Indoor map data management in Azure Maps.| Microsoft Docs

farah-alyasari

# Indoor map data management

Azure Maps provides the Indoor Maps SDK, which has a plug-in for the QGIS application. The Azure Maps QGIS plug-in, the Azure Maps APIs, and the Indoor Maps module can be used together to develop indoor maps for your application. This article overviews the data management pipeline for developing applications with indoor maps. The process starts with data ingestion into Azure Maps resources.Throughout the development process, the Azure Maps APIs can be used for heavy data modification, such as uploading or importing data. The Azure Maps QGIS plug-in can be used for light data modification, with a friendly interface. The Indoor Maps module can be used to render indoor maps and control indoor maps events. After development, data resource can be administrated using the Azure Maps QGIS plug-in.

## Prerequisites

You need to [obtain a primary subscription key](quick-demo-map-app.md#get-the-primary-key-for-your-account) by [making an Azure Maps account](quick-demo-map-app.md#create-an-account-with-azure-maps). The primary subscription key is a required parameter for all Azure Maps APIs.

In this article, the [Postman](https://www.postman.com/) application is used to call the Azure Maps APIs, but you may use any API development environment.

Before you feed data to the Azure Maps resources, make sure that your DWG package adheres to the [DWG package requirements](dwg-requirements.md).

## Data Ingestion

The first step is to upload your DWG package using the Data Upload API. The Data Upload API consumes a DWG package, and it returns a URL pointing to the uploaded package. Once the package is uploaded, you can use the URL to access and download the package. The URI also contains the udid of the uploaded package. Use the package udid as an input to the Conversion API. This API accesses the uploaded DWG package and validates it against the [DWG package requirements](dwg-requirements.md). If the package meets the requirements, it’s converted from design data to map data. Then, a resource URL is returned. The URL contains the conversion udid, and obtaining this ID concludes the data ingestion phase. You can always come back and import more data into you data, using the Dataset Import API. Once the data is uploaded and converted, the map data can be accessed using the conversion ID and modified by other APIs.

The Conversion API will return error codes in the response body if the DWG package doesn’t meet the requirements. If you come across errors, then see Conversion API error codes for details.

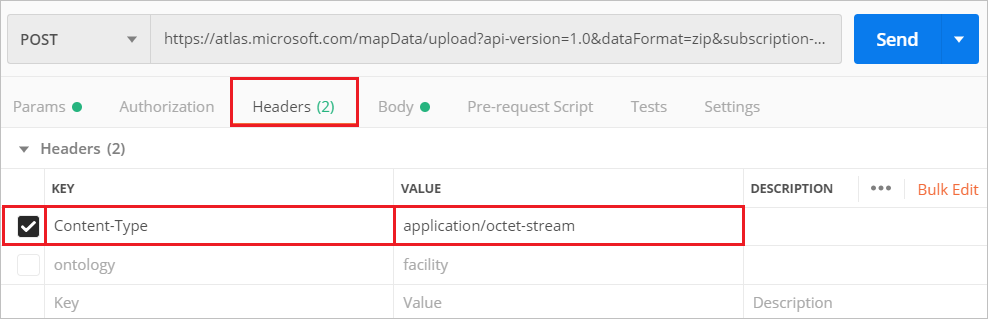
### Data upload process

Follow the processes below to upload your DWG design package:

1. Open the Postman app. Near the top of the Postman app, select **New**. In the **Create New** window, select **Collection**. Name the collection and select the **Create** button.
2. To create the request, select **New** again. In the **Create New** window, select **Request**. Enter a **Request name** for the request. Select the collection you created in the previous step, and then select **save**.
3. Select the **POST** HTTP method in the builder tab and enter the following URL to upload your DWG package to the Azure Maps service. For this request, and other requests mentioned in this article, replace <Azure-Maps-Primary-Subscription-key> with your primary subscription key.

* https://atlas.microsoft.com/mapData/upload?api-version=1.0&dataFormat=zip&subscription-key=<Azure-Maps-Primary-Subscription-key>

1. In the **Headers** tab, specify a value for the Content-Type key, as in the image below. In this case, our DWG package type is application/octet-stream. In the **Body** tab, select **binary**. Click on **Select File** and choose your DWG package.

