

How to call Spectrum workflow from a custom component

Version 12.2

Contents

[Prerequisites 1](#_Toc517711407)

[Introduction 1](#_Toc517711408)

[How to call Spectrum workflow from Custom Analyst component 2](#_Toc517711409)

[Notes 9](#_Toc517711410)

[Tips to debug the code. 9](#_Toc517711411)

## Prerequisites

* Basic knowledge of Angular 2/4 and Openlayer.
* Ability to code in Typescript
* Basic understanding of SSA terminologies (Mapconfig, LoggedInUser, Annotation)
* Understanding of spectrum workflow and permission model of it
* Basic understanding of GIS/Web Mapping
* All the sample code needs to be put under <SSA\_INSTALL\_PATH\ customerconfigurations\analyst\theme\extensions\app\spectrum-workflow>
* You need to login as an admin to make a call to Spectrum workflow or assign a permission to execute workflow from spectrum manager to AnalystGuestRole.

**Time duration to complete: 40 minutes**

## Introduction

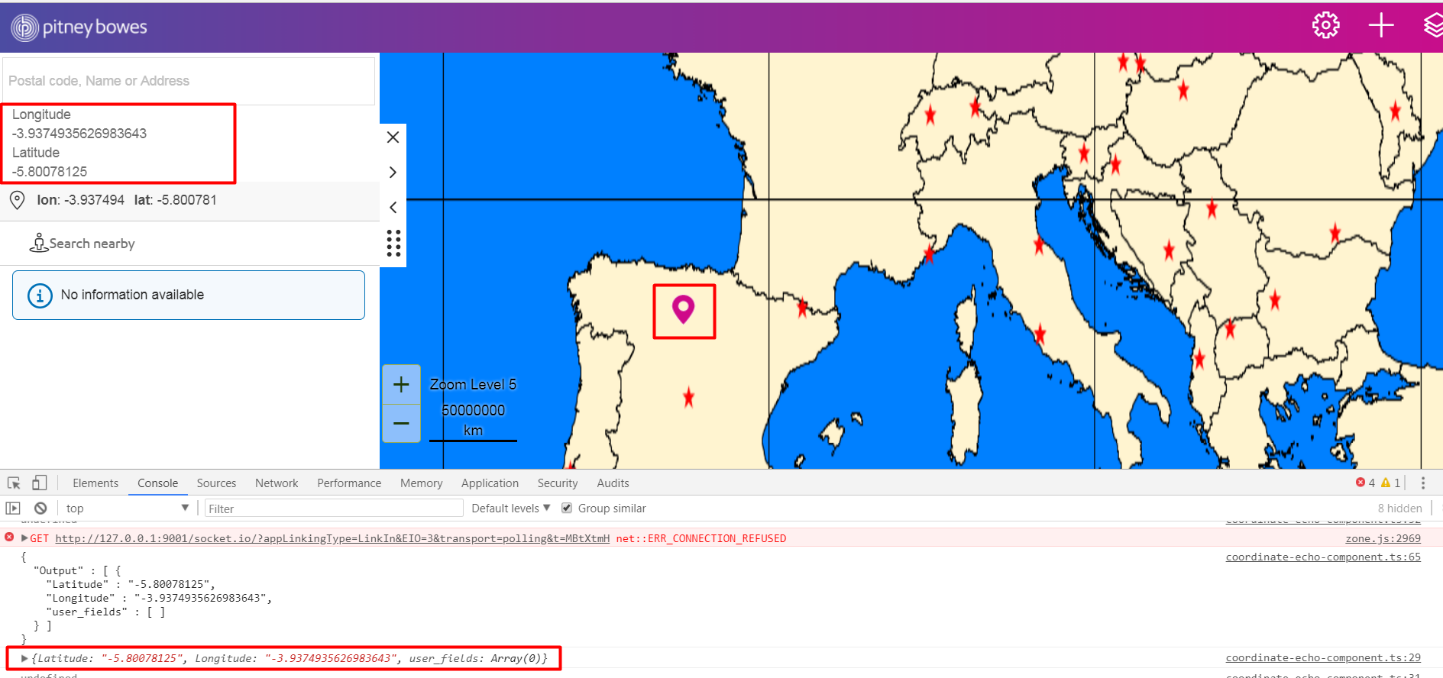
This example will create a new component using SSA’s extensibility APIs which will call a Spectrum workflow which we expose as a REST service.

In this example we are going to demonstrate three things.

* How to use XhrInvokeService to make http post call
* How to add a call to the Spectrum data flow REST service via the connect proxy
* How to use the left panel to display information received via workflow processing.

By the end of this exercise you will see below things on your screen as an outcome when you click on the map.

Please ensure that the DF file for the workflow is deployed into Spectrum by following the detailed notes on part 1 below



## How to call Spectrum workflow from Custom Analyst component

There are two parts to this exercise

In the first part you will create a custom workflow in Spectrum via the Spectrum Enterprise Designer and expose it as REST service. We have provided a sample workflow as DF file that can be used. The workflow provided with this example is just an “echo service” that simply takes takes the coordinates in terms of XY (or lon/lat) and returns the same coordinates back to the caller.

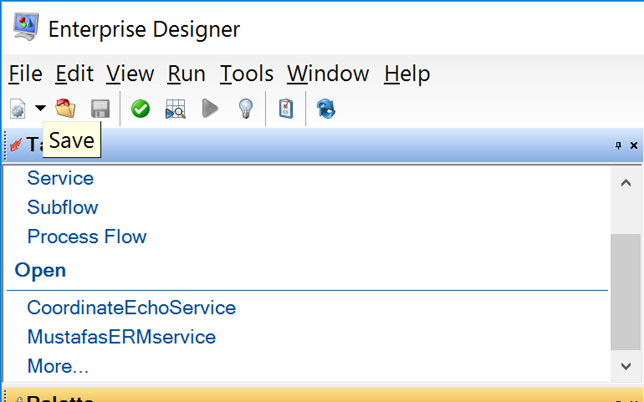
The second part is the new SSA component that will call the exposed REST API for the workflow. Here we will use the connect proxy to call Spectrum via a HTTP POST request.

Note that unless you have granted permission for your users or roles to execute a data flow (in the Spectrum Management Console), only admins will be able to run this sample. Please ensure you are first logged in as an administrator in SSA. If not you will see a service error in SSA like this instead of seeing the XY shown in the left panel.

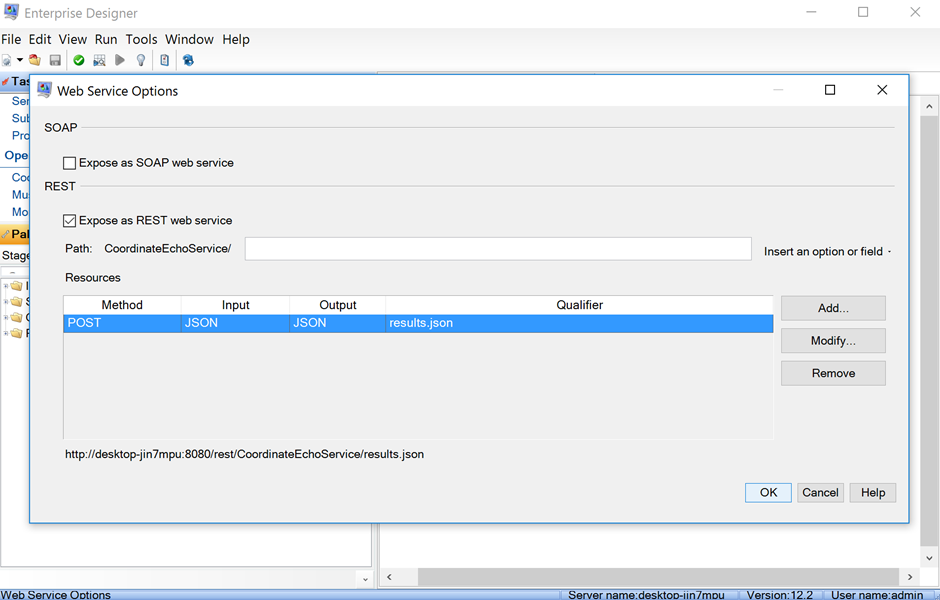
**Part 1: setting up the data flow**

You need to download the Spectrum Enterprise Designer from Spectrums landing page and installed it. After running and logging into Spectrum Enterprise Designer please follow these steps.

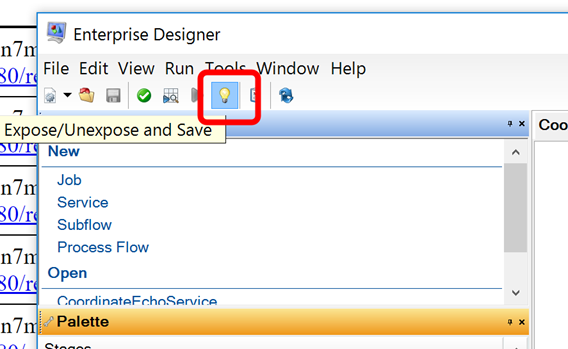
1. On the file menu choose File -> Import -> Dataflow and browse to the DF file supplied with this example.
2. Next save the data flow using the save button



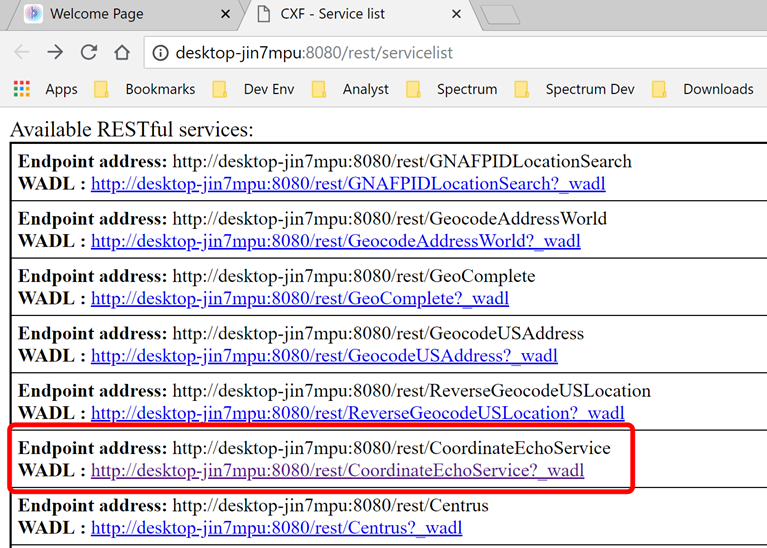
1. In the menu choose Edit -> Web Service Options. Ensure that RST is ticked and SOAP can be uncticked.



1. Then expose the dataflow as a REST service by clicking the bulb icon



1. You can see if this ws exposed by looking at the REST service page. Below we see the service has been exposed OK.



**Part 2: Example**

coordinate-echo.component.ts

import { Directive, Inject, OnDestroy, OnInit, EventEmitter, Output, Input, Component, ViewChild, ElementRef } from '@angular/core';

import { Observable } from 'Rxjs/Observable';

import { Store } from '@ngrx/store';

import { State } from 'src/app/common/store/states';

import \* as fromRoot from 'src/app/common/store/reducers/index';

import { Subscription } from 'rxjs/Subscription';

import { DomSanitizer } from '@angular/platform-browser';

import { MapUtilityService } from 'src/app/pb-maps/services/map-utility.service';

import { WeatherSharedService } from './weather-shared-service.ts';

import { RequestMethod } from '@angular/http';

import { XhrInvokeService } from 'src/app/common/services/xhr-invoke.service';

@Component({

selector: 'coordinate-echo-component',

template: `

<div class="col-xs-12">

<div [innerHtml]="myVal"></div>

</div>

`

})

export class CoordinateEchoComponent implements OnInit, OnDestroy {

private myVal: any;

constructor(private store: Store<State>, private mapUtilityService: MapUtilityService, private sanitizer: DomSanitizer,

private xhrInvokeService: XhrInvokeService) {

this.store.select(fromRoot.getLocatorMarkerInfo).subscribe((info) => {

this.rDSURL(info.markerInfo.location.x,info.markerInfo.location.y).subscribe(data => {

console.log(data.Output[0]);

                const lonlat = data.Output[0];

                console.log(lonlat.longitude);

                console.log(lonlat.latitute);

this.myVal = `<div>

                                <div style='display:"inline:block"'>longitude</div>

                                <div>`+lonlat.longitude + `</div>

                                <div style='display:"inline:block"'>latitude</div>

                                <div>`+lonlat.latitute +`</div>

                             </div>`;

});

});

}

ngOnInit() {

}

private getRequestForSpectrumWorkflow(lon: string, lat: string) {

return {

"url": '../controller/connectProxy/rest/CoordinateEchoService',

"method": RequestMethod.Post,

"params": {

"url": "results.json",

"postData":

{ "Input": { "Row": [{ "lon": + "" + lon, "lat":""+ lat }] } }

}

}

}

rDSURL(param1, param2): Observable {

const options = this.getRequestForSpectrumWorkflow(param1, param2);

return this.xhrInvokeService.invokeControllerAPI(options).map(function (data) {

console.log(data);

return JSON.parse(data);

});

}

}

Create an CoordinateEchoComponent class

1. Provide dependencies via import which are required

import { Directive, Inject, OnDestroy, OnInit, EventEmitter, Output, Input, Component, ViewChild, ElementRef } from '@angular/core';

import { Observable } from 'Rxjs/Observable';

import { Store } from '@ngrx/store';

import { State } from 'src/app/common/store/states';

import \* as fromRoot from 'src/app/common/store/reducers/index';

import { Subscription } from 'rxjs/Subscription';

import { DomSanitizer } from '@angular/platform-browser';

import { MapUtilityService } from 'src/app/pb-maps/services/map-utility.service';

import { WeatherSharedService } from './weather-shared-service.ts';

import { RequestMethod } from '@angular/http';

import { XhrInvokeService } from 'src/app/common/services/xhr-invoke.service';

1. Provide template @Component metatag where <coordinate**-echo**> component

template: `

<div class="col-xs-12">

<div [innerHtml]="myVal"></div>

</div>

`

1. Add a constructor as mentioned below where we are subscribing to a mapclick event with the help of selector of map click via ngrx store instance. Once the coordinates of map click are received we are calling a method named rDSURL that we are going to add in next step.

constructor(private store: Store<State>, private mapUtilityService: MapUtilityService, private sanitizer: DomSanitizer,

private xhrInvokeService: XhrInvokeService) {

this.store.select(fromRoot.getLocatorMarkerInfo).subscribe((info) => {

this.rDSURL(info.markerInfo.location.x,info.markerInfo.location.y).subscribe(data => {

console.log(data.Output[0]);

                const lonlat = data.Output[0];

                console.log(lonlat.Longitude);

                console.log(lonlat.Latitute);

this.myVal = `<div>

                                <div style='display:"inline:block"'>longitude</div>

                                <div>`+lonlat.Longitude + `</div>

                                <div style='display:"inline:block"'>latitude</div>

                                <div>`+lonlat.Latitude +`</div>

                             </div>`;

});

});

}

1. Add this method to the component that returns an observable that is subscribed in constructor. This method is making use of xhrInvokeService to make and http request call to spectrum workflow. rDSURL is making use of getRequestForSpectrumWorkflow to create input for xhrInvokeService.

rDSURL(param1, param2): Observable {

const options = this.getRequestForSpectrumWorkflow(param1, param2);

return this.xhrInvokeService.invokeControllerAPI(options).map(function (data) {

console.log(data);

return JSON.parse(data);

});

}

1. Now add a method mentioned in below code snippet, pay attention to the endpoint url we are hitting CoordinateEchoService is the spectru spatial end point is exposing Spectrum workflow functionality. Everything before that is endpoint address of Connect proxy exposed in Analyst. Spectrum workflow expects Input in the format as mentioned against postData in json.

private getRequestForSpectrumWorkflow(lon: string, lat: string) {

return {

"url": '../controller/connectProxy/rest/CoordinateEchoService',

"method": RequestMethod.Post,

"params": {

"url": "results.json",

"postData":

{ "Input": { "Row": [{ "lon": lon, "lat": lat }] } }

}

}

}

1. Now configure the CustomAnalystModuleConfig.json which should be inside customerconfigurations\analyst

{

"modules": [

                {

"name": "Spectrum workflow sample",

"description": "Weather Information",

"modulePath": "extensions/app/spectrum-workflow/spectrum-workflow.module.ts#SpectrumWorkflowModule",

"components": [

{

"componentName": "CoordinateEchoComponent",

"parentComponentName": "LeftPanelContainer"

}

],

"mapConfigAssociated": [

{

"mapConfigName": "defaultmap",

"components": [

"CoordinateEchoComponent"

]

}

]

}

        ],

"componentsToRemove": []

}

1. Make sure the component name in CustomAnalystModuleConfig.json file should be same as the name of the class of your component
2. Now go to the browser and type in URL of SSA in it. If it is already opened just refresh the page by hitting F5. You should find this component (as highlighted in above image) under add “ + ” section of SSA.

## Notes

This example will only work for the spectrum workflow hosted on same spectrum instance that Analyst is refering for its data. In cases where user wants to call workflow from different instance of Spectrum component developer needs to make CORS request to Spectrum instance passing is full url to Spectrum including host name and port. Component developer needs to make use of http module comes with Angular as XhrInvokeService is tightly coupled to Analyst backend controller.We use ng-rx store in Analyst platform as a standard to do communicate between our own components. In order to consume any of Analyst capabilities either you need to subscribe to selector of store or dispatch and action on store. In any case if there is no action available suitable for your use case, you may want to make use of services exposed by analyst like SpectrumSpatialQueryService.

## Tips to debug the code.

Your module is standard typescript module, you can use all the debugging capability of typescript there. Since you are developing in a platform you can use console logs for debugging purpose. Some of the common errors you will encounter may be something like

1. Module not found.

This is a standard error due to bad configuration in CustomAnalystModuleConfig json file. Make sure that path you mentioned in the config file is correct along with correct file name of the module. Remember it is a module file path not component file path and relative to index.html file in connect\mobile folder of Analyst deployment.

1. Unable to resolve service

This error occur for the cases when you have not imported your service/component correctly in custom component. Make sure the path you give include src at the beginning of path.

1. Unable to resolve component

This error occurs if you have started using the modules component but have not added module to import definition in the module file. For any component outside current module you need to import its module in module definition file. For example PbAnnotationsModule is imported in ExtendedAnnotationToolModule.