

**Configuring the ArcGIS Indoors Sample Application**

Esri Content Team, Esri Professional Services

Contents

[1.0 Requirements 3](#_Toc476300442)

[2.0 Purpose of this Document 3](#_Toc476300443)

[3.0 Deploy the ArcGIS Indoors App 3](#_Toc476300444)

[4.0 Configure the Application for Your Services 4](#_Toc476300445)

# Requirements

The ArcGIS Indoors web app uses all of the services published in the previous steps of the ArcGIS Indoor Tools workflow:

1. Routing Service
2. Web Scene
3. Feature Services
4. Related tables

If you have not yet finished these steps, the app will still work in its default form with support services from ArcGIS Online. However, the support service data is specific to the Esri Campus, and will have limited value outside of Esri. Following the previous ArcGIS Indoors Tools steps with your own data will ensure that you create a useful business application with the ArcGIS Indoors web app that can help view your built environment in new ways, and leverage the science of where in your own organization’s day-to-day activities.

Software Requirements:

* A functioning web server such as IIS or Apache
* A viable JavaScript editing environment
* Access to supporting services and attribute information

# Purpose of this Document

This document provides the basic steps required to configure the ArcGIS Indoors sample application with the services created in the first 3 steps of the ArcGIS Indoors Tools research project.

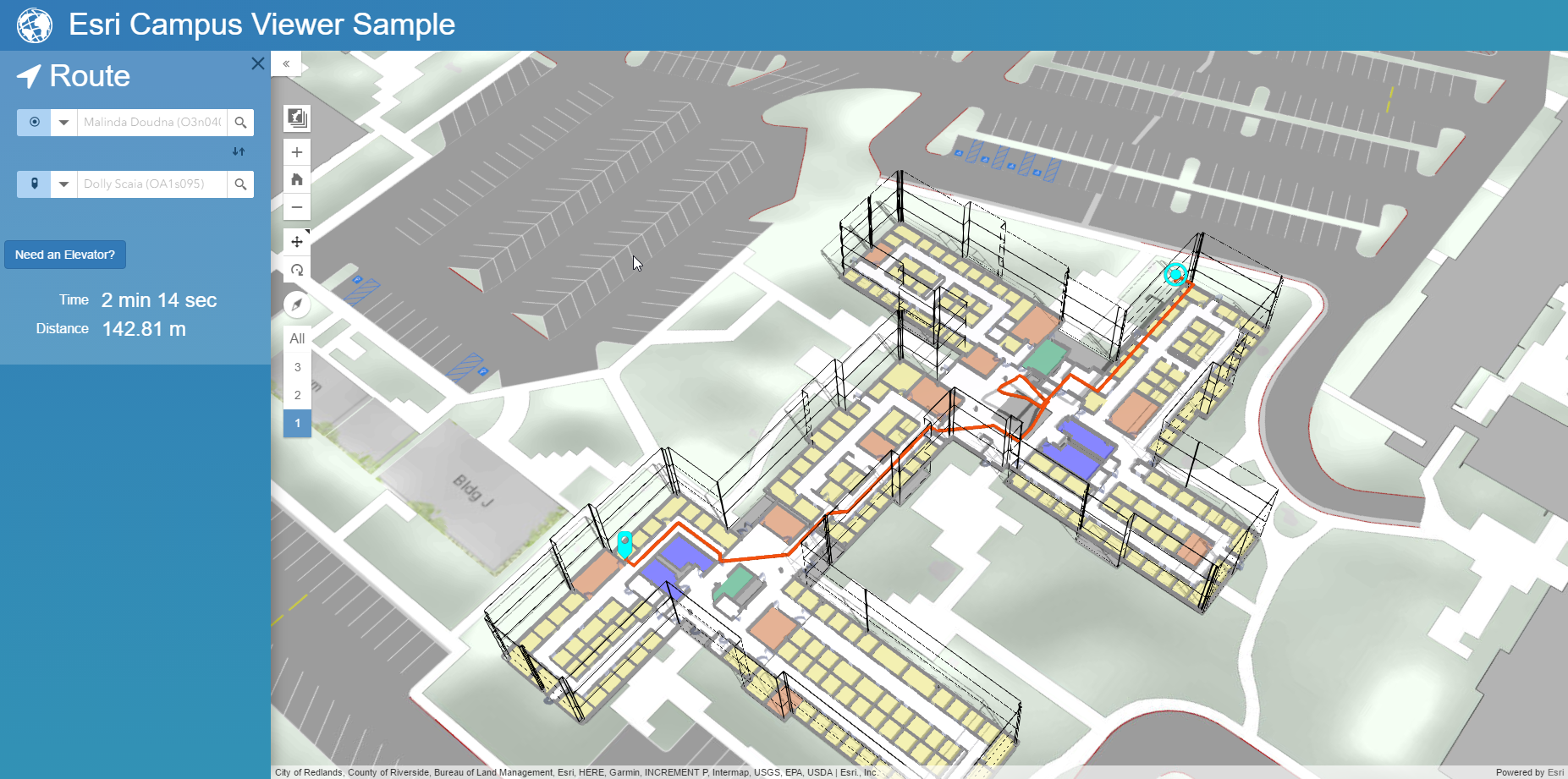
# Deploy the ArcGIS Indoors App

This step assumes you have some competence in web server and app management. If you are unsure how to deploy and administer web applications in your organization, it is a good idea to identify someone who does and make time to go through these steps with them.

* In the ArcGIS Indoors “Web Application” folder, find the home file for the app – arcgis\_campus\_navigation – and copy it to the web root directory of your web server, or a valid virtual directory.
* Browse to the application in a browser widow, and verify that it loads properly. By default, the application will be pointing to public support services for a building on the Esri campus. We will configure the app in the next section to use your new web scene, destination point, and routing services.



* Test the app by changing views with the “change view” widget, and navigating between some of the points on campus. By default, you can search on employees, rooms, and points of interest on the Esri Redlands campus. Try searching for “Jo Tanks,” and route from her office to “Dolly Scaia.” Take some time to familiarize yourself with the application.