* 
* data-management

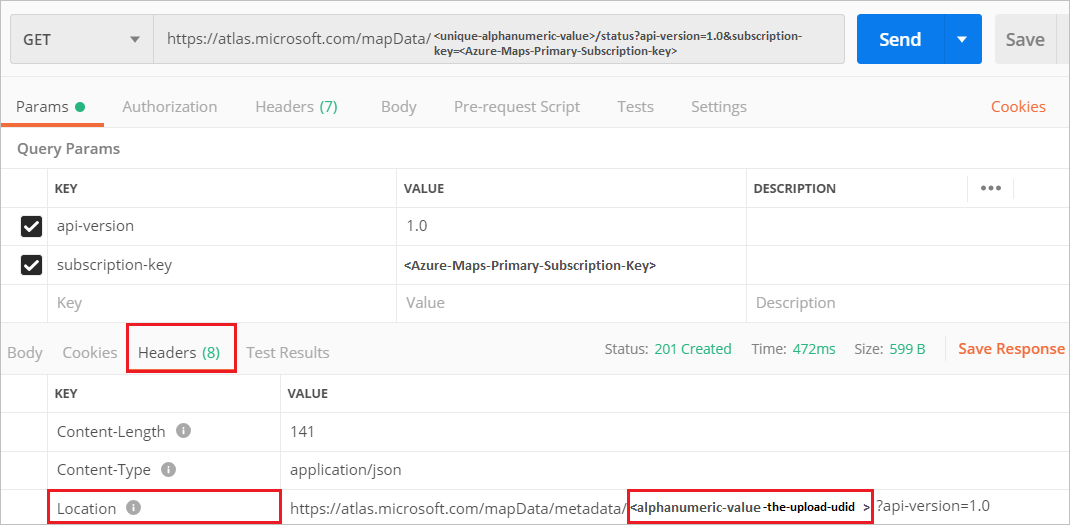
1. Click the blue **Send** button and wait for the request to process. Once the request completes, go to the **Headers** tab for the response, and copy the URL value for the **Location** key. This value points to the Azure Maps resource that contains the DWG package, and it looks like this:

* https://atlas.microsoft.com/mapData/<unique-alphanumeric-value>/status?api-version=1.0

1. Obtain the package ID, or the package udid, by making a **GET** HTTP request at the **Location** URL. You’ll need to append your primary subscription key to the URL, as seen below:

* https://atlas.microsoft.com/mapData/<unique-alphanumeric-value>/status?api-version=1.0&subscription-key=<Azure-Maps-Primary-Subscription-key>

1. When the **GET** HTTP request completes successfully, the response header will contain the udid of the uploaded DWG package. Copy the udid, it’s highlighted in this image.

* 
* data-management

### Data conversion process

keep the Postman application open. Now that your DWG package is uploaded, we’ll use the package udid, from the previous section, to convert the DWG package into map data.

1. Select **New** again. In the **Create New** window, select **Request**. Enter a **Request name**, and select a collection. Click **Save**.
2. Select the **POST** HTTP method in the builder tab and enter the following URL to convert your uploaded DWG package into map data. Use the udid of the uploaded DWG package.

* https://atlas.microsoft.com/conversion/convert?subscription-key=<Azure-Maps-Primary-Subscription-key>&api-version=1.0&udid=<alphanumeric-value-upload-udid>&inputType=DWG

