Use Private Atlas to render indoor maps | Microsoft Docs

# Use Private Atlas to render indoor maps

This tutorial shows you how to use the Azure Maps Private Atlas APIs. The Private Atlas lets you show indoor maps within your web application, and it provides you with the necessary functionalities to control your indoor maps data. In this tutorial, you learn how to use the Private Atlas APIs to:

[!div class=“checklist”]

\* Upload your indoor map DWG package

\* Convert your DWG package into map data

\* Create a data set from your map data

\* Create a tile set from the data in your data set

\* Query the Azure Maps WFS service to learn about your map features

\* Create a feature state set by using your map features and the data in your data set

\* Update your feature state set

## Prerequisites

To use the Azure Maps Private Atlas APIs, [make an Azure Maps account](quick-demo-map-app.md#create-an-account-with-azure-maps) and [enable Private Atlas](how-to-manage-private-atlas.md) in your Azure Maps. [Obtain a primary subscription key](quick-demo-map-app.md#get-the-primary-key-for-your-account), also known as the primary key or the subscription key.

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

This tutorial uses the Postman application, but you may choose a different API development environment.

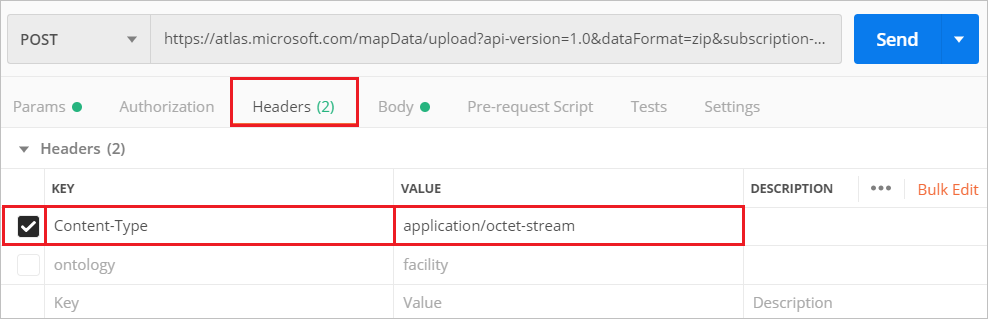
## Upload DWG package

Begin by using the Data Upload API to upload your DWG package to Azure Maps resources. The API returns a status URL pointing to the uploaded package. You can use the status URL to check the upload status. Download the uploaded data. Or obtain the udid for the uploaded data. In this case, we want the udid to access the uploaded package. Follow the steps below to obtain the udid.

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. Our DWG package is a zipped folder, so use the application/octet-stream value. 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 of the response. Copy the status URL, it’s the value for the **Location** key.
2. Obtain the udid for the package by making a **GET** HTTP request at the status URL. You’ll need to append your primary subscription key to the URL for authentication.
3. When the **GET** HTTP request completes successfully, the response header will contain the udid for the uploaded DWG package. Copy the udid.

* {  
   "status": "Succeeded",  
   "resourceLocation": "https://atlas.microsoft.com/mapData/metadata/<upload-udid>?api-version=1.0"  
  }

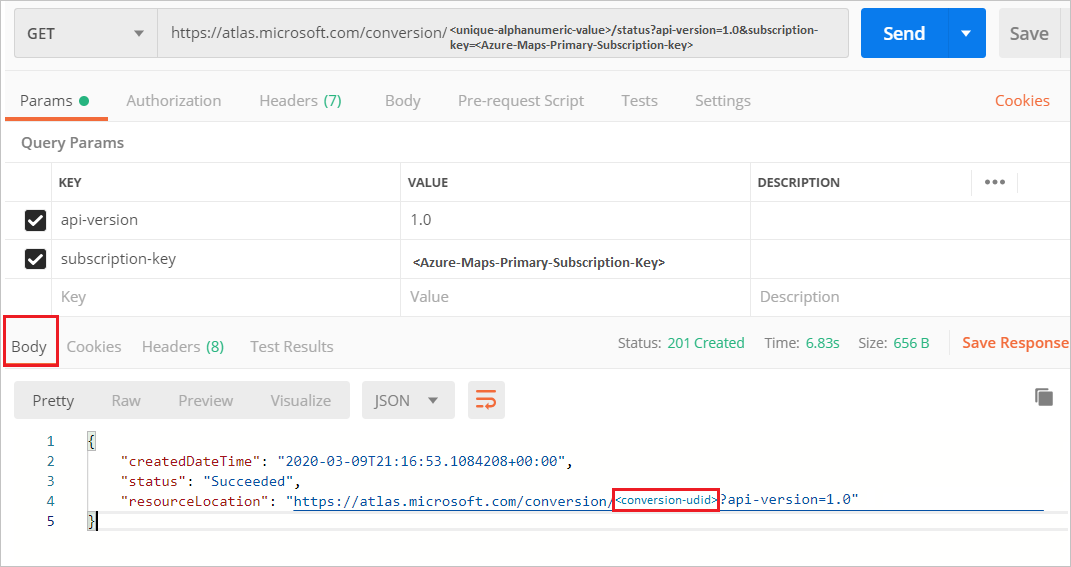
## Convert DWG package

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

1. Select **New**. 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 for the uploaded package.

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

1. Click the **Send** button and wait for the request to process. Once the request completes, go to the **Headers** tab of the response, and look for the **Location** key. Copy the status URL for the conversion request.
2. Start a new **GET** HTTP method in the builder tab. Make a **GET** request at the status URL from the previous step and append your Azure Maps primary subscription key.
3. Once the request completes successfully, you’ll see a success status message in the response body. Copy the udid for the converted package. It’s also known as the conversion ID, and it’s frequently used by other APIs to access the converted map data.

* 
* data-management

[!Note] Sometimes it may 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.

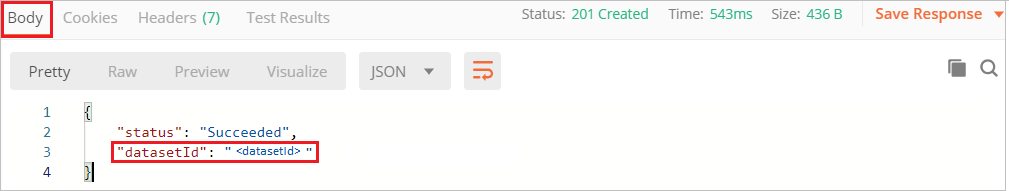
## Create a data set

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 for a DWG package, and it produces a set containing the map data. The steps below show you how to make a data set.

1. In 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=<conversion-udid>&type=facility&subscription-key=<Azure-Maps-Primary-Subscription-key>

1. Obtain the status URL in the **Location** key of the response **Headers**.
2. Make a **GET** request at the status URL to obtain the data set ID. Append your Azure Maps primary subscription key for authentication.
3. Copy the datasetId in the response body.

* 
* data-management

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. Don’t delete your data set yet, the next section shows you how to produce a tile set from the data in your data set.

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

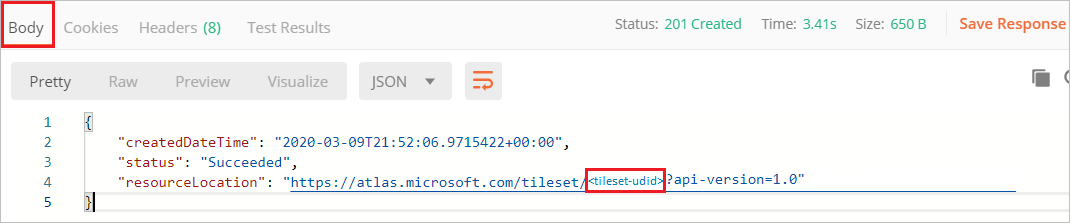
## Create a tile set

A tile set is a set of vector tiles that render on the map. It’s produced from the data in a data set, but it’s an independent entity. In other words, if the data set is deleted, the tile set will continue to exist. To create a tile set, follow the steps below:

1. In 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 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=<dataset-udid>&subscription-key=<Azure-Maps-Primary-Subscription-key>

1. Make a **GET** request at the status URL for the tile set. Append your Azure Maps primary subscription key for authentication.
2. Upon a successful request, you’ll receive the tile set udid in the response body. It may also be referred to as the tile set ID.

* 
* data-management

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. Use the Tileset List API to see all the tile sets in your Azure Maps Private Atlas.

## Query Azure Maps WFS

Learn about your map features by calling the WFS API. The steps below show you how to obtain a feature ID, we’ll use this feature ID later to create a feature state set.

1. In 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 **GET** request to the WFS API to view a list of the collections in your data set. Replace <dataset-udid> with your data set ID, and similarly use your Azure Maps primary key instead of the placeholder.

