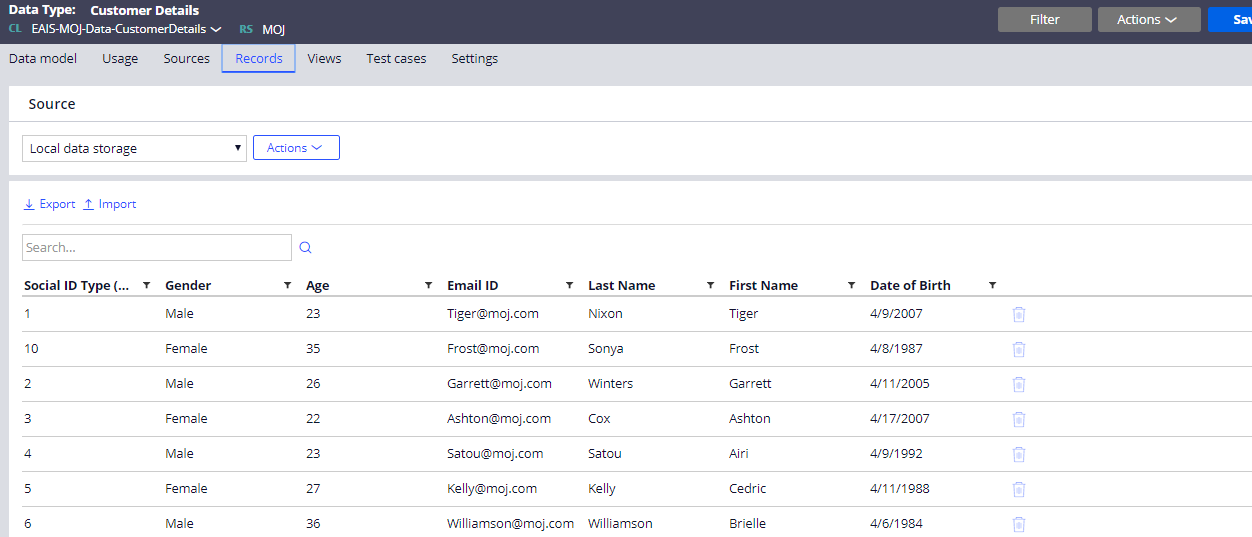
**How to expose REST Service from Pega**

**Requirement**:

* Create a data type in your application which has  10 data instances
* This data needs to be exposed as Service-REST to the outside world.
* When user passes SocialIDType via request to Pega Service, we need to parse the request and open that record (which is an instance of below Data type) and send that record details in JSON as response to the client.

**Note: Pega don’t provide any wizard to create Service-REST rule.**

**First create a data type as Customer Details and insert 10 records:**

****

Now we are good with the data, and we can proceed to configuration of hosting a Pega Service.

**Step 1**: First we need to create Integration class structure manually.

1. One main Integration class to hold the service rules.
2. Other classes to support request and response messages.

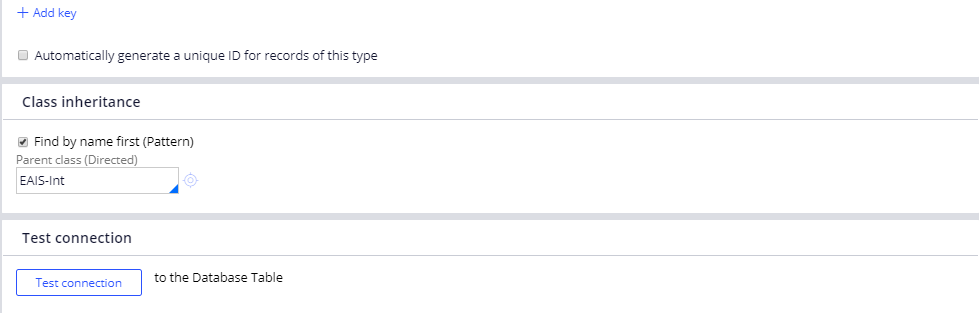
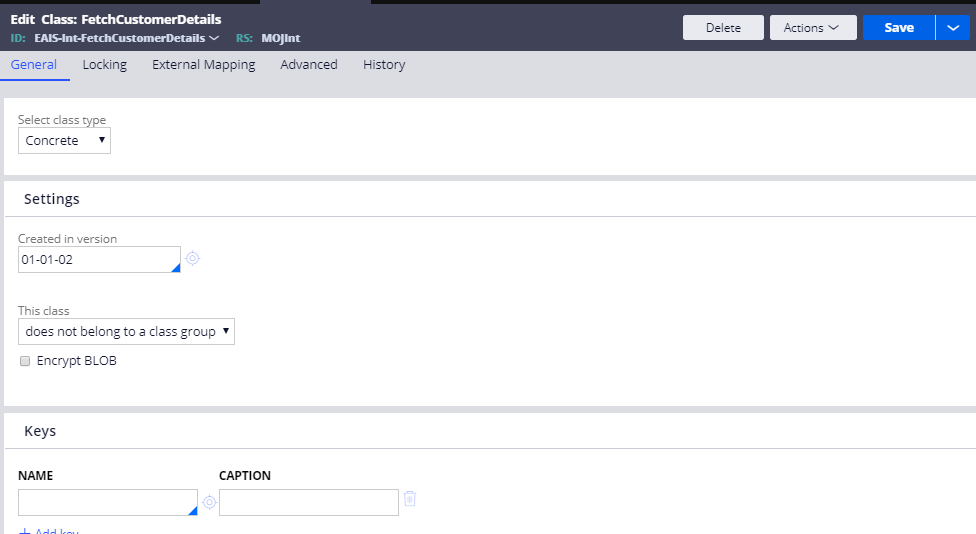
In our example,

**Request** – SocialIDType – Single value, no need to create any class structure.

**Response** – Complex Structure, we need to create a class for Response.

Main Integration class:

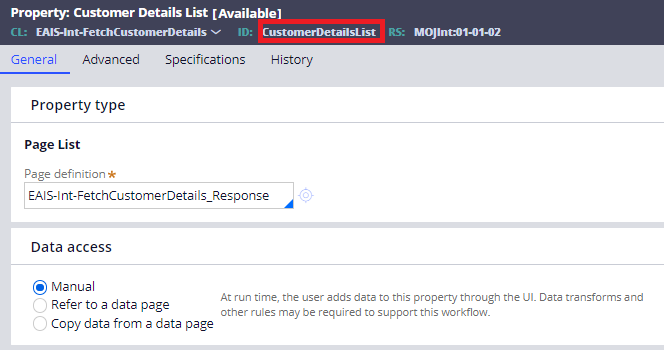
EAIS-Int-FetchCustomerDetails



**Step 2**: Create the request and response properties:

Create the below properties inside this class:

1. SocialIDType -- For Request
2. CustomerDetailsList of type page list -- For Response

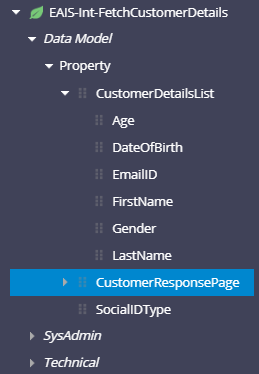


For creating the above property, we need to create a data class as “EAIS-Int-FetchCustomerDetails\_Response”

Create the below properties in the “EAIS-Int-FetchCustomerDetails\_Response” class:

1. Age
2. DateOfBirth
3. EmailID
4. FirstName
5. Gender
6. LastName

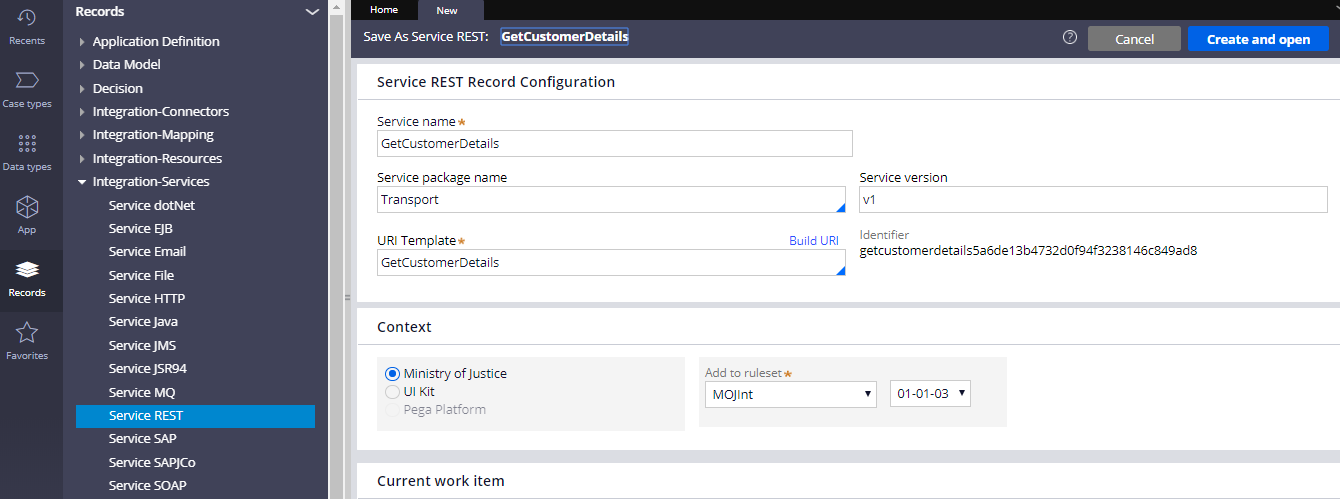
Our Data model structure will look like below:



Now our class and property structure is ready.