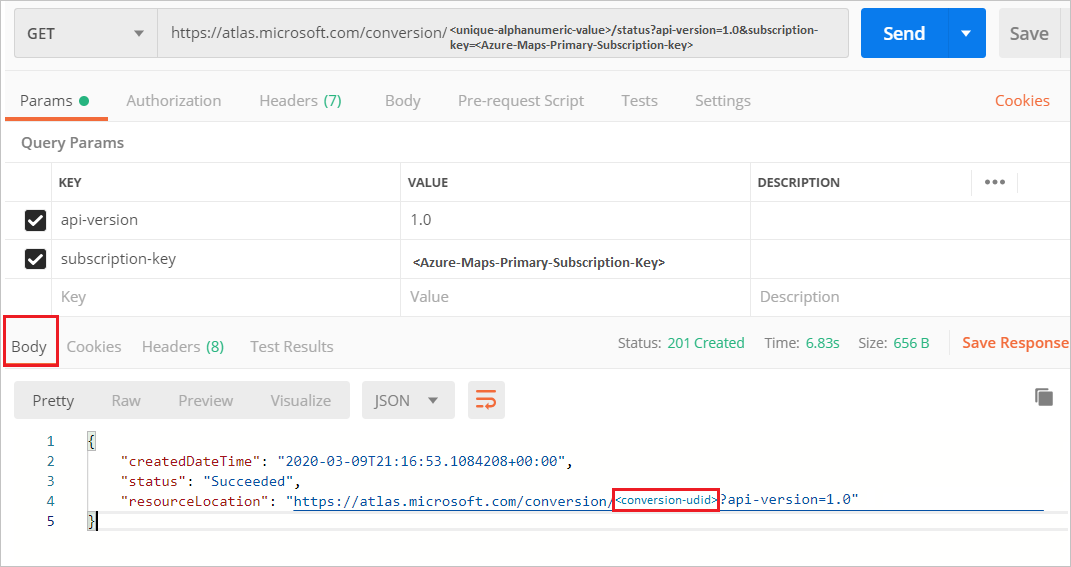
1. When the request completes, you’ll receive a response header containing a **Location** key. The **Location** key returned by the conversion API holds the URL of the converted map data. Copy the location URL, it would have a similar format as the URL below:

* https://atlas.microsoft.com/conversion/<unique-alphanumeric-value>/status?api-version=1.0

1. Start a new **GET** HTTP method in the builder tab. Make a **GET** request at the location URL from the previous step and append your Azure Maps primary subscription key.

* https://atlas.microsoft.com/conversion/<unique-alphanumeric-value>/status?api-version=1.0&subscription-key=<Azure-Maps-Primary-Subscription-key>

1. Once the request completes successfully, you’ll see a success status message in the response body. Copy the udid of the converted map data from the “resourceLocation”. Or copy it from the **Headers** response tab. We’ll refer to this udid as the conversion ID for short, and we’ll use it in the data curation stage.

* 
* data-management

[!Note] Sometimes it’ll take a while to show the response body of the API call. If you’re using the postman application and it takes a long time for the response status to change, then click on the send button again.

## Data Curation

In the data curation stage, the way the map data is managed depends on your application. The data set APIs, tile set APIs, and state set APIs will regularly be used in this stage. The Data set APIs let you create, list, and delete a data set. You can also import more data into a data set after it has been uploaded. To clarify, a data set is the primary resource for compiling one or more facility data. A tile set is a set of gridded vector tiles. A state set contains state information for map properties that can be dynamically manipulated. The tile set and state set APIs let you create, list, and delete a tile set and a state set, respectively.

You’ll first need to make a data set, before you can generate a tile set or a state set, because the tile set and the state set use data from the data set. However, each of the tile set and state set are independent entities. In another word, if you delete the data set that you used to generate the tile set with, then the tile set will continue to exist. You will also need to use the Azure WFS service to learn about your map features, and obtain a feature ID to use as a parameter for the state set APIs. Normally, the tile sets and state sets are used in combination with the Indoor Maps module to render tiles and feature states on the map.

The next sections detail the process to generate data sets, tile sets, and state sets. In the state sets section, the steps demonstrates how to use Azure Maps WFS service to learn about your feature and obtain a feature ID. The Azure Maps QGIS plug-in section and the Indoor Module section provide an overview of each tool. Because these tools entail a lot of details, we linked how-to guides for each the QGIS plug-in and the Indoor Module at the end of this article.

