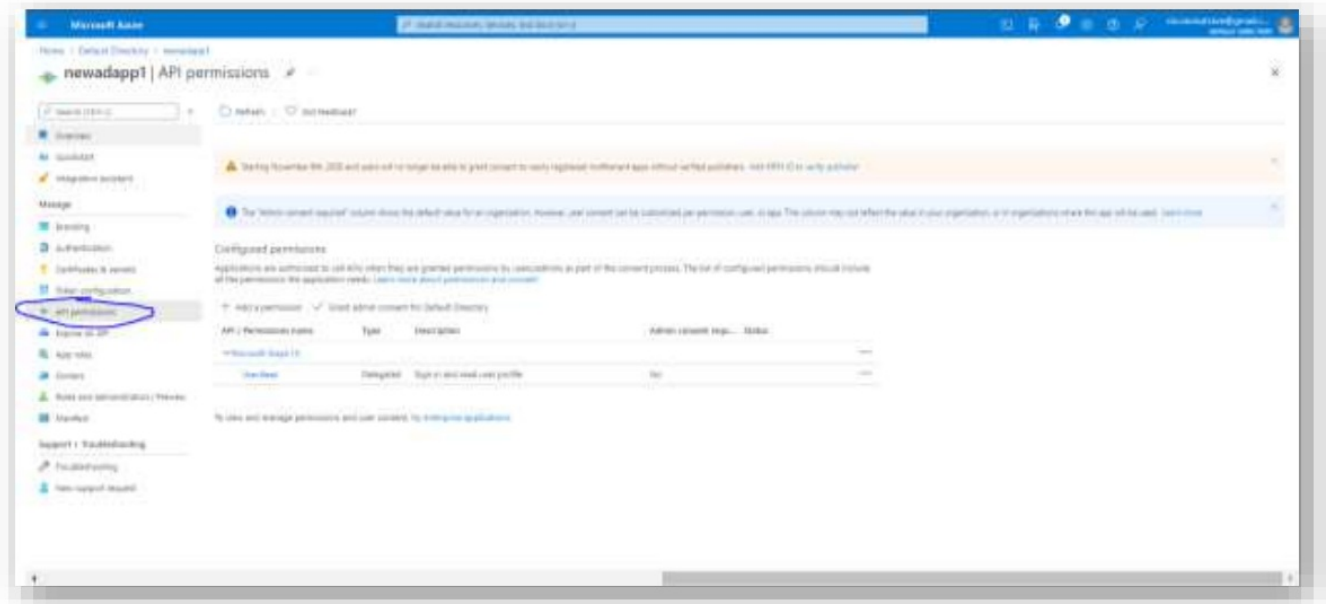
5. Add Azure Subscription as a Resource



6. Add API permissions for AD App

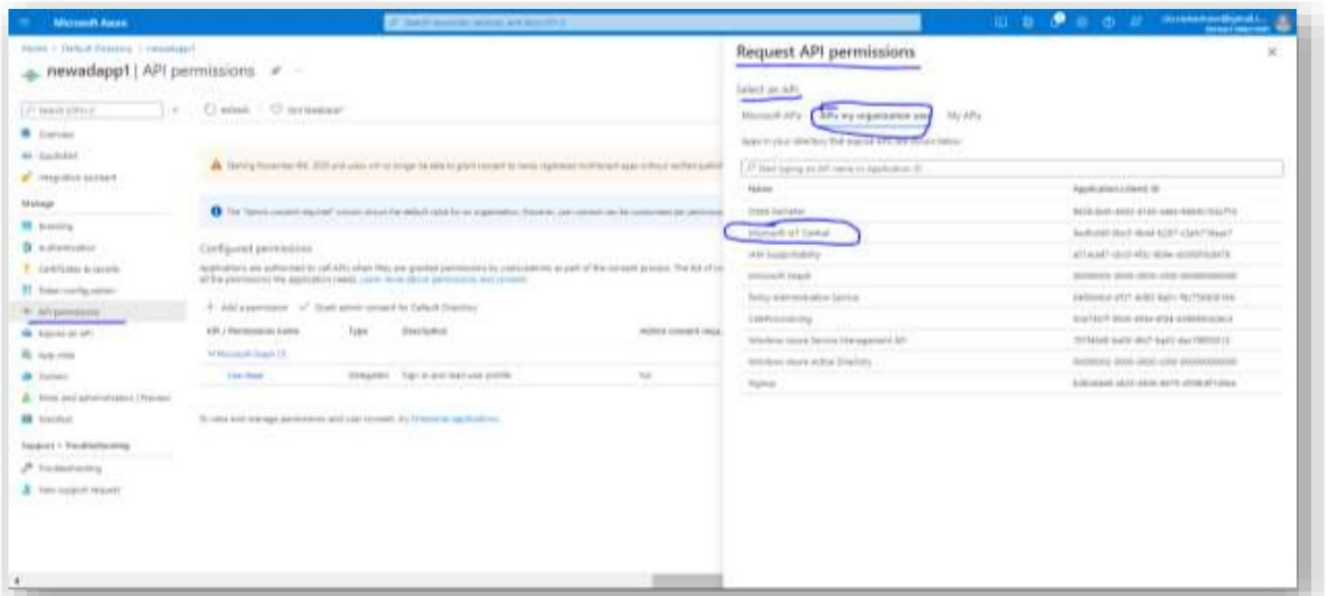
At this step, we would need to give write permission for the AD App to write to the Azure IoT Central App. (listed as Microsoft IOT Central). Click on the API permission tab on the left.

Go to the newly created Azure IoT Central (AIC) AD App (newadapp1) for “API permissions” on the left tab:

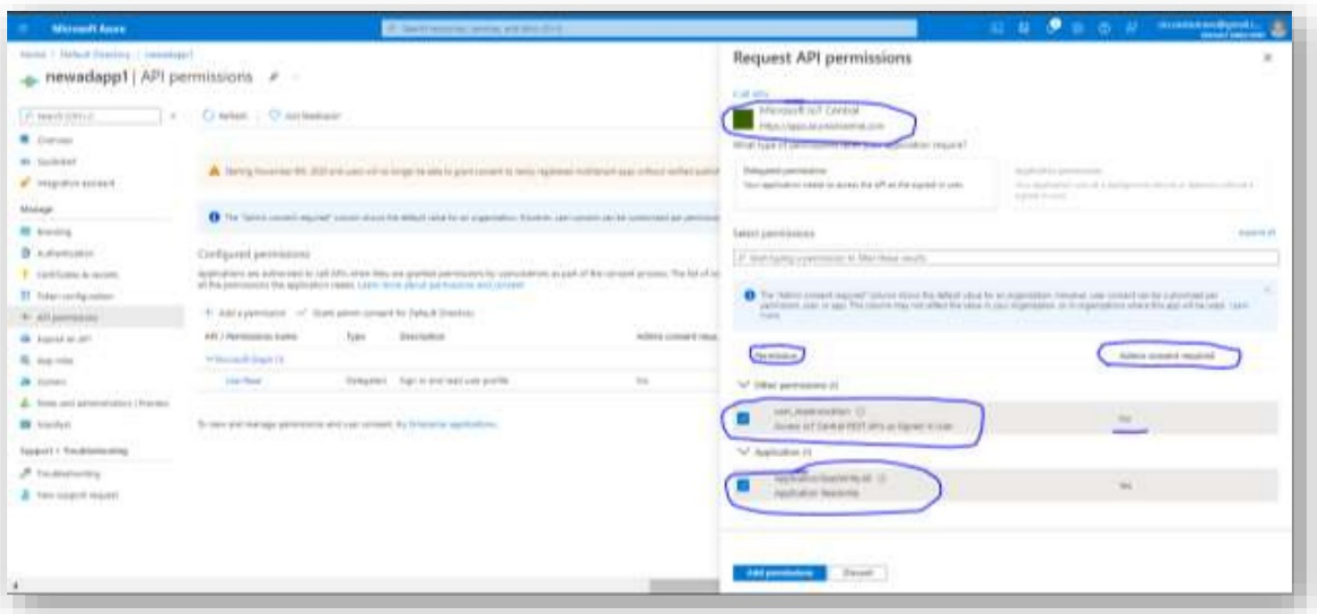


Now choose the “+ add a permission” icon. This will open a dialog where we need to choose the “API’s my organization uses” tab.

Under the search box, please search for the “Microsoft IoT Central” string as shown below:

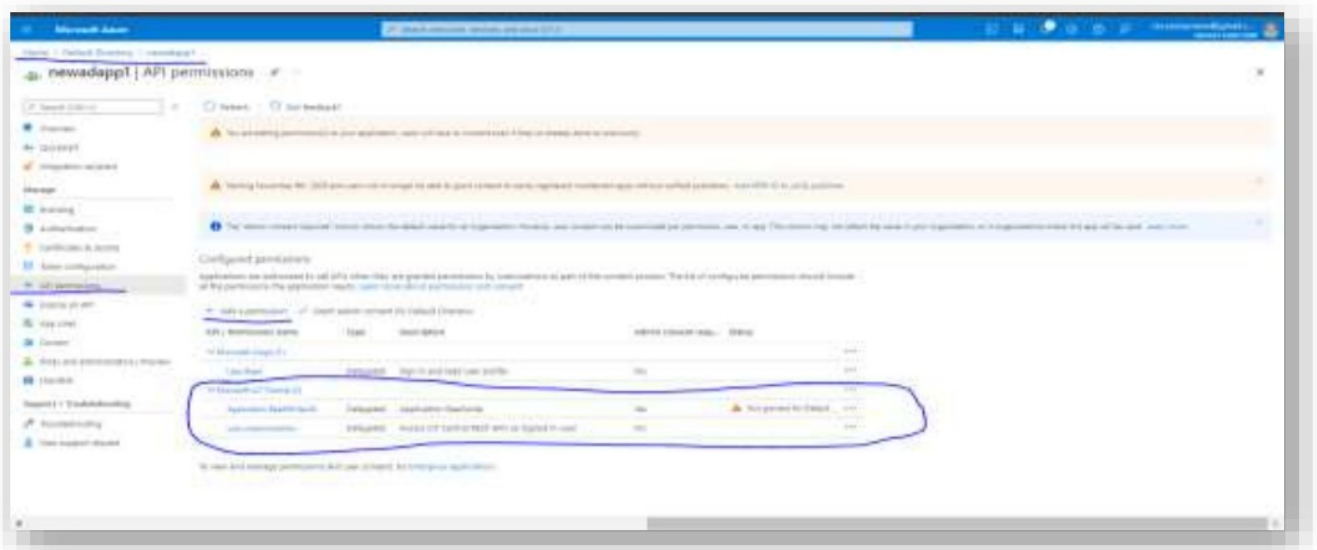


Once “Microsoft IoT Central” is selected click on both the radio button boxes to give the AD App write permission:



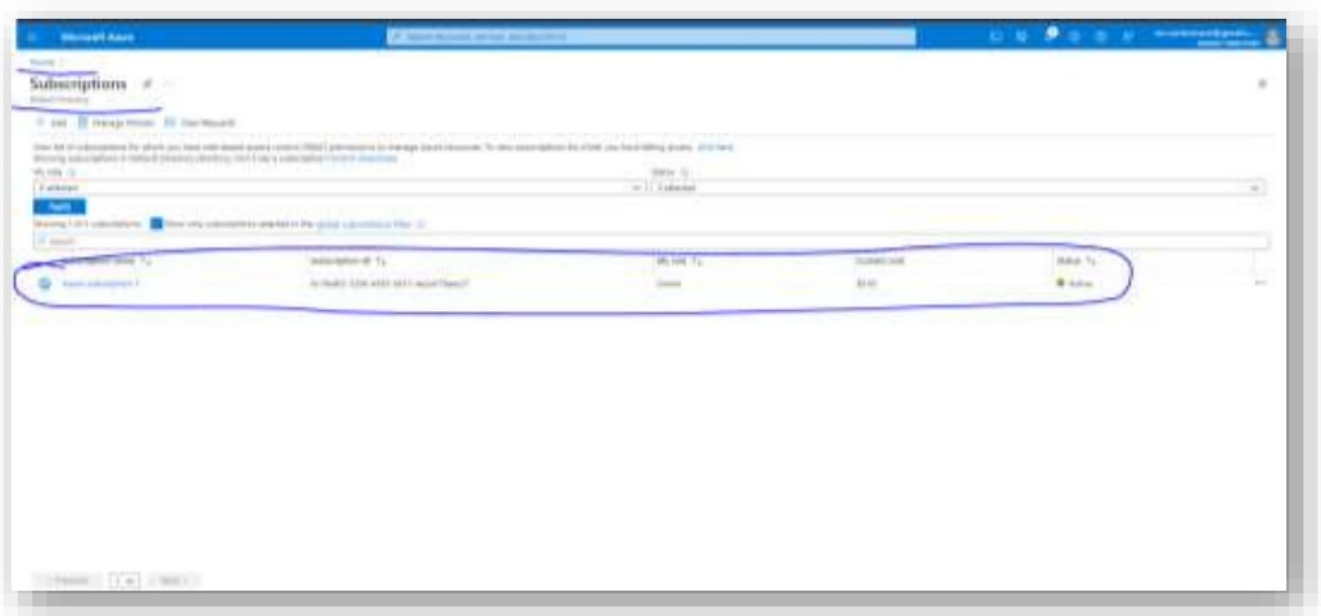
NOTE: If the user does not see “Microsoft IoT Central” check for 2 things:

- Make sure your account has a resource group, and a resource. Otherwise, create one and store the name to send as part of the request.
- Go to the home page and search for IoT central. This would activate IoT central on the API permissions page. It may ask to enter credit card details at this stage.