**Step 3**: Create Service-REST rule.

It is a part of the Integration-Services Category



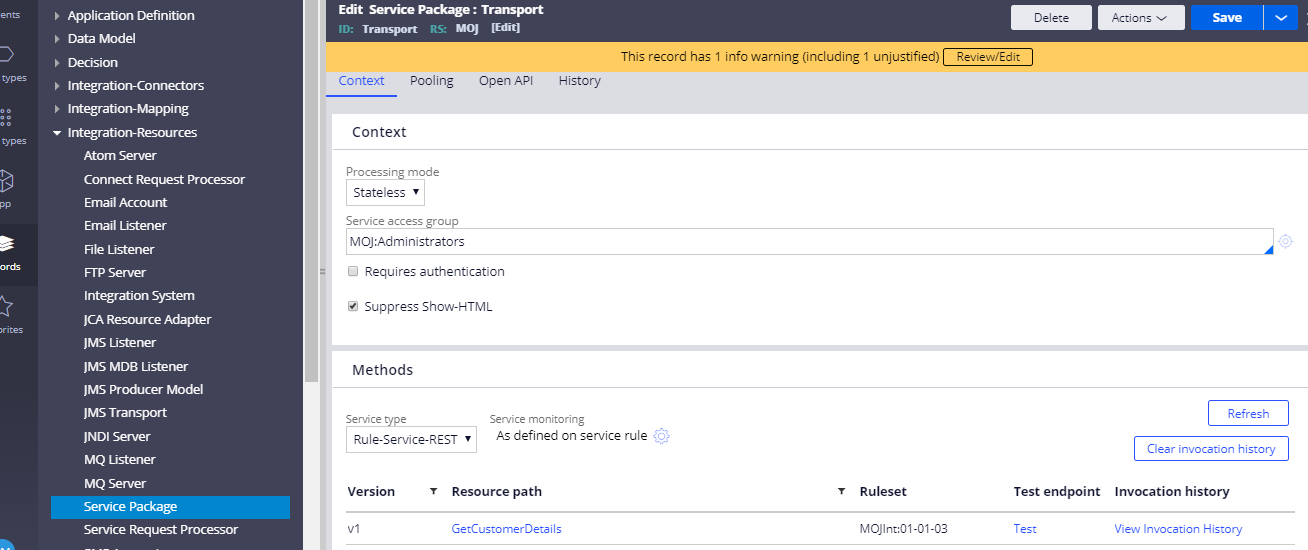
Create a service package which will store this service.

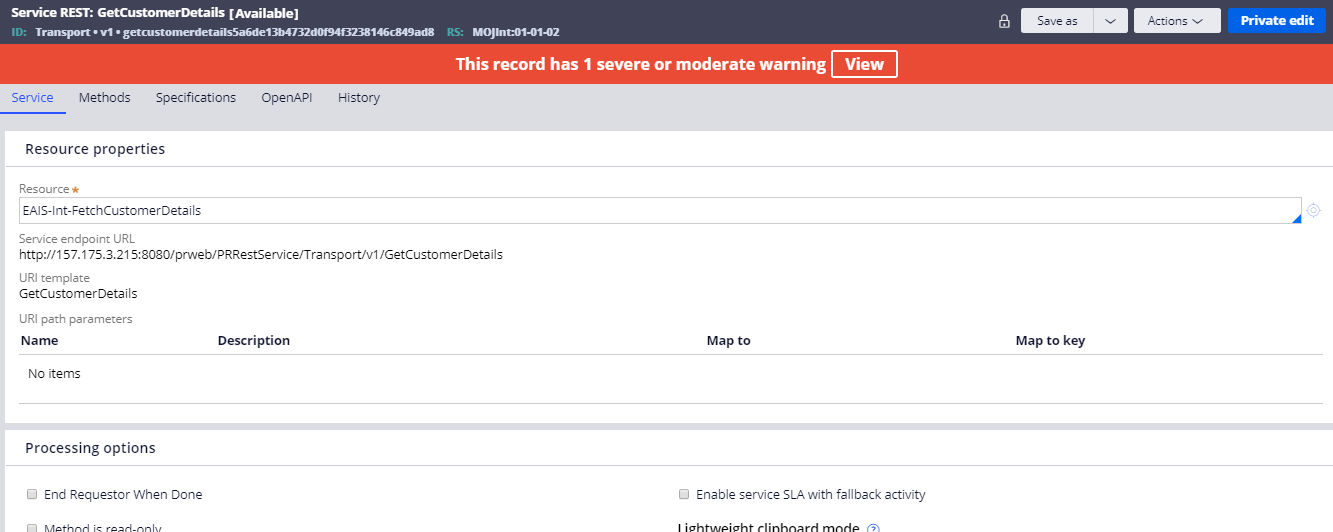
In this example service package name is “Transport”.

Service packages are data instances that support service processing in Pega. They decide the processing mode as either stateful or stateless. It provides security by enabling authentication for the service. The client should send the authorization parameters to get serviced

It is a part of Integration-Resources category.

In service access group, provide the access group of your application.





In the Resource field, provide the main integration class which we have created before. In this example the class name is **EAIS-Int-FetchCustomerDetails.**

There are two main tabs:

1. Service tab
2. Methods tab

Methods Tab:



**Get** – Get all the data from the database

**Post** – Create a new record in the database

**Put** – Update a record in the database

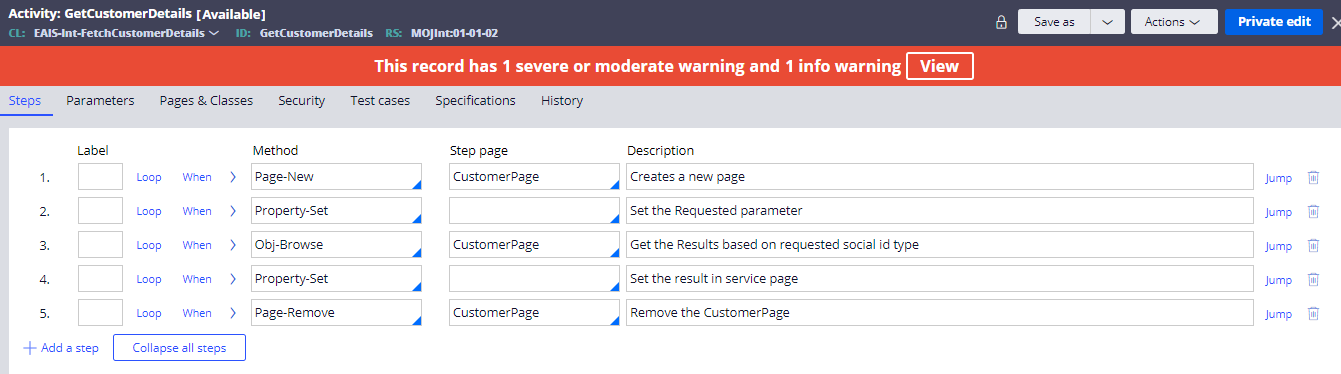
**Patch** – Update a record in the database

**Delete** – Delete a record from the database.

For our requirement we will make use of “Get” Method.

**Step 4:** Create a new service activity – We will handle all the processing logic in the service activity.

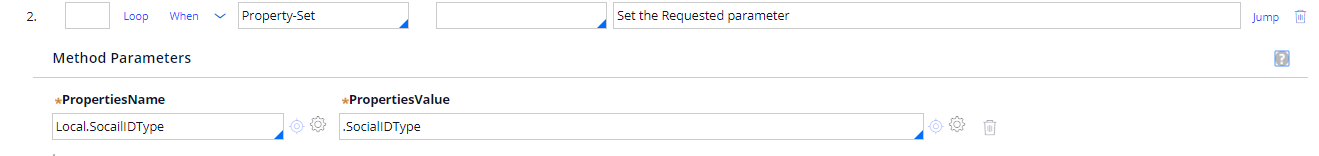
We will create the activity in the main integration class which we have created as “EAIS-Int-FetchCustomerDetails”



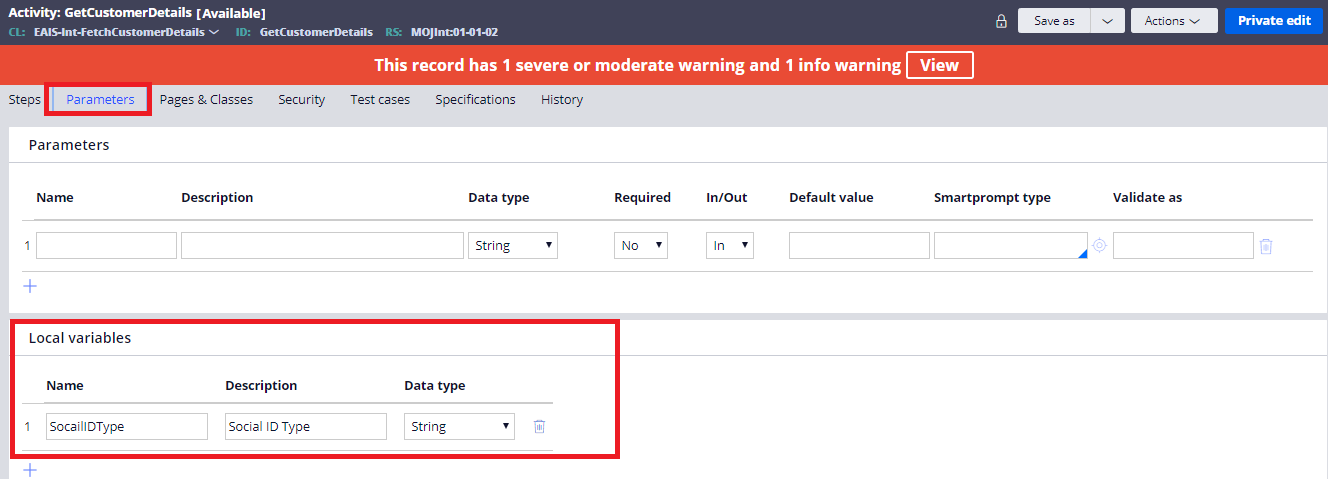
We will do the below steps in the activity:

1. Page-New – This will create a new page as “CustomerPage” in this page we will store the response in this page.
2. Property-Set – This is used to set the request parameter which will entered by user if he wants to get the customer details as per the SocialIDType.

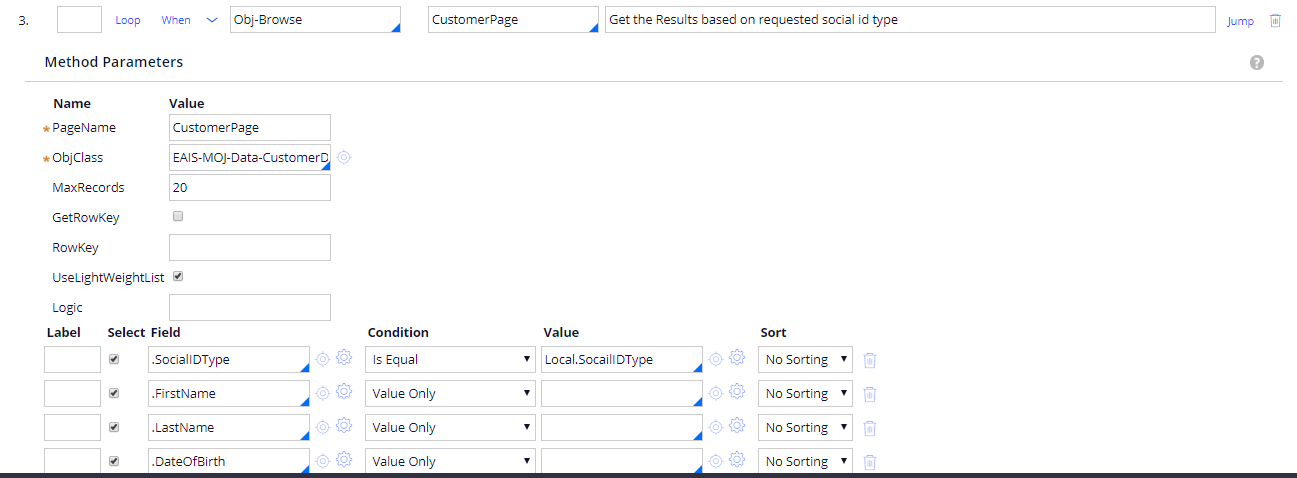
Local.SocailIDType = .SocialIDType



Specify the local variable in parameters tab of the activity:



1. In the third step, use obj-browse method which will fetch the customer details from the table.



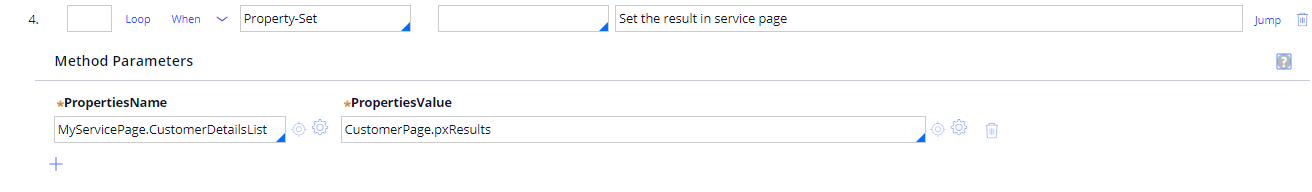
In the first condition of obj-browse method put the logic as

.SocialIDType = Local.SocailIDType

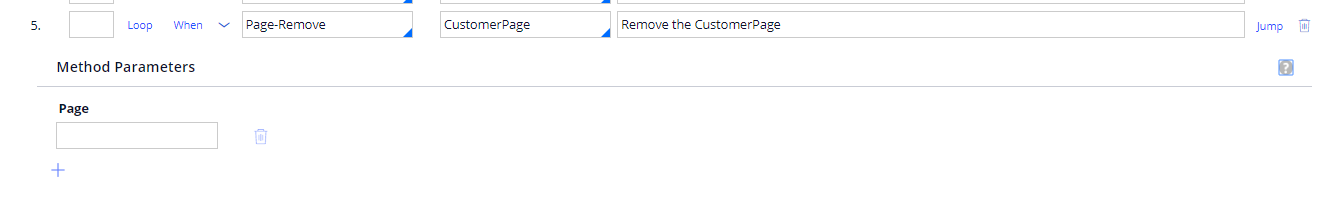
Local.SocailIDType stores the value entered by the user.

Mention all the fields in the Field value which we want to fetch from the table.

1. In the step 4, copy the results from the “CustomerPage.pxResults” to service page “MyServicePage.CustomerDetailsList” by using property-set method.

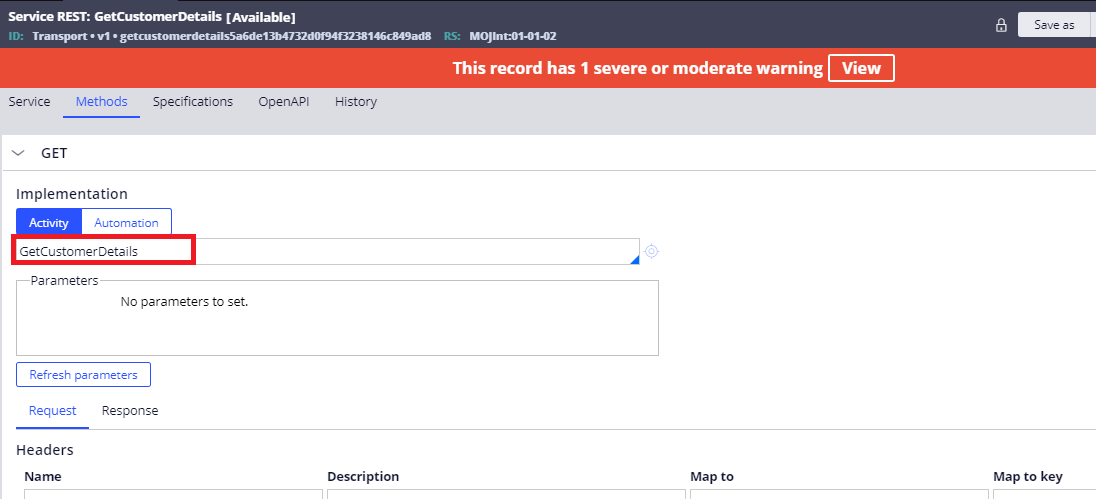


1. Finally remove the unwanted pages from the clipboard by using page-remove method.



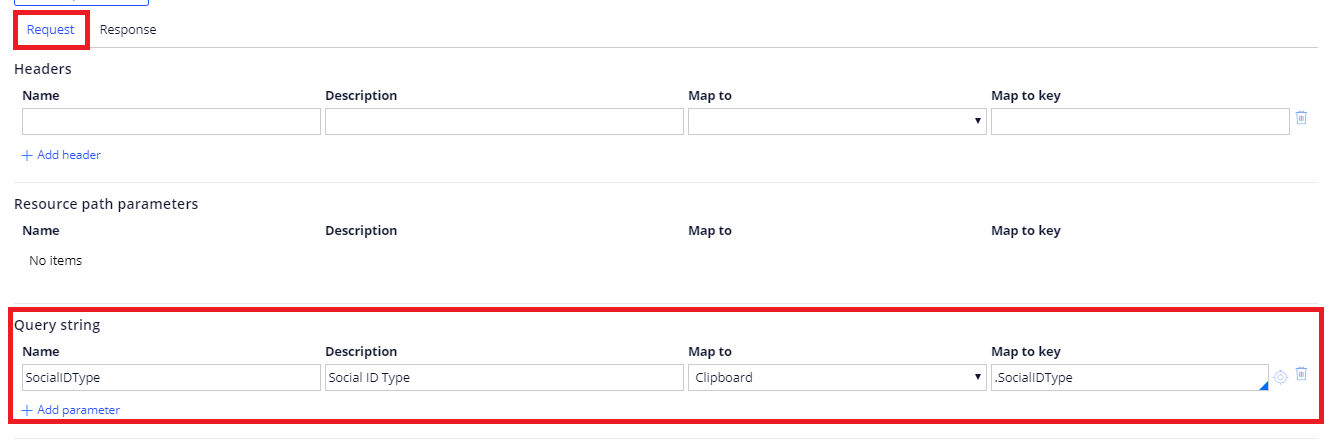
We will now call this activity from the method tab of the service-REST rule.

We will use the Get Method here.

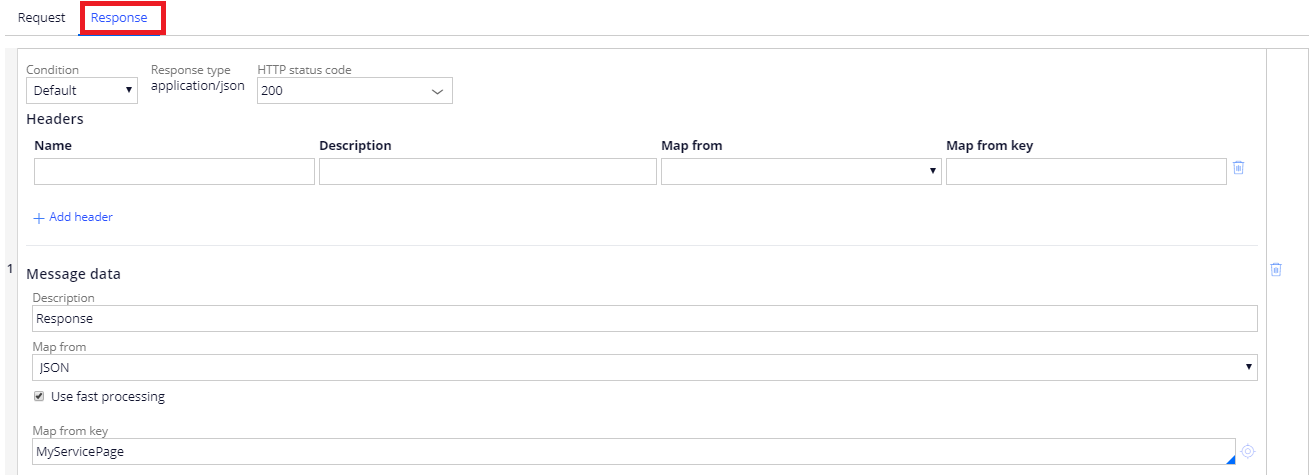


After specifying the activity, scroll down and in the Request specify the Query String as “SocialIDType” which will be mapped to clipboard.

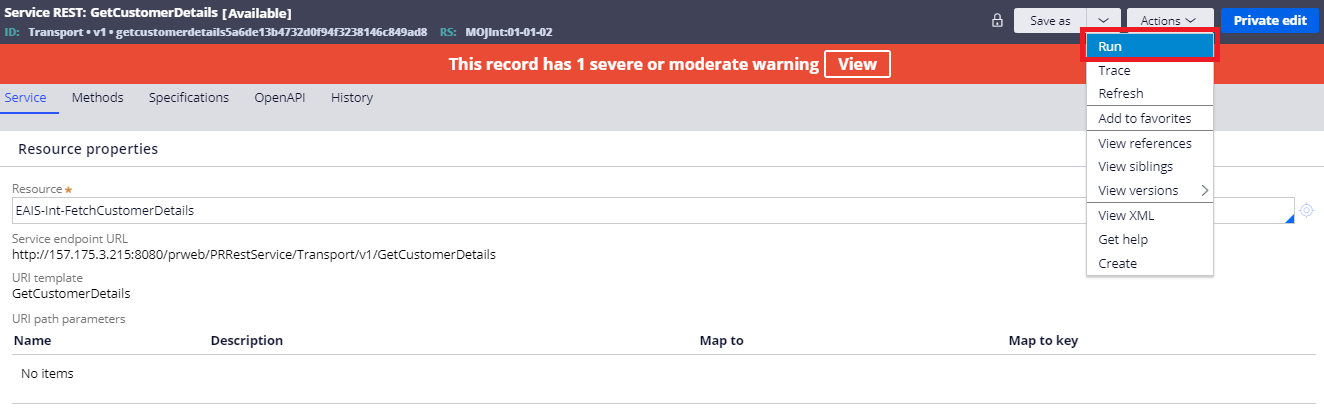
Please see the below screenshot:



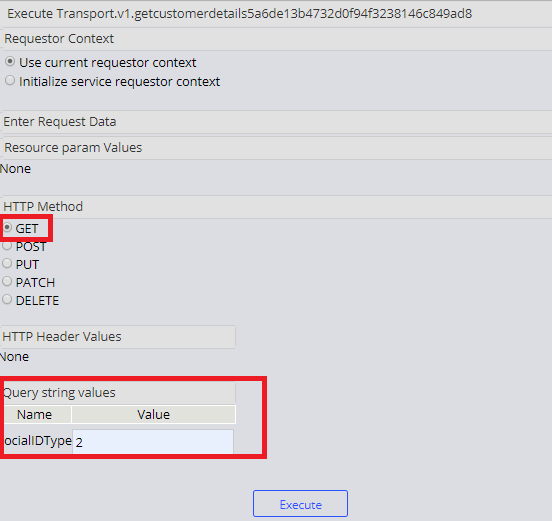
Now go the Response Tab for mapping Response properties:



Once this done, save the service-REST rule and test it by using a Run menu from the Action button.



Select the Get Method and give the value of the SocialIDType of the customer.

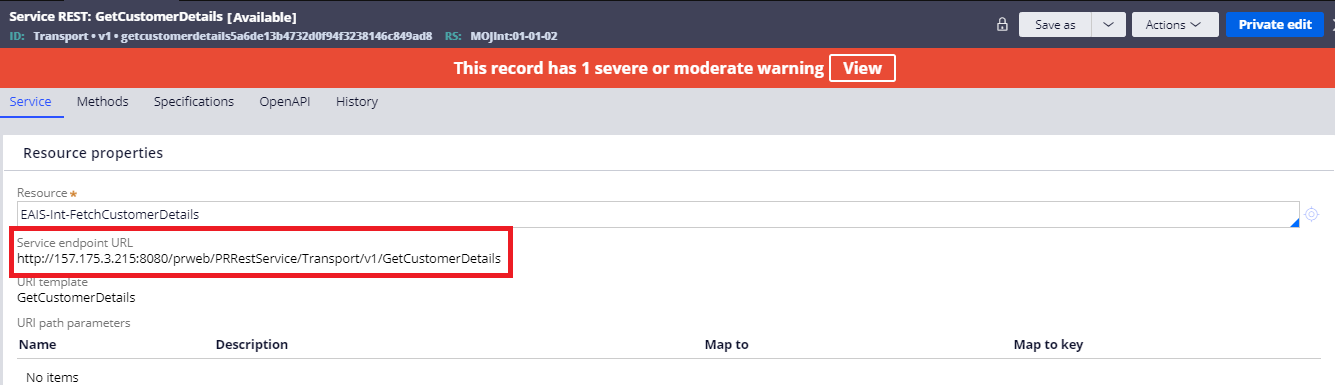


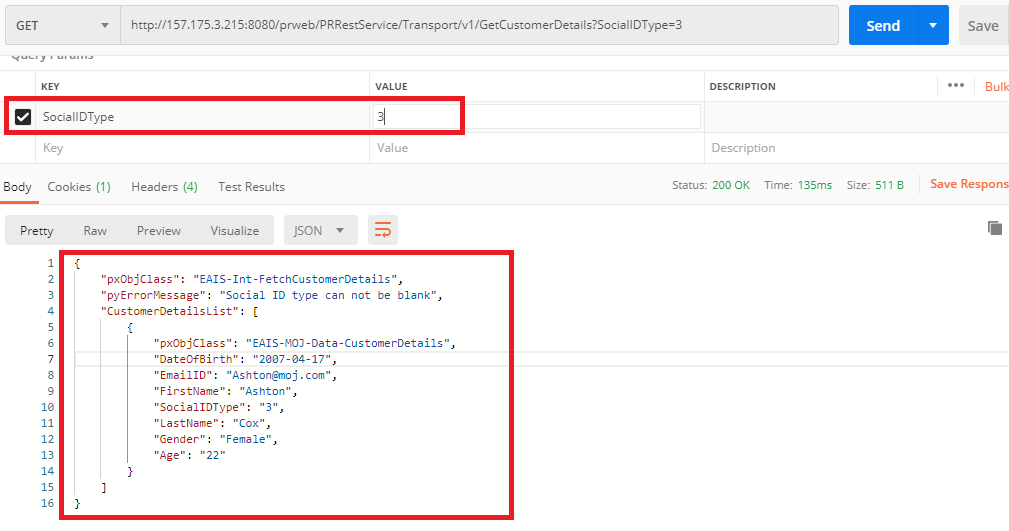
When we click on “Execute” button, we will get the customer details whose SocialIDType is 2, we will get the result in the JSON format.



Another way of testing this service is through Post Man.

We will copy the url present in the service-REST rule and paste in the post man





Our service-REST is working as expected.

**Now we will see how to configure Connect-REST rule**

1. It comes under Integration-Connectors category
2. This rule helps in implementing connectors with other application using REST.

Before configuring a Connect-REST rule, there are some prerequisites to collect from service provider.

1. The Service URL
2. Request and Response parameters
3. HTTP method, they host the service.
4. Authentication, if any

We have created a service API right?! We are going to use the same service API as service provider.

<http://157.175.3.215:8080/prweb/PRRestService/Transport/v1/GetCustomerDetails>

Request – SocialIDType

Response –

{

    "pxObjClass": "EAIS-Int-FetchCustomerDetails",

    "pyErrorMessage": "Social ID type can not be blank",

    "CustomerDetailsList": [

        {

            "pxObjClass": "EAIS-MOJ-Data-CustomerDetails",

            "DateOfBirth": "2007-04-17",

            "EmailID": "Ashton@moj.com",

            "FirstName": "Ashton",

            "SocialIDType": "3",

            "LastName": "Cox",

            "Gender": "Female",

            "Age": "22"

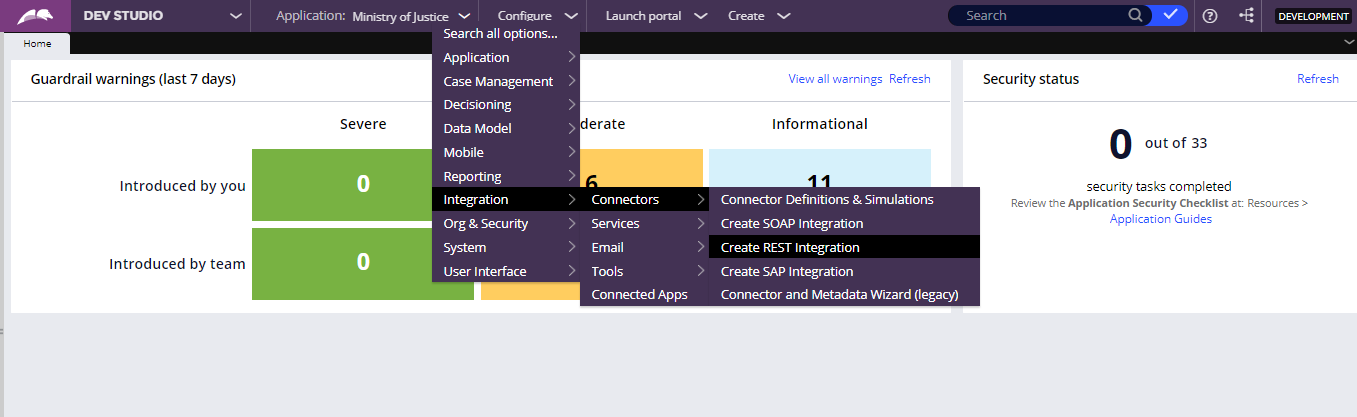
        }

    ]

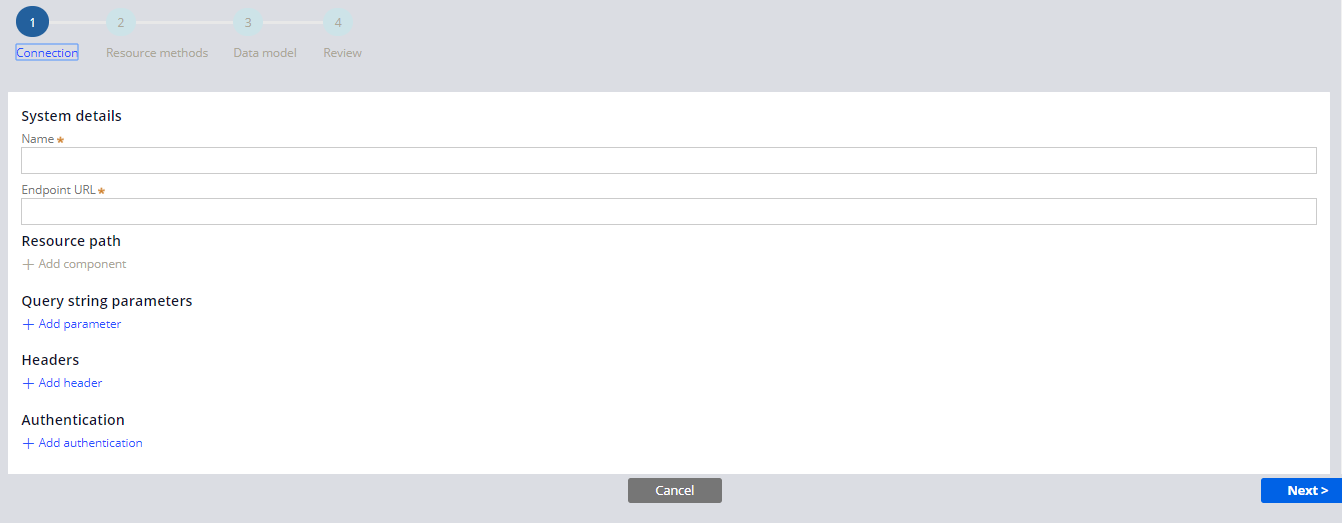
}

All the three basic information are handy, Let’s straightly jump to configuring Connect-REST.

Pega provides a wizard to connect REST integration.

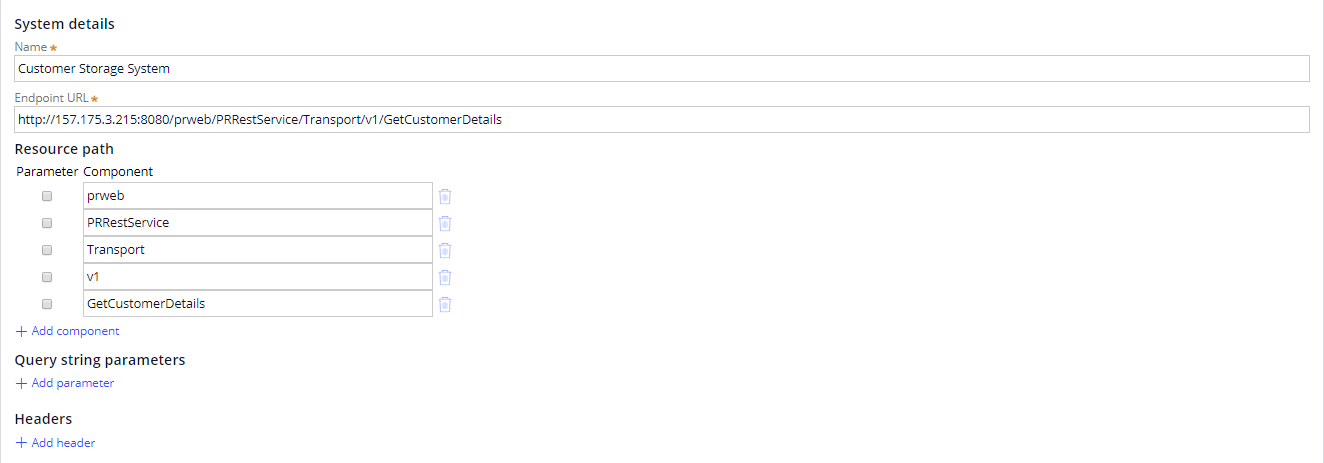


This is basically a 4 step screen flow:



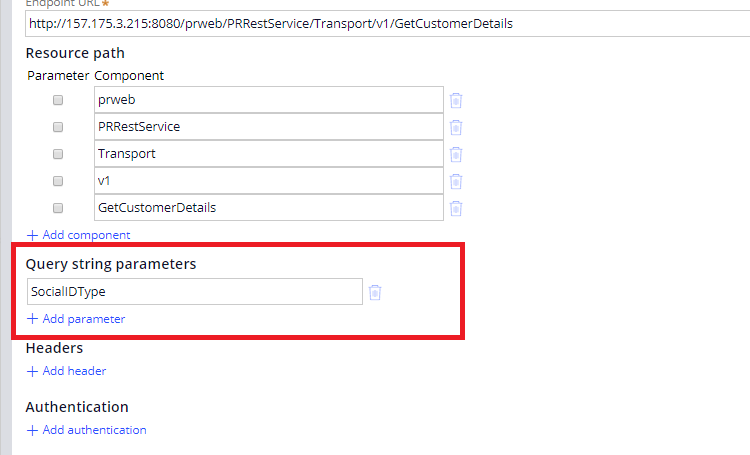
**Name** – Enter the name of the system, which we are going to connect to consume the service.