### Data sets

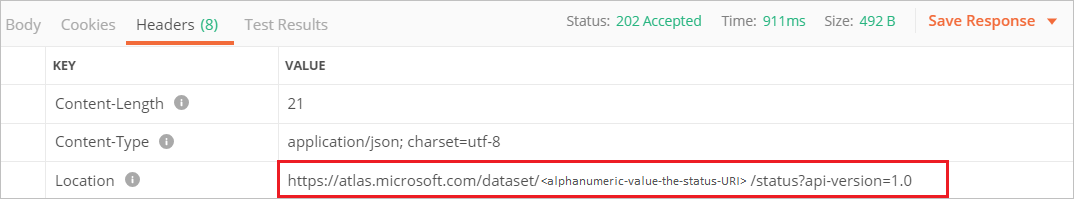
The data set is a collection of map data entities, such as buildings. To make a data set, use the Dataset Create API. It takes a conversion ID of a DWG package, and produces a data sets, with a unique data set ID.

The steps below show you how to make a data set, and obtain its ID:

1. Open the Postman application. Select **New**. In the **Create New** window, select **Request**. Enter a **Request name**, and select a collection. Click **Save**
2. Make a **POST** request to the Dataset Create API to create a new data set. The URL of the request should have a format like the one below:

* https://atlas.microsoft.com/dataset/create?api-version=1.0&conversionID=<your-dwg-package-conversion-ID>type=facility&subscription-key=<Azure-Maps-Primary-Subscription-key>

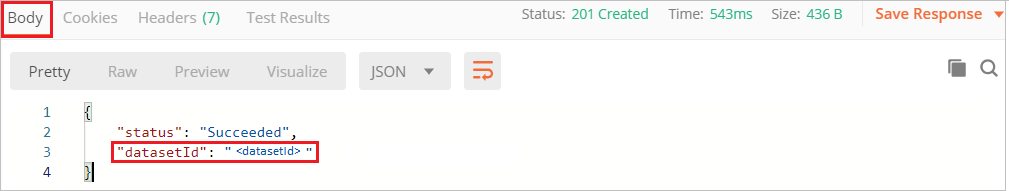
1. Obtain the URL in the **Location** key of the response **Headers** tab.

* 
* data-management

1. Make a **GET** request at the URL to Obtain the data set ID. As usual, append your Azure Maps primary subscription key for authentication.

* https://atlas.microsoft.com/dataset/<alphanumeric-value-status-URI>/status?api-version=1.0&subscription-key=<Azure-Maps-Primary-Subscription-key>

1. Copy the datasetId in the response body.

* 
* data-management

You can use your data set ID for various purposes. For example, you can use the [Azure Maps QGIS plug-in](azure-maps-qgis-plugin.md) to edit the data set. You may access your data set features by querying the Azure Maps Web Feature Service (WFS) API. It’s okay to use another WFS API, as long as it follows the [OGC Web Feature Service 3.0](http://docs.opengeospatial.org/DRAFTS/17-069.html)

You can also generate a tile set and state set from the data in the data set, by using the data set ID. Then, you can [use the Indoor Maps module](how-to-use-indoor-module.md) to render your vector tiles or feature states. And at any time, you may use the Dataset List API to view details of the data sets you have generated. When you’re done using a data set, you can remove it using the Dataset Delete API.

[!Note] The Dataset Create API doesn’t prevent you from appending duplicate blob into a data set.

### Tile sets

A tile set is a set of vector tiles to render on the map. It’s produced from the data in a data set, so, the Tileset Create API requires a data set ID. And although the tile set is built from the data in the data set, it’s an independent entity. If the data set is deleted, the tile set will continue to exist.

To create a tile set, follow the steps below:

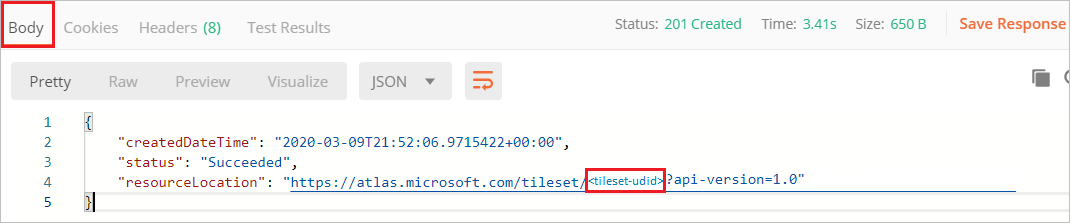
1. Open the Postman application. Select **New** again. In the **Create New** window, select **Request**. Enter a **Request name**, and select a collection. Click **Save**
2. Make a **POST** request in the builder tab. The request URL should look like to the one below:

* https://atlas.microsoft.com/tileset/create/vector?api-version=1.0&datasetID=<datasetID>&subscription-key=<Azure-Maps-Primary-Subscription-key>

1. The response headers will contain a location header with the tile set URL. Make a **GET** request at the URL, with your Azure Maps subscription appended to the end for authentication.