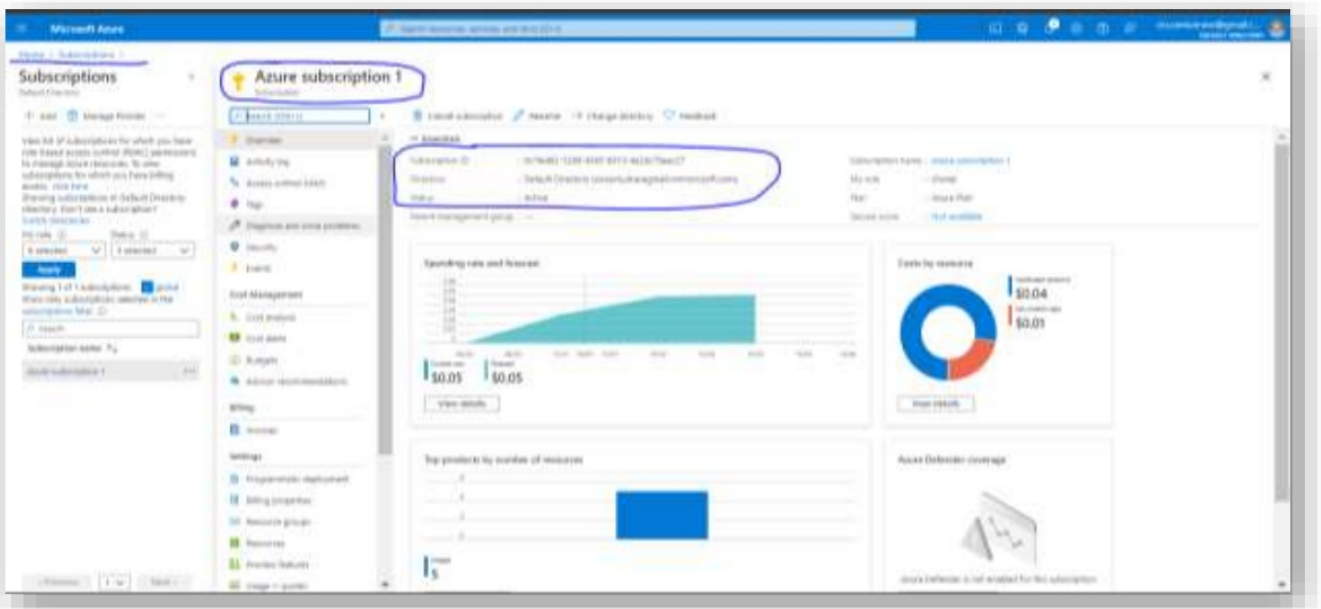


7. Add Role assignment via Access Control (IAM) to the newly created AD app

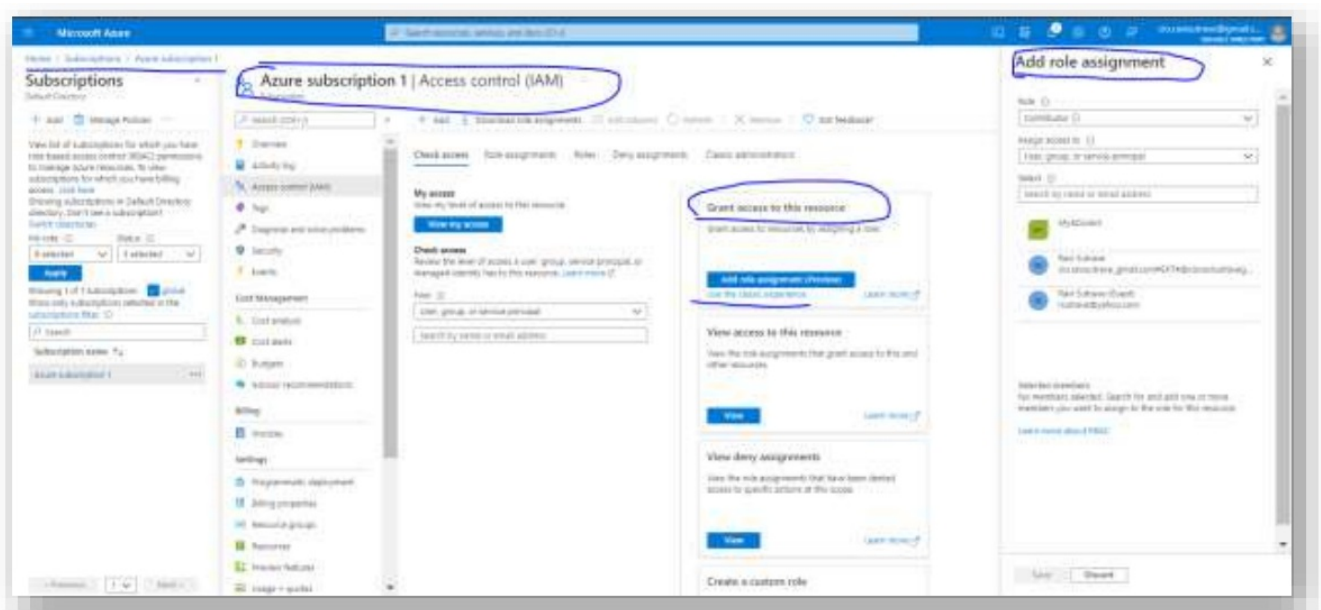
Add Subscriptions via Access control (IAM) to give permissions (role assignment) to the newly created AD app
Go to the Azure portal home page and search for the Subscription:



Select the subscription as follows:

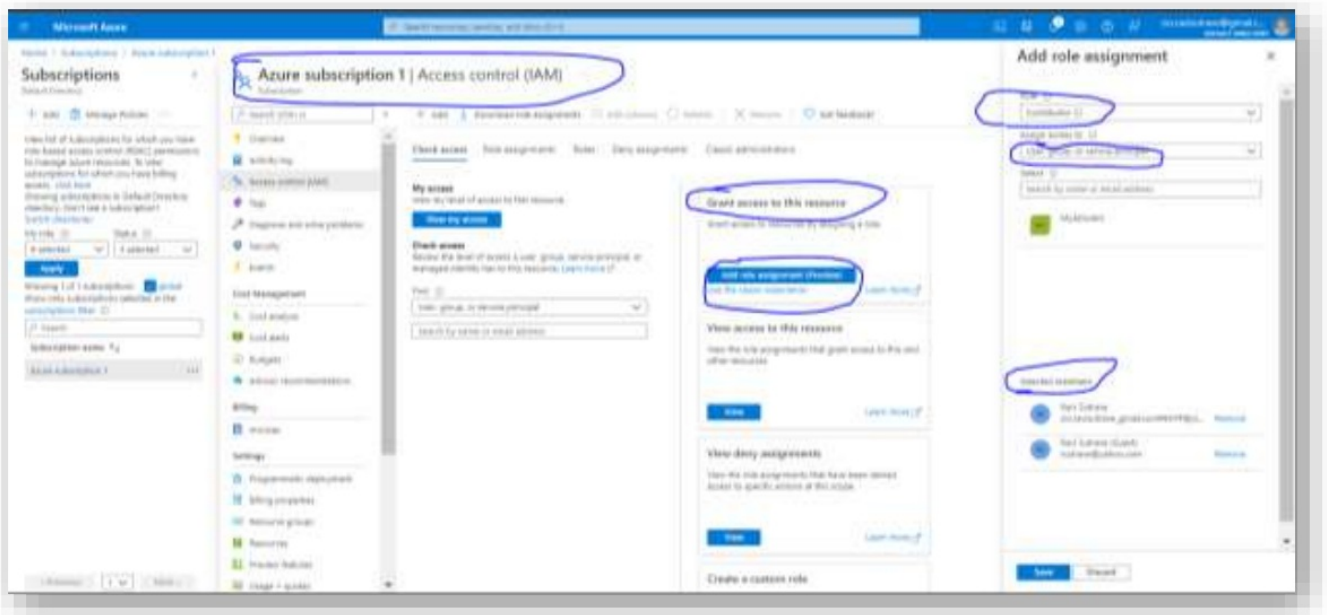


Click on the Access control (IAM) on the left tab and choose “add role assignment: as indicated below. Choose “Use the classic experience”:



Choose the “Role” as “Contributor” and select the newly created AD App. Save this option.

NOTE: You need to add both the “AD App” (newadapp1) which you have created earlier in the process and the Users to the API permissions here:



8. Creation of Azure IOT Central App using ARM Template API

The key headers from the above steps that are required to build the ARM Template API for creation of the App on Azure IOT

Central Platform are:

1. resourcegroup
2. clientSecret
3. clientId
4. subscriptionID
5. tenantID

Collect these required headers to be used in the ARM Template API. With these details create a Postman request: <https://thingspace.verizon.com/api/cc/v1/targets/actions/newaiaic>

The body of the HTTP POST request, showcasing all the key elements:

```
{
  "appName": "newravisarmapp1",
  "billingAccountID": "0342301460-00001",
  "clientId": "ec08d228-1254-4c11-a8a4-fb79a20fadc1",
  "clientSecret": "EN77Q~_ANGC1GeHqVjH08zGfbVfR199mHJm1w",
  "emailIDs": "pon.arasu@one.verizon.com,chandrababu.kamani@one.verizon.com,ravinder.sutrave@one.verizon.com",
  "resourcegroup": "Myresourcegroup",
  "sampleIOTcApp": "5aa2bd25-801c-4413-8031-1fb1842f3f85",
  "subscriptionID": "0c1fed62-3296-4365-b913-4a2dc79eec27",
  "tenantID": "ad6fb56e-5a01-47c0-955e-e2aa5e924201"
}
```

NOTE: One of the parameter/values i.e., the “email IDs” to be added to/sent to with this API/POST Request (as shown above), should be the user’s email IDs who is creating the Azure IOT Central (AIC) App, and other users email IDs additionally, if any can also be added, if they are going to use this App. Please make sure that these email IDs have the Azure Subscription services “ON”, and be role-assigned as “contributor” using the IAM procedures.

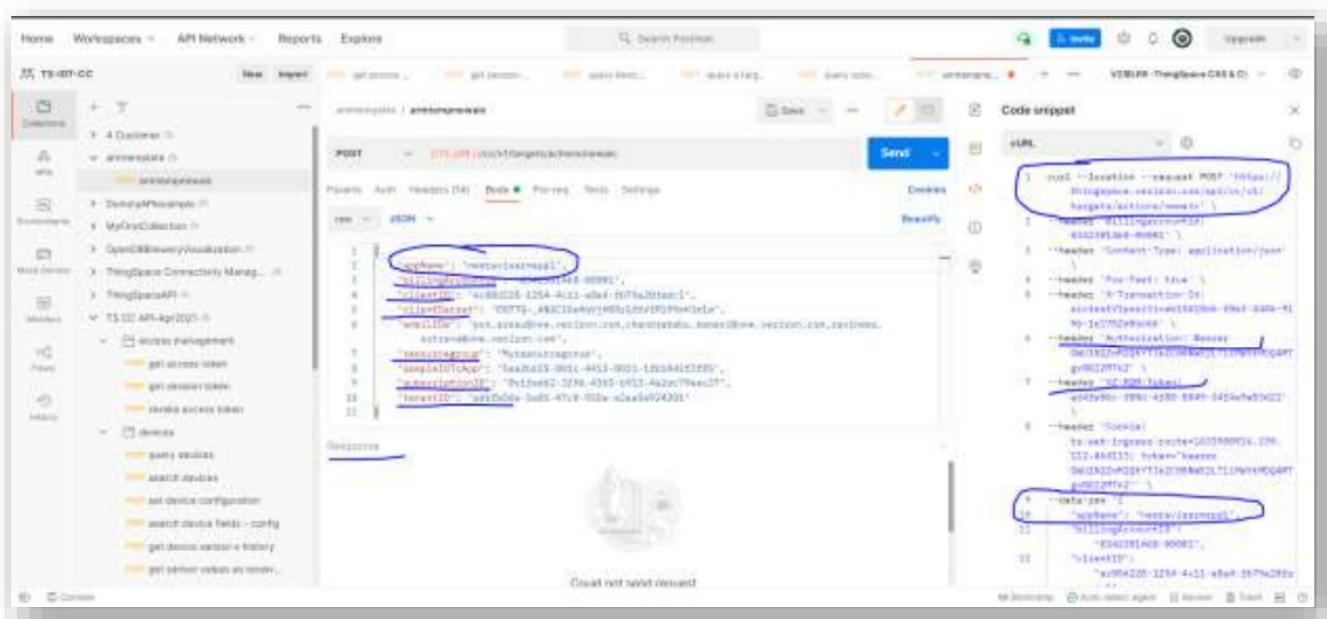
The cURL for the HTTPS POST Request:

```
curl -location -request POST 'https://thingspace.verizon.com/api/cc/v1/targets/actions/newaiaic' \
  -header 'Billingaccountid: 0342301460-00001' \
  -header 'Content-Type: application/json' \
```