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

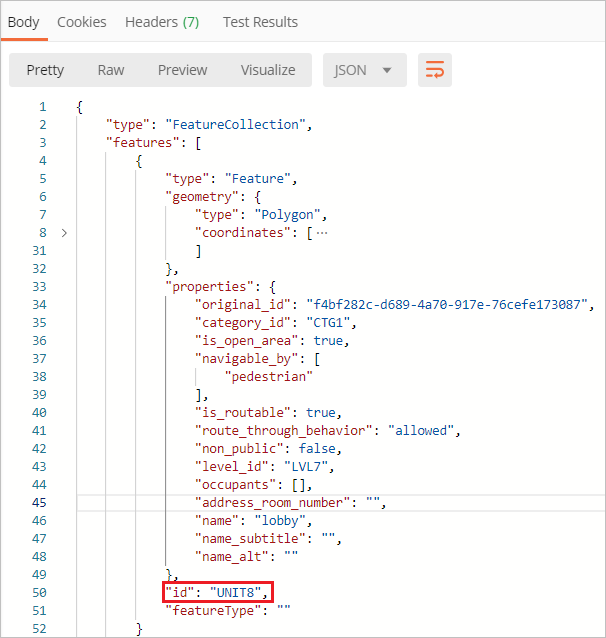
1. The response body will contain GeoJSON similar to the code shown below, which is not fully shown for simplicity. You may click the links inside the collections element and sub-elements to learn more about the described feature.

* {  
  "collections": [  
   {  
   "name": "unit",  
   "description": "A physical and non-overlapping area which might be occupied and traversed by a navigating agent. Can be a hallway, a room, a courtyard, etc. It is surrounded by physical obstruction (wall), unless the is\_open\_area attribute is equal to true, and one must add openings where the obstruction shouldn't be there. If is\_open\_area attribute is equal to true, all the sides are assumed open to the surroundings and walls are to be added where needed. Walls for open areas are represented as a line\_element or area\_element with is\_obstruction equal to true.",  
   "links": [  
   {  
   "href": "https://atlas.microsoft.com/wfs/datasets/<dataset-udid>/collections/unit/definition?api-version=1.0",  
   "rel": "describedBy",  
   "title": "Metadata catalogue for unit"  
   },  
   {  
   "href": "https://atlas.microsoft.com/wfs/datasets/ab8a71e7-ea96-fff9-6cc4-c857b997d9df/collections/unit/items?api-version=1.0",  
   "rel": "data",  
   "title": "unit"  
   }  
   ]  
   },

1. Make a **GET** request at a specific collection to learn about the features inside it. For example, to learn about features inside the unit collection, use the following URL:

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

1. Familiarize yourself with your map features. Choose and copy a feature **ID** for a feature that has style properties that can be dynamically modified. We’ll copy the feature **ID** for a feature inside the unit.

* 
* data-management

We’ll refer to the style property of this feature as states, and use it to make a state set that can be dynamically styled.

## Create a feature state set

1. In 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 Create Stateset API. Use the data set ID of the data set, which 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=<dataset-udid>&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 the URL of a **POST** request to update the state:

* https://atlas.microsoft.com/featureState/state?api-version=1.0&statesetID=<stateset-udid>&datasetID=<dataset-udid>&featureID=<feature-ID>&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 update will only be saved, if the time posted stamp is after the time stamp of the previous request. Here’s a sample JSON body that updates the feature we previously chose:

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

1. After you update the feature, the response **Headers** will contain the status URL for the feature. You may make a **GET** request at this URL to check if the feature has been updated successfully.

The Feature Get States API lets you learn about the state of a feature, using its feature ID. You can also delete the state set and its resource using the Feature State Delete API.

## Next steps

In this tutorial, you learned how to:

[!div class=“checklist”]

\* Upload your indoor map DWG package

\* Convert your DWG package into map data

\* Create a data set from your map data

\* Create a tile set from the data in your data set

\* Query the Azure Maps WFS service to learn about your map features

\* Create a feature state set by using your map features and the data in your data set

\* Update your feature state set

You’re now equipped with the skills you need to move on to the next guides:

[!div class=“nextstepaction”] Use the Indoor Maps module

[!div class=“nextstepaction”] Implement dynamic styling for indoor maps

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

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

Learn more about the different Azure Maps services discussed in this article:

[!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