* https://atlas.microsoft.com/tileset/operations/<alphanumeric-value-status-URI>?api-version=1.0&subscription-key=<Azure-Maps-Primary-Subscription-key>

1. Upon a successful request, you’ll receive the tile set udid in the response body. For short, this udid is later referred to as the tile set ID.

* 
* data-management

To display the map data that was originally obtained from the DWG package, [use the Indoor Maps module](how-to-use-indoor-module.md#use-the-indoor-maps-module) with your tile set ID. You may also render indoor maps using the Get Map Tile API. You can also fetch a list of all the tile sets, including the tile sets that aren’t used in your application, using the Tileset List API.

To delete your tile set, use the Tileset Delete API. If you delete an in use tile set, then those tiles won’t render on the map at the application runtime.

[!Caution] Whenever you review the list of items and decide to delete them, consider the cascading dependencies impact. For example, you may have a tile set being rendered in your application using the **Get Map Tile API** and deleting the tile set will result in failure to render that tile set.

### State sets

A state set contains style information of properties that can dynamically change. Accessing and changing the style of a feature is useful for highlighting events or states. For example, you may toggle the tile colors to indicate the room occupancy status.

The process of using the feature state set APIs is similar to using the data set and tile set APIs. You would first make a state set. And you would use the feature ID and the state set ID to update feature. The steps below explain how to create a state set, obtain the feature ID, update the feature ID state, and acquire the state of the feature.

1. Open the Postman application. Select **New**. In the **Create New** window, select **Request**. Enter a **Request name**, and select a collection. Click **Save**
2. If you don’t already have a feature ID, obtain a feature ID from a data set using the WFS API. First, make a **GET** request to learn about the collections in your data set.

* https://atlas.microsoft.com/wfs/datasets/<datasetID>/collections?subscription-key=<Azure-Maps-Primary-Subscription-key>&api-version=1.0

1. The response body will contain a set of feature collections in your data set. Once you choose a collection, make a **GET** request at the collection URL to learn about its features. The URL format would look like the one below:

* https://atlas.microsoft.com/wfs/datasets/<datasetID>/collections/unit/items?subscription-key=<Azure-Maps-Primary-Subscription-key>&api-version=1.0

1. Choose and copy a feature **ID** from the response body. We’ll refer to the dynamic properties of this feature as states, and we’ll use the feature ID to modify its states.
2. At this point we have a feature ID and data set ID. To make a state set, make a **POST** request to the Create Stateset API. Use the data set ID of the data set that contains the state you want to modify. Here’s how the URL should look like:

* https://atlas.microsoft.com/featureState/stateset?api-version=1.0&datasetID=<datasetID>&subscription-key=<Azure-Maps-Primary-Subscription-key>

1. Copy the state set ID from the response body.
2. Use the Feature Update States API to update the state. Pass the state set ID, data set ID, and feature ID, with your Azure Maps subscription key. Here’s a URL of a **POST** request to update the state:

* https://atlas.microsoft.com/featureState/state?api-version=1.0&statesetID=<statesetID>&datasetID=<datasetID>&featureID=<featureID>&subscription-key=<Azure-Maps-Primary-Subscription-key>

1. In the **Headers** of the **POST** request, set Content-Type to application/json. In the **body** of the **POST** request, write the JSON with the feature updates. The updates will only occur if the time posted stamp is after the time stamp of the previous request. Here’s a sample JSON body that updates a feature:

* {  
   "states": [  
   {  
   "keyName": "occupied",  
   "value": true,  
   "eventTimestamp": "2019-11-14T17:10:20"  
   }  
   ]  
  }