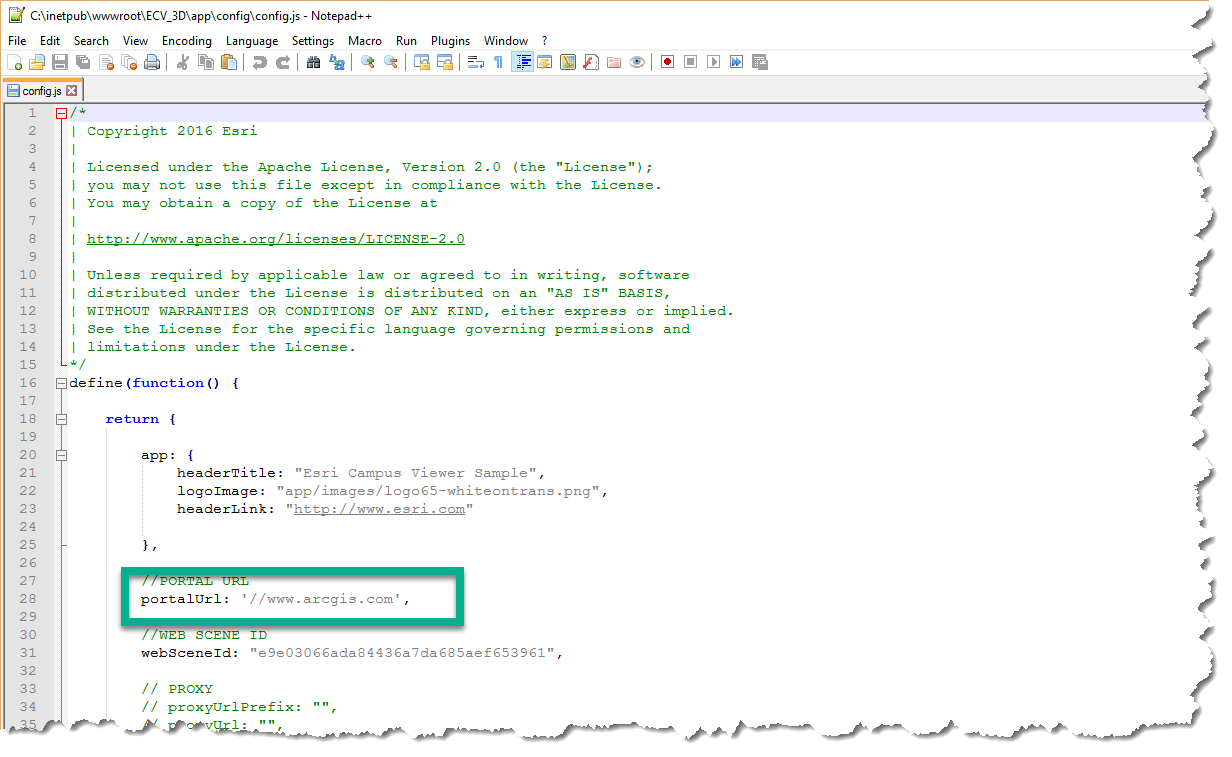


# Configure the Application for Your Services

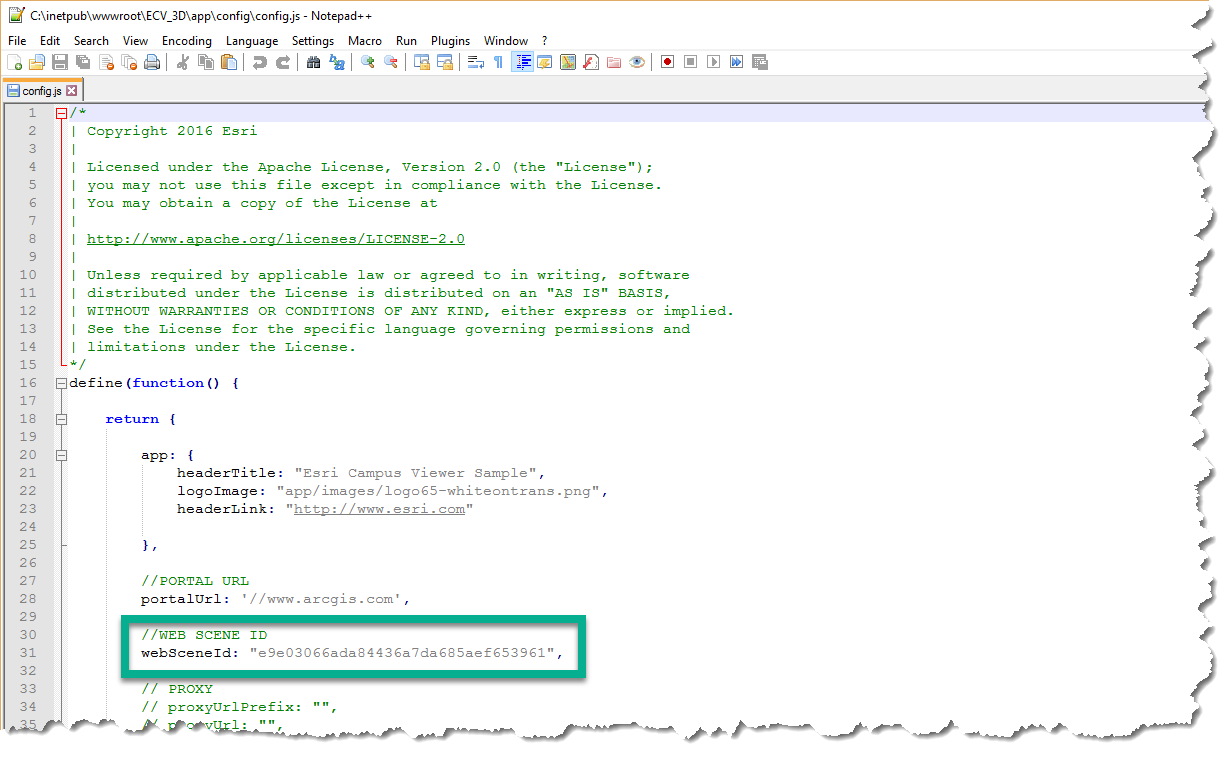
Configuration of the application is done by modifying the config.js file. This file is located in the …\<app home>\app\config folder.

Use your favorite Javascript editor to open the config.js file. If you don’t have a default option already installed, [Notepad ++](https://notepad-plus-plus.org/download/v7.3.2.html) is a good free option that provides basic Javascript code functions.

* Change the value for “portalURL” to match the location of your portal. By default, this will be pointing to ArcGIS online. This can stay the same if you are publishing using your AGOL organization, but will need to be changed if deploying to portal. For portal, but sure to include the “/portal” at the end of this URL.

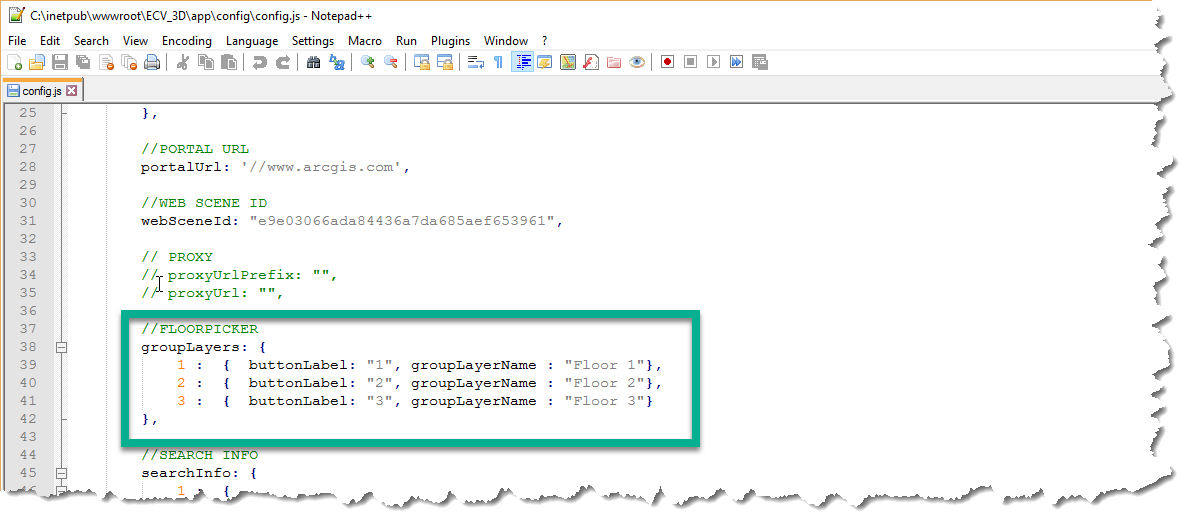


* Update the “webSceneId” value to the ID for your web scene. This value can be obtained by accessing the scene in a browser, and copying the id value from the URL. It can also be obtained from the item description in the portal if you prefer.



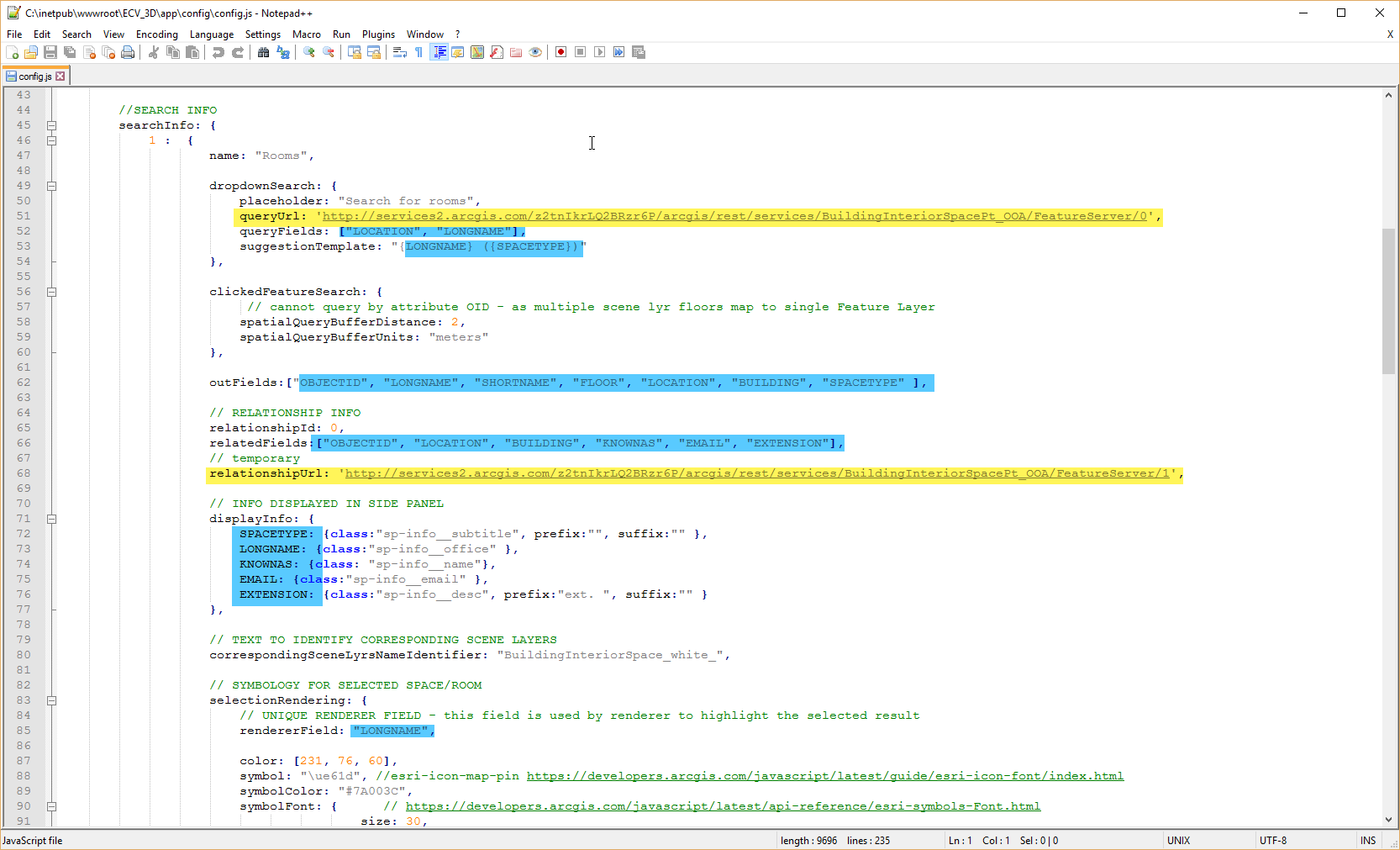
***HINT:*** Any slides you’ve captured in the web scene will automatically show up in the change view widget in the app. As a result, you might want to take time to configure a few slides for common views users will want to access for your campus.

* The floor display control or “floor picker” in the app is configured with the “groupLayers” JSON object. To configure the Floor Picker widget, you can list out all the floor layers in your web scene and the name you would like displayed in the widget. For the default data, there are 3 floors. Other floors can be added or removed as necessary.



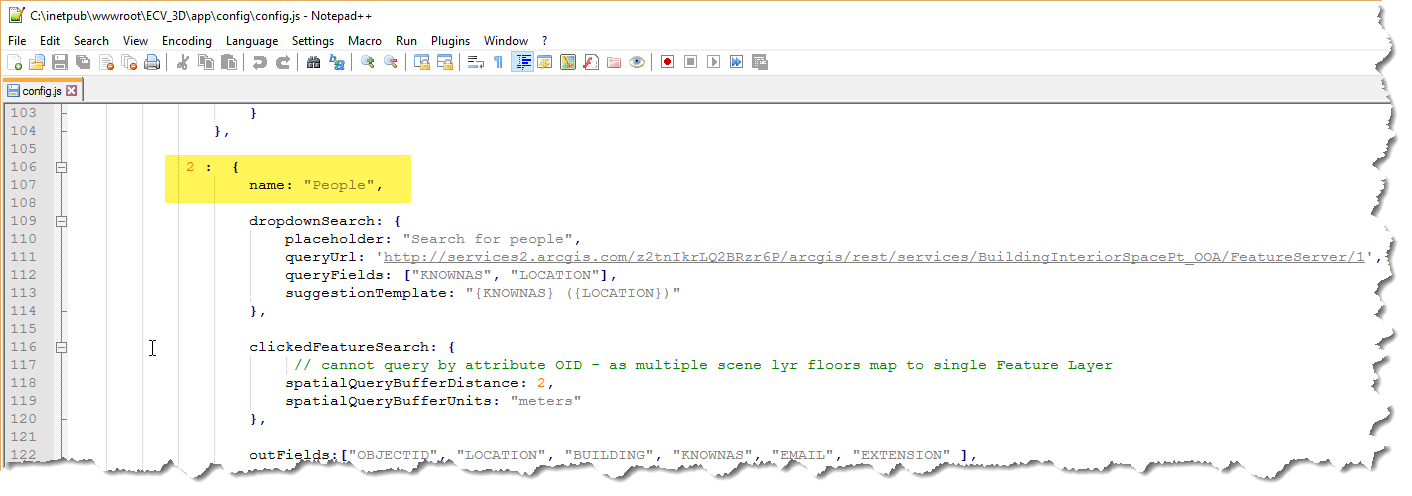
***HINT***: Make sure that the “groupLayerName” value explicitly matches the name of the group layer in the web scene. This value is case-sensitive.

* Setting up the “searchInfo” section of the app will configure the Search box for the app. This is where most of the work will be required, but it is also what enables a smooth search experience for users. The application allows for the searching of several resources at once. This can be very useful for enabling application users to search for someone by name, or for an asset by its ID number. For each search source, you will need to configure the following:
  + The name for the source
  + The information to display in the autoComplete drop-down list in the app
  + Search radius for supporting “Click to identify” in the scene view.
  + The output fields you want returned from the search.
  + The information for a “buddy” service or related table search.
  + The information to display in the blue side panel that presents information to the user. These setting are pointing to a default CSS class, but this is where you would change the CSS for this information if you desired. You could author the new class for the app and point to it here to change the look and feel of the presented information.
  + An associated scene layer if you want to change the scene look to better present query results.
  + Symbology information for highlighting returned features.