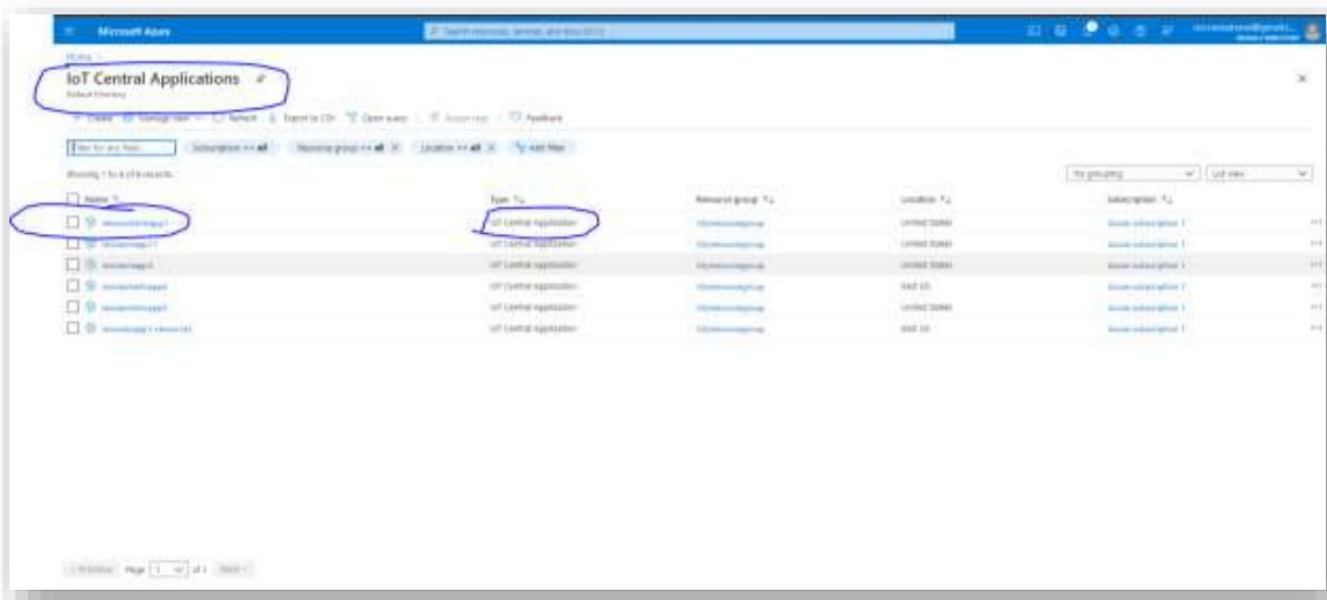
```

-header 'For-Test: true' \
-header 'X-Transaction-Id: aictestV1positiveb13d18d6-f0e3-4446-9196-1d1752d0bd66' \
-header 'Authorization: Bearer OWU1N2ZmM2QjYtYjZkZC00NWE2LTliMWYtMDQ4MTgxNGI2MTk2' \
-header 'VZ-M2M-Token: ad4fe06c-308d-4f88-8049-34f4e9e03d22' \
-header 'Cookie: ts-web-ingress-route=1633900916.239.112.464113; token="bearer OWU1N2ZmM2QjYtYjZkZC00NWE2LTliMWYtMDQ4MTgxNGI2MTk2"' \
-data-raw '{
  "appName": "newravisarmapp1",
  "billingAccountID": "0342301460-00001",
  "clientId": "ec08d228-1254-4c11-a8a4-fb79a20fadcd",
  "clientSecret": "EN77Q~_ANGC1GeHqVjH08zGfbVfR199mHJm1w",
  "emailIDs":
    "pon.arasu@one.verizon.com,chandrababu.kamani@one.verizon.com,ravinder.sutrave@one.verizon.com",
  "resourcegroup": "Myresourcegroup",
  "sampleIOTcApp": "5aa2bd25-801c-4413-8031-1fb1842f3f85",
  "subscriptionID": "0c1fed62-3296-4365-b913-4a2dc79eec27",
  "tenantID": "ad6fb56e-5a01-47c0-955e-e2aa5e924201"
}'

```



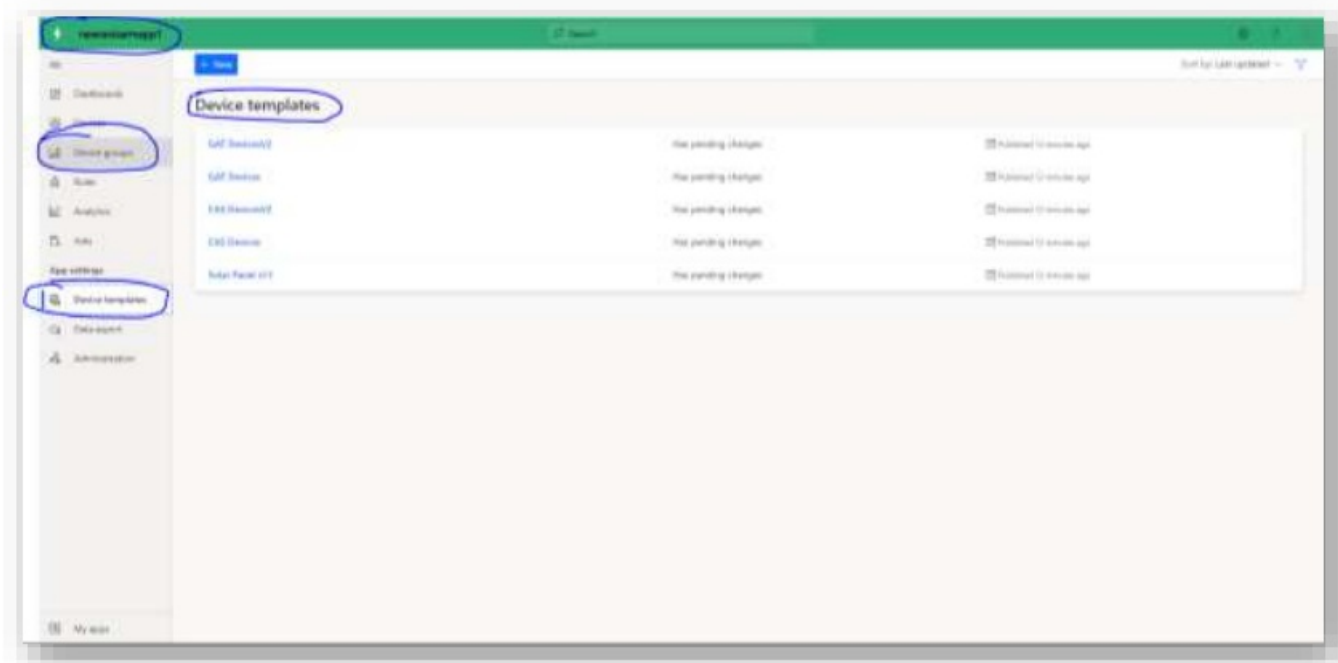
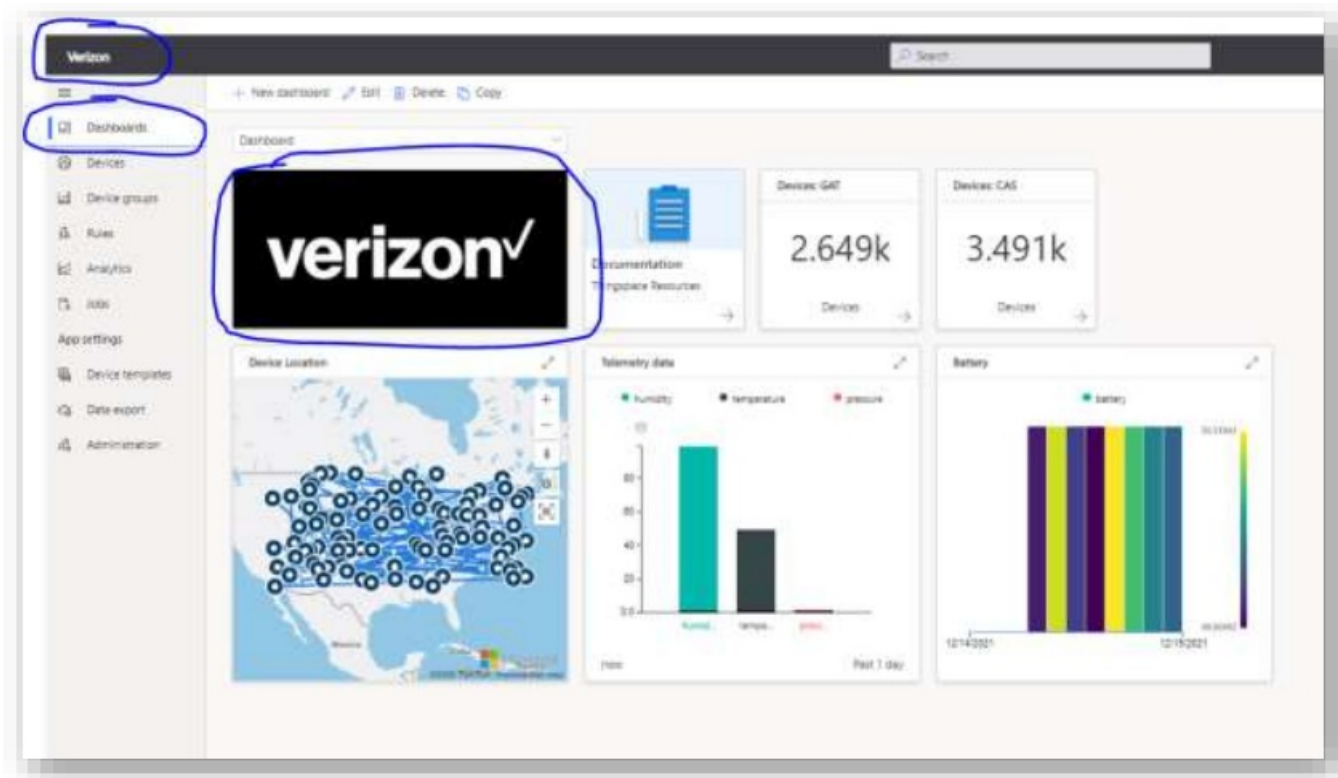
After successful execution of this ARM Template API, you will get the final product – Azure IOT Central App, with Verizon's Device Templates, Dashboards, (and Rules, Analytics and Jobs if any configured in the sample Application template). Please see the screenshot below showing the newly created Azure IOT Central App called "newravisarmapp1":



NOTE: There is a Microsoft bug here; it is supposed to show the Azure IOT App that was just created.



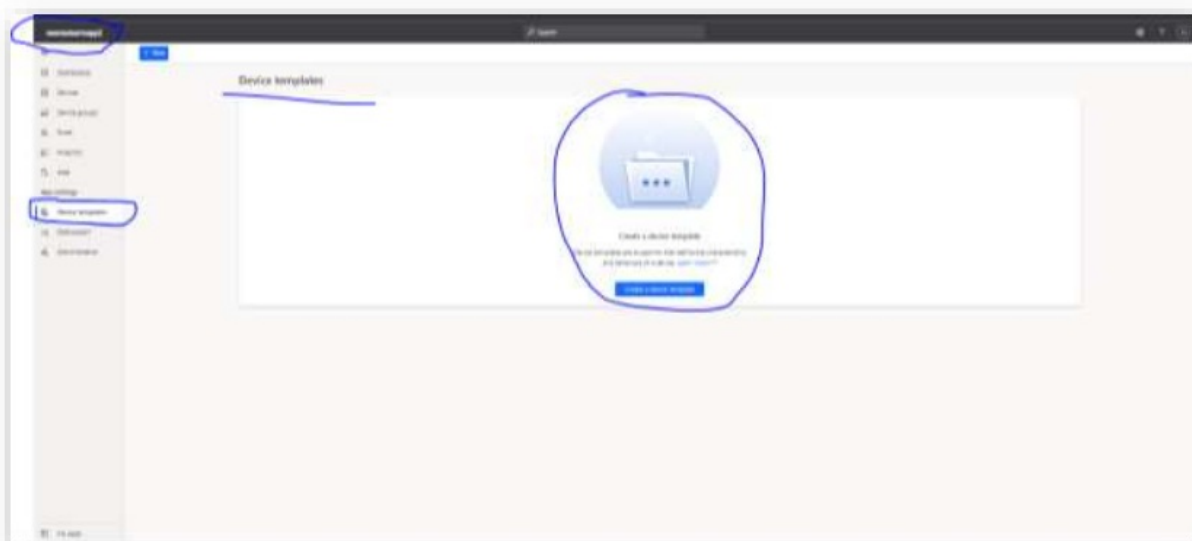
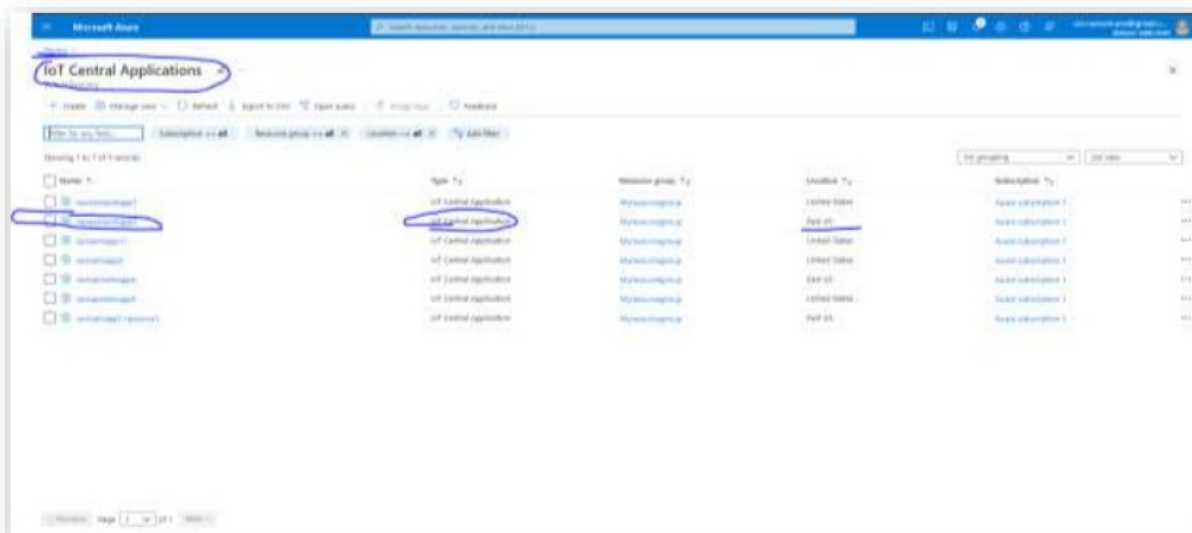
Alternatively, if we type the actual Azure IOT Central App URL, [<https://newravisarmapp1.azureiotcentral.com>] in any browser you will see the App, as shown below. And further, as you can see the Device Template also gets populated with Verizon's CAT/GAT device templates, as expected:

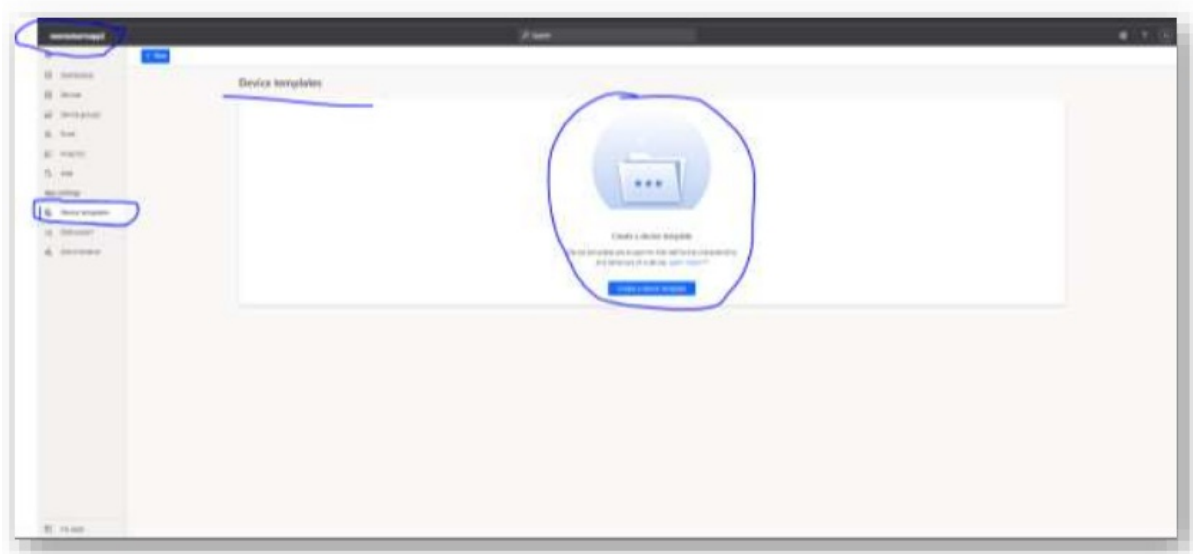
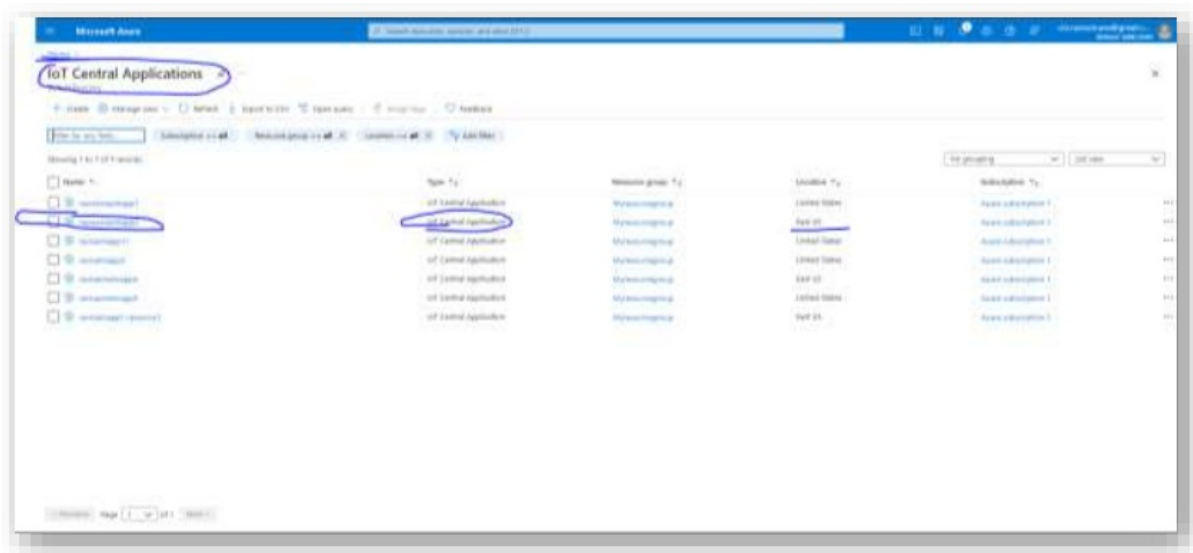


Differences between the AIC App directly from the Portal

The following section shows the difference of AIC App creation from the Portal directly, versus the AIC App from the ARM Template APIs.