1. You may use the Feature Get States API to obtain the status of the state. The URL below shows how the **GET** request would look like:

* https://atlas.microsoft.com/featureState/state?api-version=1.0&statesetID=<statesetID>&datasetID=<datasetID>&featureID=<featureID>&subscription-key=<Azure-Maps-Primary-Subscription-key>

All the dynamic styling properties of all features can become states, where the color is a common style to add as a state set. For example, the indoor map tile color can change based on room fire code occupancy limit. Another use case, would be to toggle the tile color based on room availability, for a scheduling application. For more information, see Implement dynamic styling for Private Atlas Indoor Maps

The Feature Get States API lets you learn about the state of a feature, using its feature ID. You can render the states on the indoor map using the Indoor Maps manager from the Indoor Module or the Azure Maps QGIS plug-in. You can also delete the state set and its resource using the Feature State Delete API.

[!Caution] Whenever you review the list of state sets and decide to delete them, consider the cascading dependencies impact. For example, if you’re rendering your state set using the Indoor Maps module, deleting a state set will result in failure to render that state set.

### Azure Maps QGIS Plug-in

The Azure Maps QGIS Plug-in lets you make the same data map manipulation as the Azure Maps APIs. However, it’s recommended that you use the plug-in for minor data modification and use the APIs for heavy revisions. In the main window of the plug-in, the user can see the indoor map rendered as it would render in the application. Changes to the data set are observed right after the modification, in near real-time.

The plug-in is still in the experimentation phase. So, there are some limitations to keep in mind. Only one user at a time can make changes to the data set. And before switching the floor number, make sure you save your edits for the current floor. Changes done on a given floor will be lost if you don’t save before changing the floor. For more information, see the [Azure Maps QGIS Plug-in](azure-maps-qgis-plugin.md).

### Indoor Maps Module

The Azure Maps Web SDK provides the Indoor Maps module. The Indoor Maps module is an extension to the Map Control module, with functionalities specifically tailored to indoor maps development. It offers a control for the indoor map floor level. The user can attach listener events to execute when a floor level changes, or when the facility changes. To use the Azure Maps module, you would need to obtain the tile set id to render tiles on the maps. You also render feature states using the state set id. Both IDs must be assigned in the Indoor Maps manager. Overall, the Azure Maps web SDK and the Indoor Maps module makes it convenient to integrate indoor maps with web applications. To get started with the module, read [How to use the Indoor Maps module](how-to-use-indoor-module.md) and the Indoor Module SDK documentation.

## Resource Administration

At this phase, your tile sets are ready for indoor maps rendering. Your data sets can be queried using a WFS service. And, your feature state service can accept dynamic property updates. You can continue to monitor the status of your map data using the list APIs available in the Dataset, Tileset, and Feature State services. You can review the relevance of your map data. If you wish, you may update or delete the data, using the update and delete APIs, for the respective service.

The Azure Maps QGIS plug-in is another convenient tool to administer your map data. The plug-in interface lets you see real-time updates reflected based on changes to your data set. You can also make minor modification using a graphical user interface rather than making API calls.

## Next steps

Learn more about the different APIs mentioned in this application:

[!div class=“nextstepaction”] Data Upload

[!div class=“nextstepaction”] Data Conversion

[!div class=“nextstepaction”] Dataset

[!div class=“nextstepaction”] Tileset

[!div class=“nextstepaction”] WFS service

[!div class=“nextstepaction”] Feature State set

Relevant articles:

[!div class=“nextstepaction”] [Azure Maps QGIS Plug-in](azure-maps-qgis-plugin.md)

[!div class=“nextstepaction”] Implement dynamic styling for Private Atlas Indoor Maps

[!div class=“nextstepaction”] [Use the Indoor Maps module](how-to-use-indoor-module.md)

[!div class=“nextstepaction”] [Bulk import data to an existing data set](how-to-ingest-bulk-data-in-dataset.md)

[!div class=“nextstepaction”] Indoor Module SDK documentation

Consider leveraging more Azure Maps services with your indoor maps application. For example, if your application is “smart”, then you may incorporate IoT system powering location logic or produce map visualization. Learn more by reading: