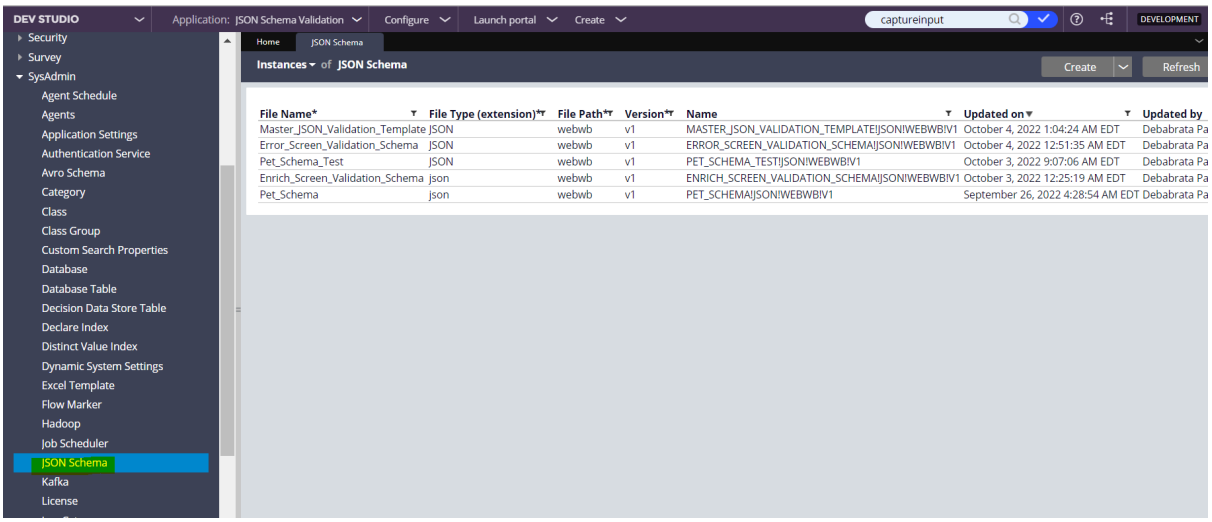
Implementation Guide

This tool has 2 major components.

1. **Defining the JSON Schema in PEGA.**
2. **Use the define schema to validate Pega Screen, Request/Response in connector/service rule.**

Defining the JSON Schema in PEGA.

To define a schema, introduce a new data instance "**JSON Schema**" of class "**`Data-SchemaVal-JSON`**" under **sys admin** category.

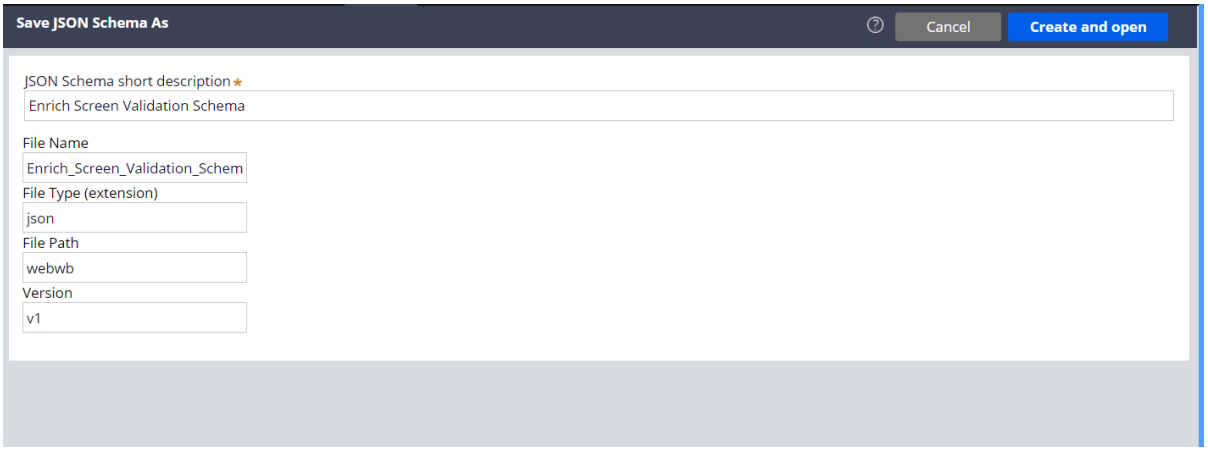


There are 2 ways we can create new json schema instance in PEGA.