**EndPoint URL** – Enter the complete url of the service.



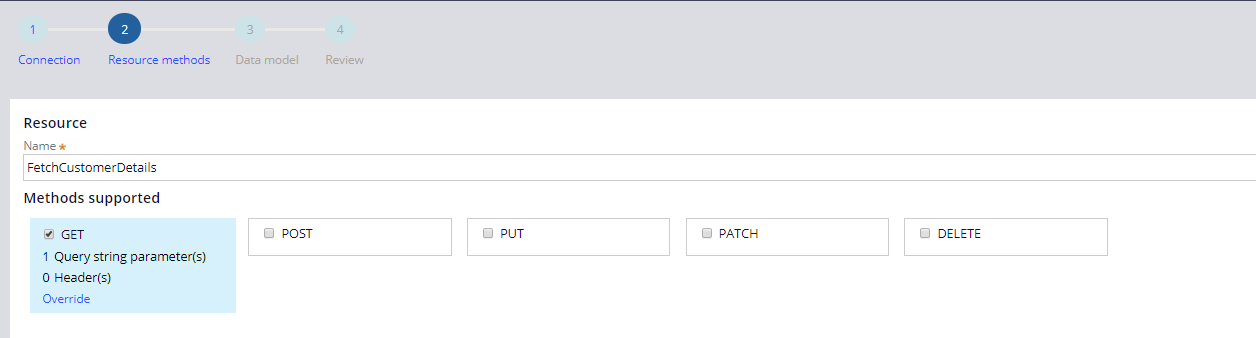
If we enter the complete url path, then resource path will be pre-populated.

We can specify the request as query string – SocialIDType



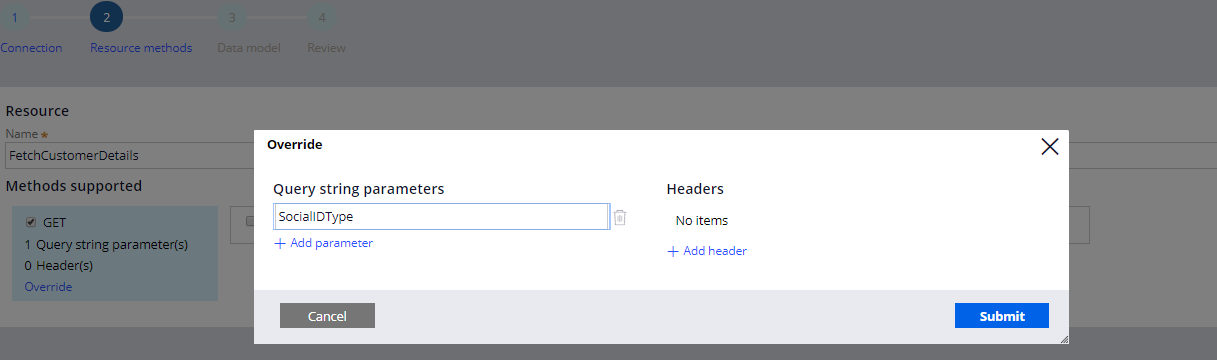
We have completed filling the configuration in connection node.

**Resource Methods**



Select the method on which the service is hosted. Here in the example, we have used GET method.

We also have an option to override the query string and header fields in this section.



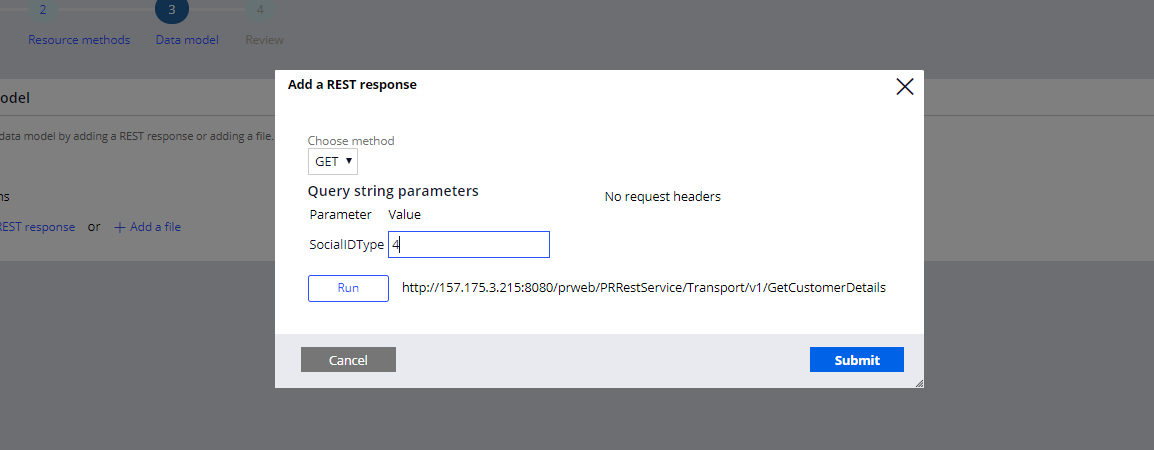
Now we will move to the third screen.

**Data Model**

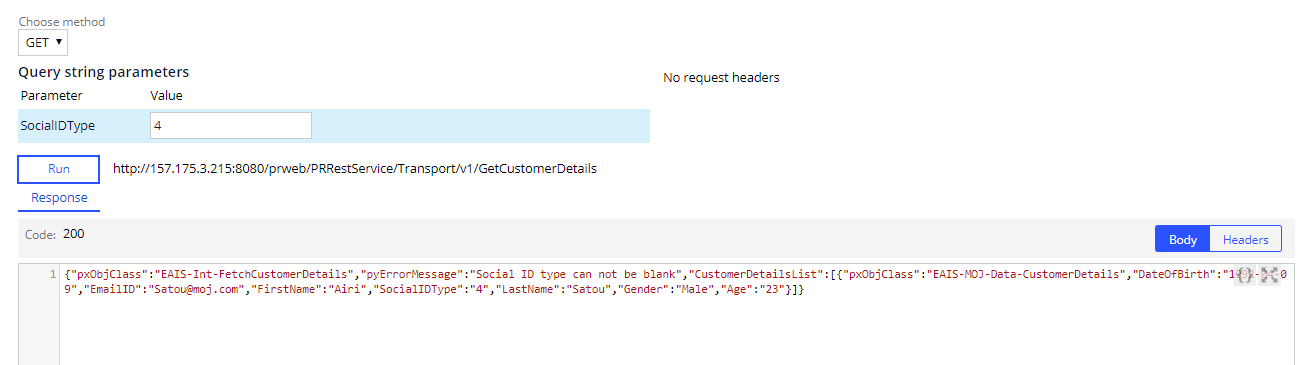
Here we can specify the configuration for request and response parameters

**Sample** – We can use the sample request and response to create the Integration data mdel.

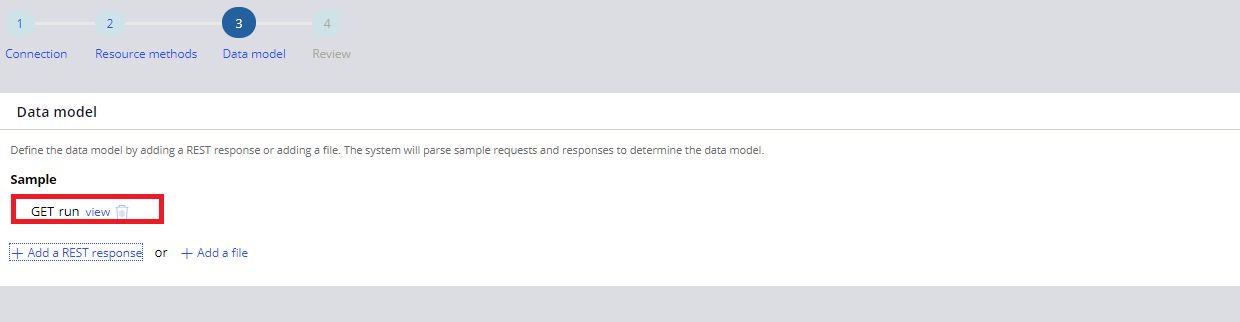
Click on “Add a REST response”



Click on the Run button, for fetching the details related to SocialIDType.