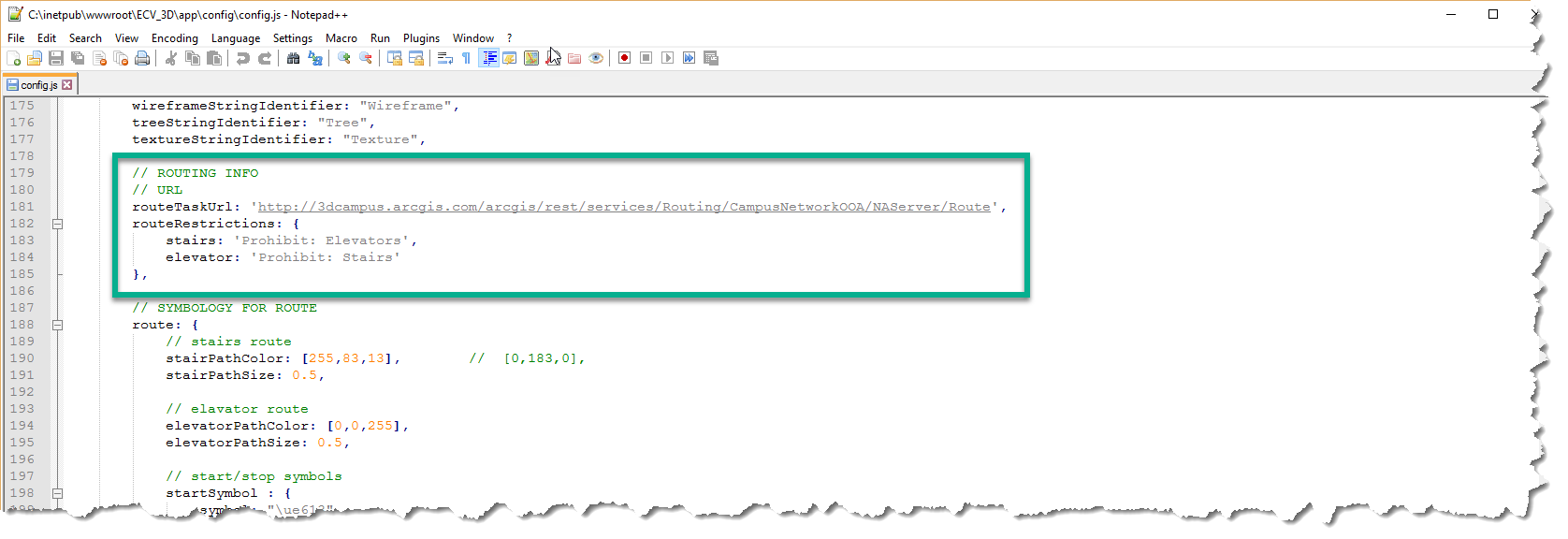
Hint: Pay close attention to the attribute values. It’s a best practice to copy and paste these values from the REST endpoint for the search service. In this case, you would copy and paste from the queryURL and from the RelationshipURL.

* The application supports multiple query sources in the search box. To add additional sources, simply add another value to the “searchInfo” object with an incremental number. There is a second search configured to query against the employee table in the config file.



Hint: Note how the queryURL and relationshipURL for the “people” search is just the inverse of the queryURL and relationshipURL for the “Rooms” search. This is intentional, as the 2 are a related table and it’s feature service, respectively. We want to be able to query a room and get the person in it from the related table, and also query the person from the related table and return what Room they sit in. This pattern can be used to link other related items - like work orders to assets.

* Configure the tip text for the search box in the “allSearchPlaceholder” value. This is the text that will show in the search box before the user enters a value.
* Set the naming values for the spaces and rooms to match your web scene values.
* Enter the route task URL that you have previously published. This enables routing from point to point using the network dataset and service you have put together. If this is on a portal server and is protected by authentication, it might be necessary to append a token value to the end of the URL.

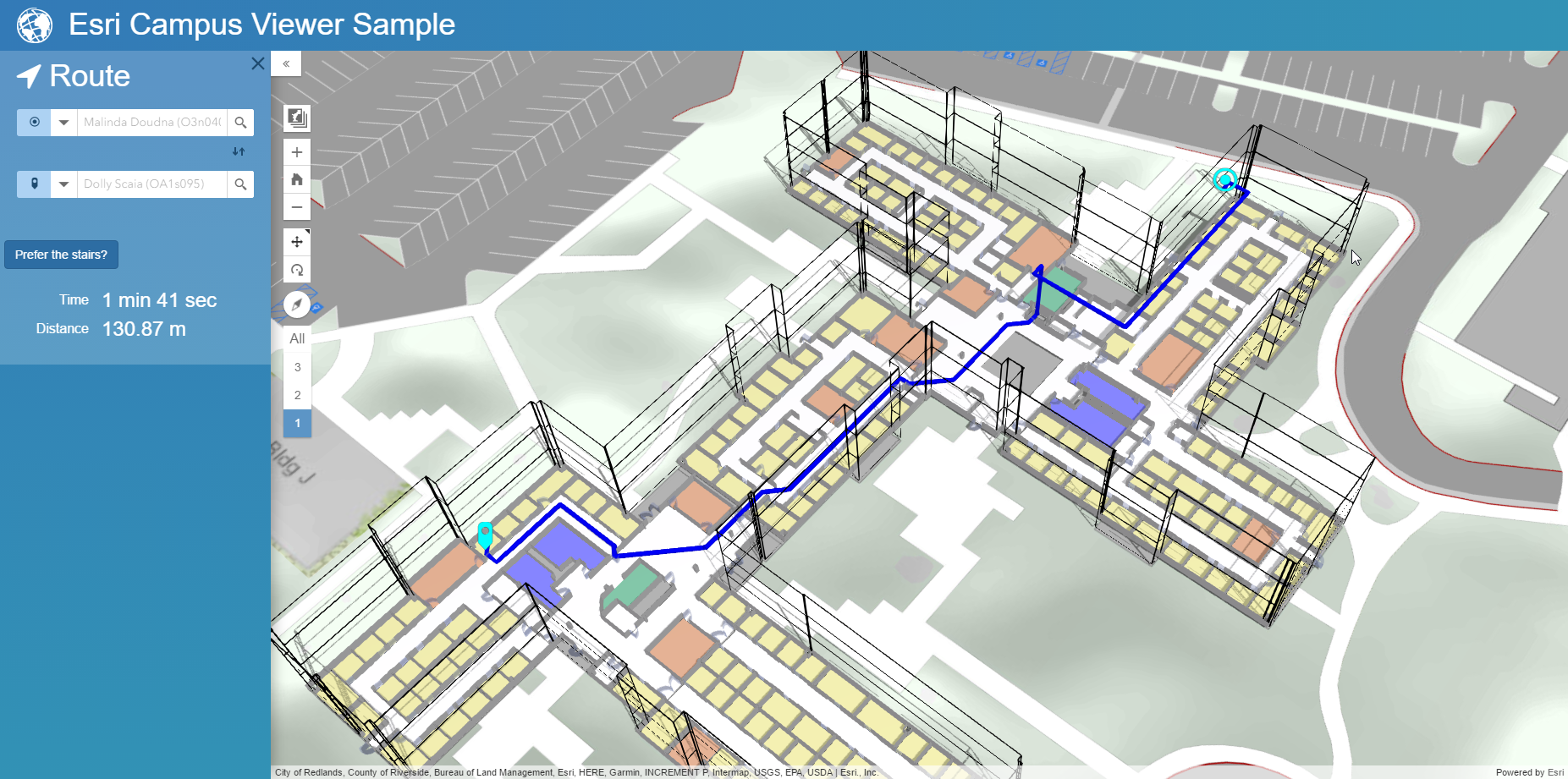


For the routeRestrictions values, you might have to check the endpoint URL for the routing service and make sure what is in the config file matches what is in the service. These values are case-sensitive.

**Note:** If you want to add additional restrictions to your own route, it will be necessary to change the application code. It is not enough to simply add or remove values in the configuration file to match what’s in your service.

* Configure the route symbology by changing the color and font values if desired.
* Change the zoom values for the search as desired.
* In the ‘renderersForWebSlide’ values, you can match up a custom web renederer to the view by name. For example, you can see the URLs to Feature Service layers mapped to specific views.

Once the settings have been changed to match your own data, save the config.js file and test the web app. You should be able to find records in your search URLs by either searching with the input boxes, or by clicking on the map. If this does not work right away, double-check the URL and attribute values in the config file, and be sure to clear your browser cache. Once a route is generated, you will be able to switch the route from using stairs (default) to using elevators. Testing routes in the app is a great way to make sure that your data is consistent.



***HINT:*** You are likely to find minor errors in the network service while routing in the app. Some examples would be stairways not being used or hallways being routed around. These are indicators of network inconsistencies. When you find them, make a note and work with appropriate colleagues to update the network dataset. Be sure that transition areas are snapped properly and that the pathways have consistent line features that snap to other pathways. It can also pay to double-check the attribution of transitions, in particular the “TYPE” attribute, which enables the stairs/elevator route prioritization.