Here is what happens if the User Attempts to create Azure IOT Central Apps directly from the Azure Portal, without Verizon's Cloud Connector for Asset Trackers:





References

- <https://docs.microsoft.com/en-us/azure/active-directory/develop/app-objects-and-service-principals>
- <https://docs.microsoft.com/en-us/azure/active-directory/develop/howto-create-service-principal-portal>
- Assign Azure roles using the Azure portal – Azure RBAC
- <https://github.com/iot-for-all/iot-central-aad-setup>
- There are more details on the concept of the Authorization model, including scopes, permissions and consent at the following referenced document (https://docs.microsoft.com/en-us/azure/active-directory/develop/v2-permissionsand-consent?WT.mc_id=Portal-Microsoft_AAD_RegisteredApps)

Appendix

Technically, an ARM template is nothing but a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. In declarative syntax, you describe your intended deployment without writing the sequence of programming commands to create the deployment.

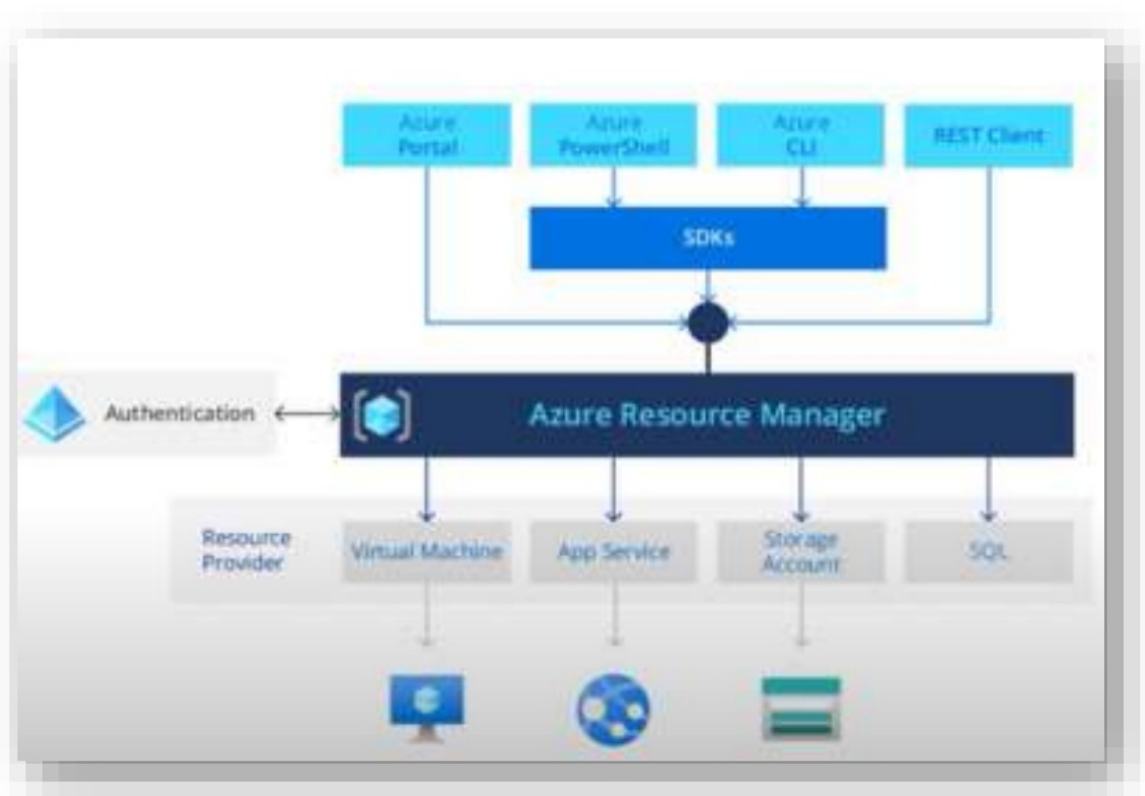
Why Active Directory App for this exercise/task?

The primary reason why we choose AD is that it plays the central role in managing the users/apps to give access, provide the identity, and permissions to the required IOT Central Applications.

We deploy Resources in Azure today by any of the following methods:

1. Azure Portal
2. Azure PowerShell
3. Azure CLI
4. Azure REST Client

All deployment interfaces are communicating with a single central location called Azure Resource Manager (ARM) as end-point



- ARM is the centralized layer for Resource Management in Azure. This is by default secure, because it is Authenticated and Secured with Azure AD
- ARM interface and communicates to a Resource Provider. Every resource like, a VM, App Service, Security groups, Policies, Subscriptions, etc. have its own Resource Provider, that handles everything related to that specific resource. On north-bound ARM interfaces with Portal, etc., and on south-bound ARM interfaces with Resource Provider/Resources

Notes on the ARM template

Microsoft recommends the following scenarios for using ARM Templates:

Application development and Maintenance, with IaC (Infrastructure as a Code)

CI/CD scenarios

Custom Applications with Cloud Connectors

Azure governance:

- Policies
- Roles
- Subscriptions
- etc

Notes on Azure IOT Central – architecture, services and capabilities

Azure IoT Central offers advanced functionality that's simple to use. It uses multiple underlying technologies. The underlying technologies include Azure IoT Hub, Azure Device Provisioning Service (DPS), Azure Maps, Azure Time Series Insights, Azure IoT Edge, and others. If you need more granular control over the underlying technologies, you can consider assembling them in your own solution.

Connected logistics

Global logistics spending is expected to reach \$10.6 trillion in 2020. Transportation of goods accounts for the majority of this spending and shipping providers are under intense competitive pressure and constraints.


You can use IoT sensors to collect and monitor ambient conditions such as temperature, humidity, tilt, shock, light, and the location of a shipment. You can combine telemetry gathered from IoT sensors and devices with other data sources such as weather and traffic information in cloud-based business intelligence systems.

The benefits of a connected logistics solution include:







- Shipment monitoring with real-time tracing and tracking.
- Shipment integrity with real-time ambient condition monitoring.
- Security from theft, loss, or damage of shipments.
- Geo-fencing, route optimization, fleet management, and vehicle analytics.
- Forecasting for predictable departure and arrival of shipments.



Documents / Resources

	Verizon Critical Asset Sensor Create AIC Apps [pdf] User Guide Critical Asset Sensor Create AIC Apps, Critical Asset, Sensor Create AIC Apps, Create AIC Apps, Apps
---	--

References

-  [Verizon: Wireless, Internet, TV and Phone Services | Official Site](#)
-  [Apps & service principals in Azure AD - Microsoft Entra | Microsoft Learn](#)
-  [Create an Azure AD app and service principal in the portal - Microsoft Entra | Microsoft Learn](#)
-  [Overview of permissions and consent in the Microsoft identity platform - Microsoft Entra | Microsoft Learn](#)
-  [GitHub - Azure/iot-central-aad-setup: How to setup an Azure Active Directory application](#)
-  [Microsoft Azure](#)