Click on submit and sample response will be saved for creating data model.

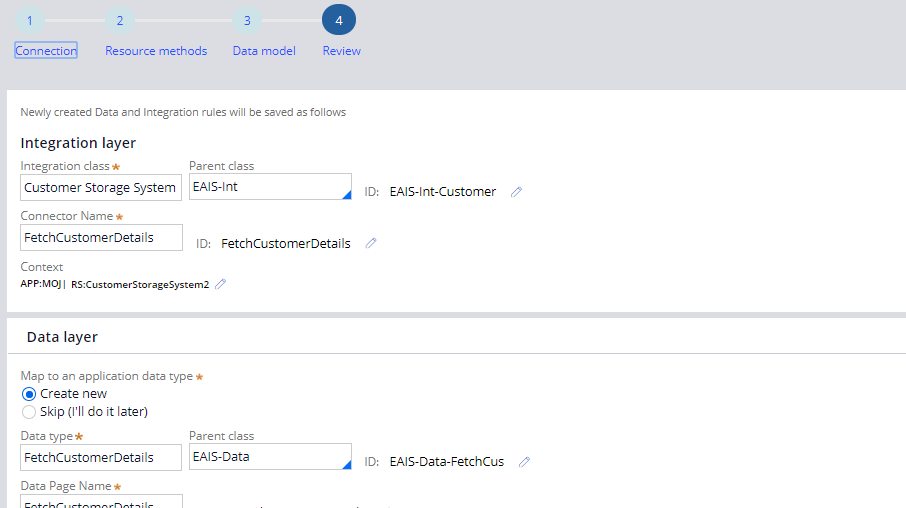


Click Next to proceed to the final step which is Review.

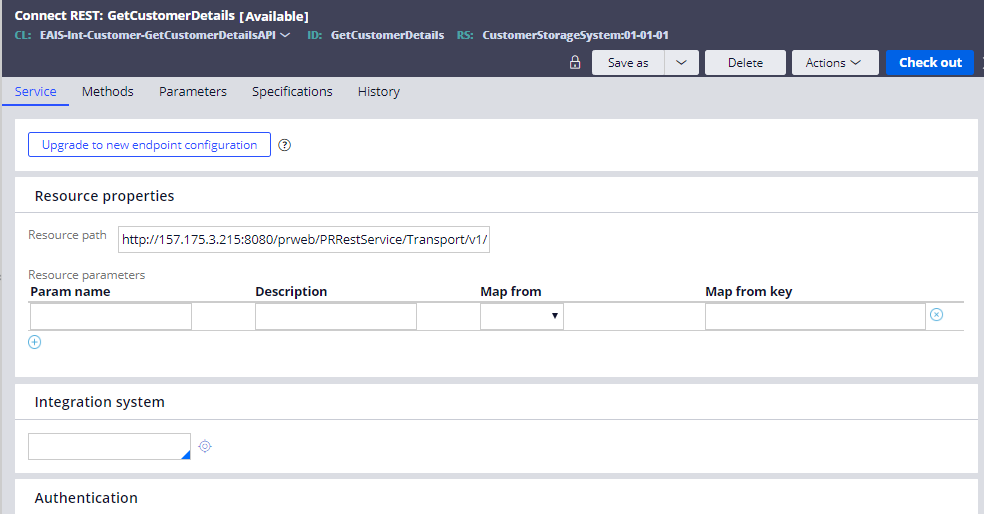
Here we can Review all the rules which we will be created by Pega. We also have an option to edit the name of the rules.

Pega also creates a Data page in Date layer section, we have an option to skip this.

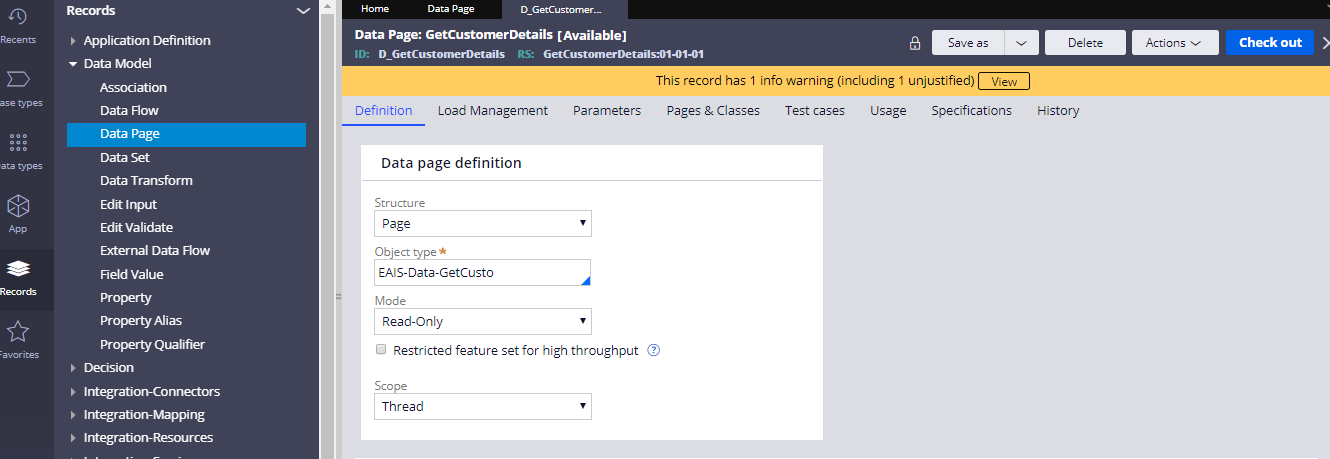
After reviewing the rules click on create.



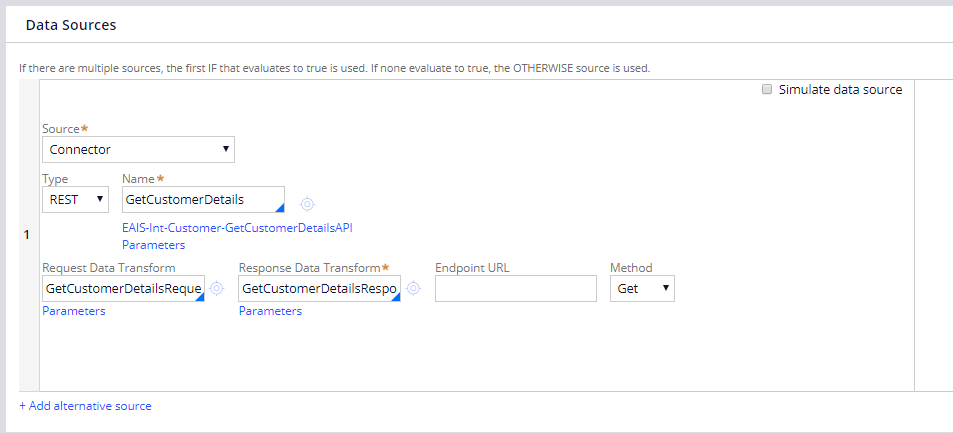
After completing this wizard, we can see the Connect-REST rule which has been created by Pega



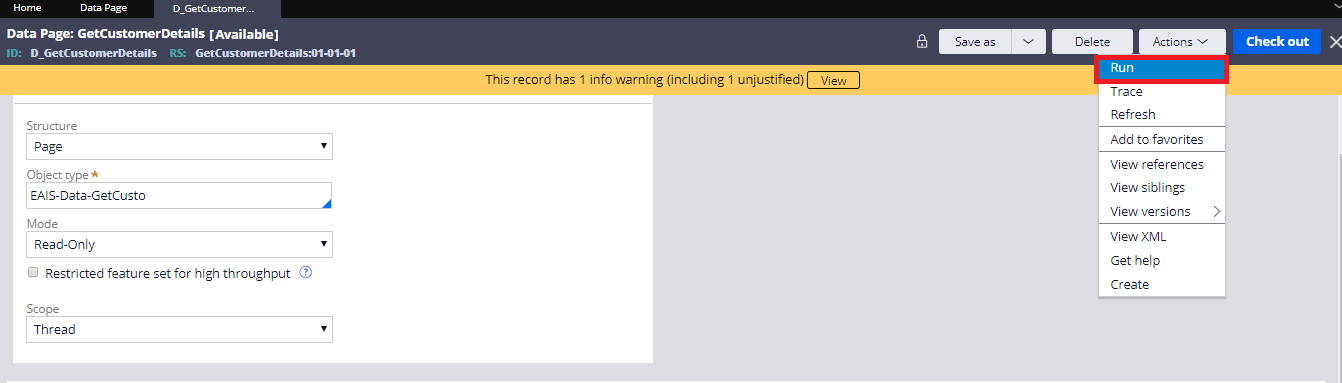
From the Data Model category, we can view the data page which has been created by Pega.

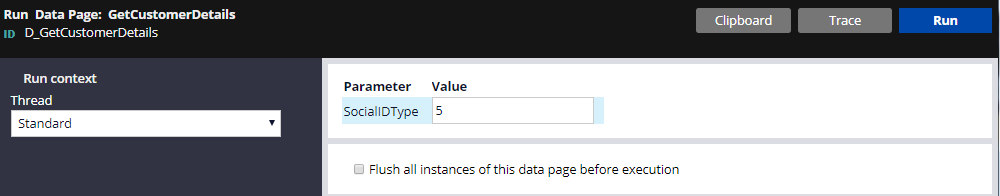


In the data source of the data page, we can view the REST service name which get called when we run this data page



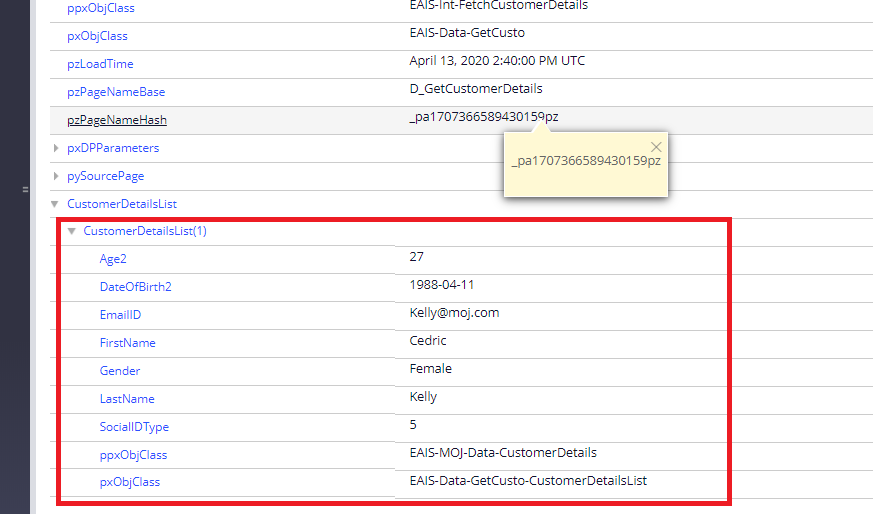
We can Test this service by running this data page.





After entering the value for SocialIDType, click on Run button.

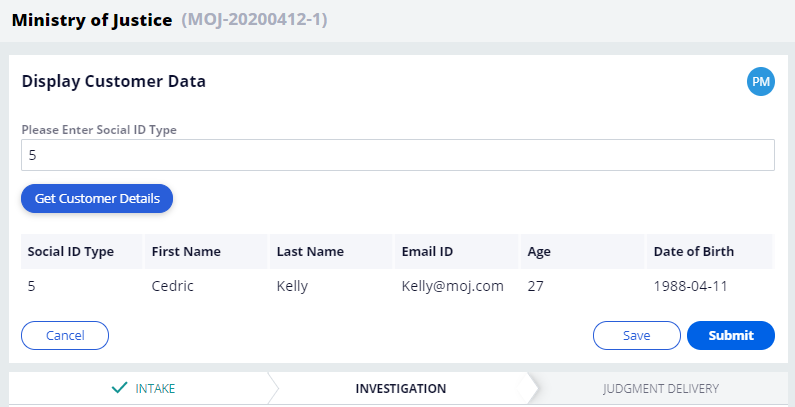
We can view out results inside the “CustomerDetailsList” page list.



We have successfully configured the connect-REST as well.

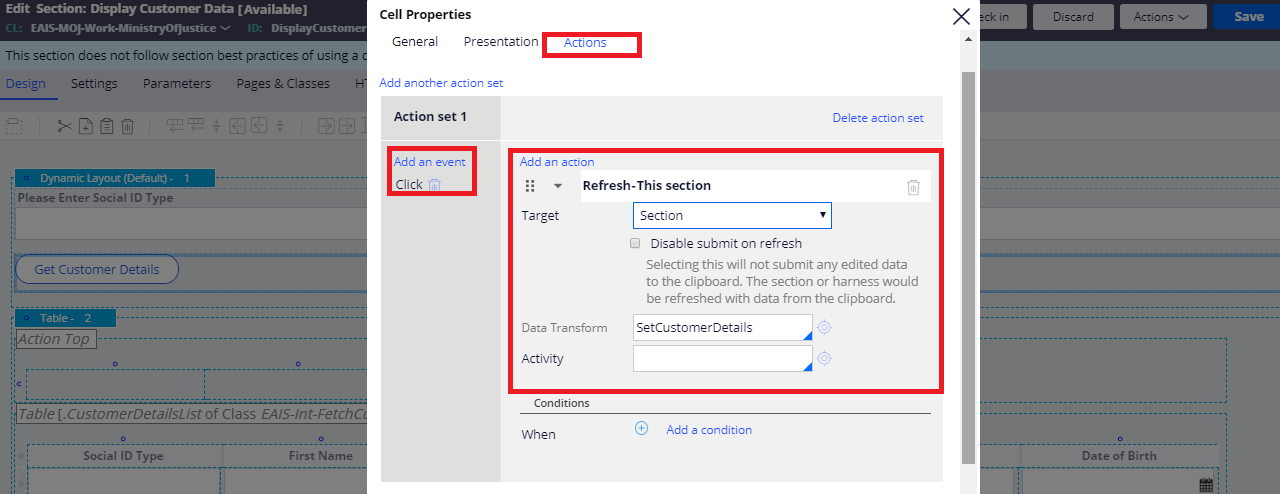
Now our next step is to show the results in the UI.

**Requirement**: Let’s say there is a requirement that when user will enter the SocialIDType and click on the “Get Customer Details” button, then customer details of that particular customer should get populated in the UI.



Step 1: Create a section as “DisplayCustomerData” and include the field storing the **SocialIDType.**

Step 2: Put the button which will fetch the customer details

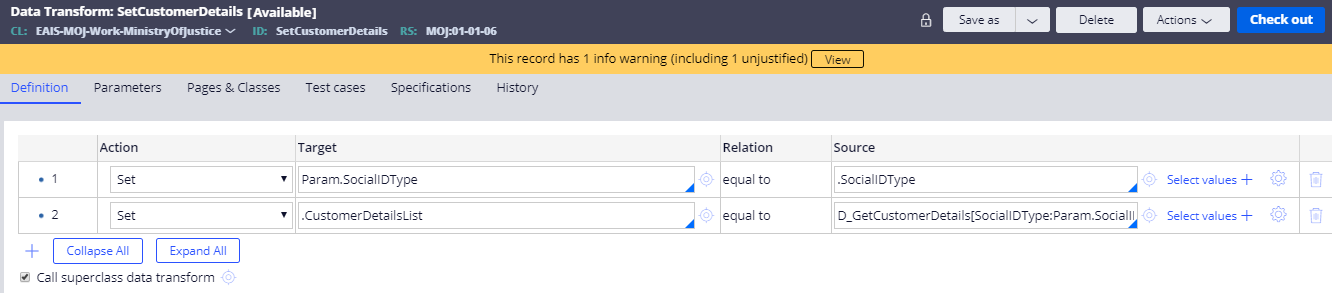


On the Action tab of the button, do the above configuration.

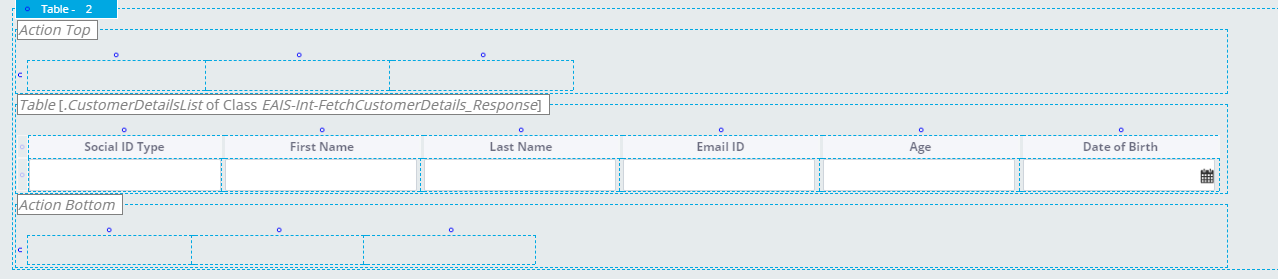
Create a data transform as “SetCustomerDetails” in EAIS-MOJ-Work-MinistryOfJustice class and do the below steps:

Param.SocialIDType = .SocialIDType

.CustomerDetailsList = D\_GetCustomerDetails[SocialIDType:Param.SocialIDType].CustomerDetailsList



Include the Table layout and add the source as property and give the property name as “.CustomerDetailsList”



Include the properties which we want to show to the users.

We are done with the section. Now include this section in the flow action and refer the flow action the flow.

Once the above configurations are done test it by running the case.

