VMSDK Application Development Guide: Using the VMD for Room Selection

Overview

The purpose of this document is to provide a quick overview of how a developer might use the VMSDK to implement a room selection use case for a custom app.

At a high level, the recommended application flow is as follows:

- Figure out which map/venue/hotel you want to show, then load that venue map data (VMD file)
- After map data is loaded, display map on screen, optionally adding your own custom styling for room colors, room labels, etc. (*More details on custom styling can be found in the Aegir VMSDK User Guide under Appendix: Vector Map Tile Style Spec*)
    - You will need to add your own custom UI controls for changing floors of the map. See the HTML class docs for VMMSMapBuilding included within the SDK "docs" folder for more details on how to query for available floors in the building.
    - You will need to add your own custom UI controls for filtering available rooms based on your own business requirements.
- Call the PMS API to query room availability & get room details
    - This could potentially happen at the same time as steps 1 and 2, depending on what info you need about the property to be able to query the PMS initially.
- For each available room, add custom map annotations that users will interact with
- Respond to user selecting map annotation for a specific room
    - Insert other custom UI screens where user confirms selection and additional PMS integrations happen

Code Samples

Load the sample map data for the venue you want (this does *not* display the map on screen)

```
NSString *zipFilePath = [[NSBundle mainBundle] pathForResource:VMD_ZIP_FILENAME
ofType:@"zip"];

id<VMDFile> localFile = [[VMDLocalZipFile alloc]
initWithAbsoluteFilePath:zipFilePath];

VMDFileCollection *fileCollection = [[VMDFileCollection alloc]
initWithBaseZipFile:localFile];

//parse the map - calls didFinishLoadingMap:customMapInfo: or
didFailToLoadMapWithError: if it fails
VMMSWaypointLabelOptions* options = [VMMSWaypointLabelOptions new];

[VMMSMap loadFromCollection:fileCollection delegate:self withOptions:options];
```

Implement the required delegate methods from the VMMSMapDelegate protocol so you can hook into when the map data has finished loading

```
- (void)didFinishLoadingMap:(VMMSMap *)map
              customMapInfo:(VMMSCustomMapInfo *)customMapInfo
{

//figure out what floor you want to display initially.. Probably the ground floor in
your venue
VMMSMapBuildingFloor *initialFloor = [[map getBuildingWithId:@"building_1"]
getFloorWithId:@"floor_b1_1"];

//create the map view and display it on screen
self.vmMapView = [[VMVectorMapView alloc] initWithVenueId:@"venue_map_sample"
                                          buildingNumber:1
                                          floor:initialFloor
                                          frame:self.baseMap.frame ];

//configure where your map tiles are coming from (these are included in the
/Assets/vector_tiles folder in the sample application)
self.vmMapView.tileBaseURL = [NSBundle.mainBundle.bundleURL
URLByAppendingPathComponent:@"vector_tiles"].absoluteString;

//additional map tile data for labels and icons (also included in the
/Assets/vector_common) folder in the sample application)
self.vmMapView.vectorCommonBaseURL = [NSBundle.mainBundle.bundleURL
URLByAppendingPathComponent:@"vector_common"].absoluteString;

//configure map initial floors, styling, and zoom levels
self.vmMapView.currentOutdoorFloor = initialOutdoorFloor;
self.vmMapView.maxZoom = 23.0;
self.vmMapView.delegate = self;
self.vmMapView.style = self.venueStyle;
self.vmMapView.map = self.map;

//this line adds the map to the view, *over* the specified "baseMap"
//You can pass in nil to add the map to the top of the view w/o requiring an
underlying map provider like Google Maps/Apple Maps/etc
[self.vmMapView attachInView:self.view :self.baseMap];
}
```

Call PMS API to find out what rooms are available

You may choose to show various loading screens in the UI while this interaction is happening.

For each available room (on the current floor), add a map annotation that will show custom-defined information about the room

```
let unit = self.vmMapView.currentFloor.getMapUnit(<the ID of the room defined in the
VMD>) //presumably there is some mapping between identifiers for rooms returned from
the PMS and those identifiers specified in the VMD

let marker = VMPointAnnotation();
marker.coordinate = unit.centerLocation;
marker.title = unit.uid;
marker.floorNumber = unit.floor.floorNumber;
marker.floorId = unit.floor.uid;
self.vmMapView.addAnnotation(marker);
```

It is up to the application developer to add/remove map annotations for available rooms as the user toggles between different floors & filter options so that only those annotations pertaining to the current floor & matching the current filter options are visible & selectable.

Customize the map annotation for each room by using the **VMMapViewDelegate** method **viewForPointAnnotation**

```
func viewForPointAnnotation(_ annotation: VMPointAnnotation) -> UIView? {
{
       //we used the ‘title’ property to store the uid of the map unit
       let unit = self.vmMapView.currentFloor.getMapUnit(annotation.title!)

      //figure out the room details for this room when you previously queried the PMS
for available rooms
       //create a UIView object that displays that information.
       //for example, you might display the price difference of the room from the
reservation price using a UILabel

      //OR: You might also keep track of the currently selected map unit, and show a
different view for the currently selected room

      let label = UILabel();
      label.text = “-$123”// the price difference of the room
      // continue configuring the label size/font properties etc
     return label;
}
```

Add custom behavior when the user selects one of the available rooms by using the **VMMapViewDelegate** method **didSelectAnnotation**

```
func didSelectAnnotation(_ annotation: VMMapAnnotation)
{
        if let mapUnit = self.vmMapView.currentFloor.getMapUnit( annotation.title )
        {
            // add your custom application logic here to advance the user to a room
detail screen, for example, to show additional information about the room and/or
confirm booking
        }
 }
```