Option 1:

Edit the File externally by following "schema_syntax_rulebook" . Create a new JSON schema instance data type by providing below details.

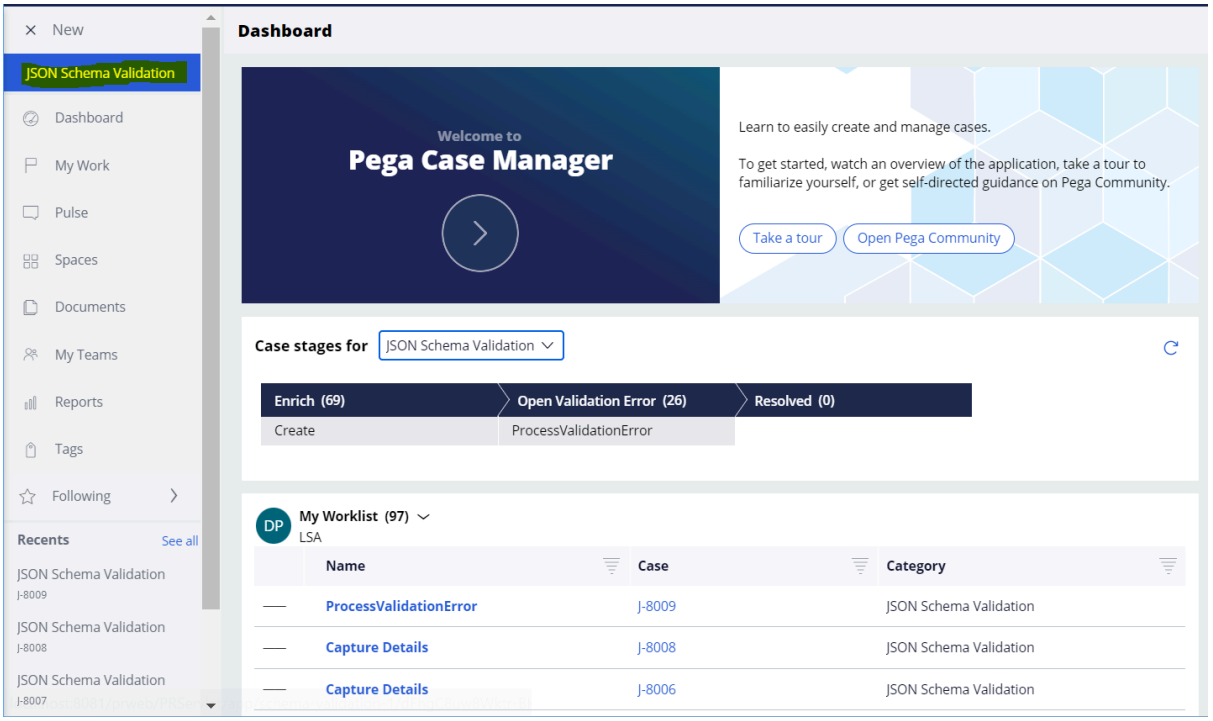
a. **JSON Schema short description** □ Meaningful description of the JSON schema data instance.
b. **File Name** □ Please Provide valid and meaningful file name that represent the purpose more . Like if the JSON is going to be used for screen validation, file name should be <flowActionName>_validation_Schema. For API request/response name should be <API Name>_<request/response>_validation_schema
c. **File Type(Extension)** □ This value should be json always .
d. **File path** □ webwb
e. **Vesion** □ version should be like v1,v2 etc.



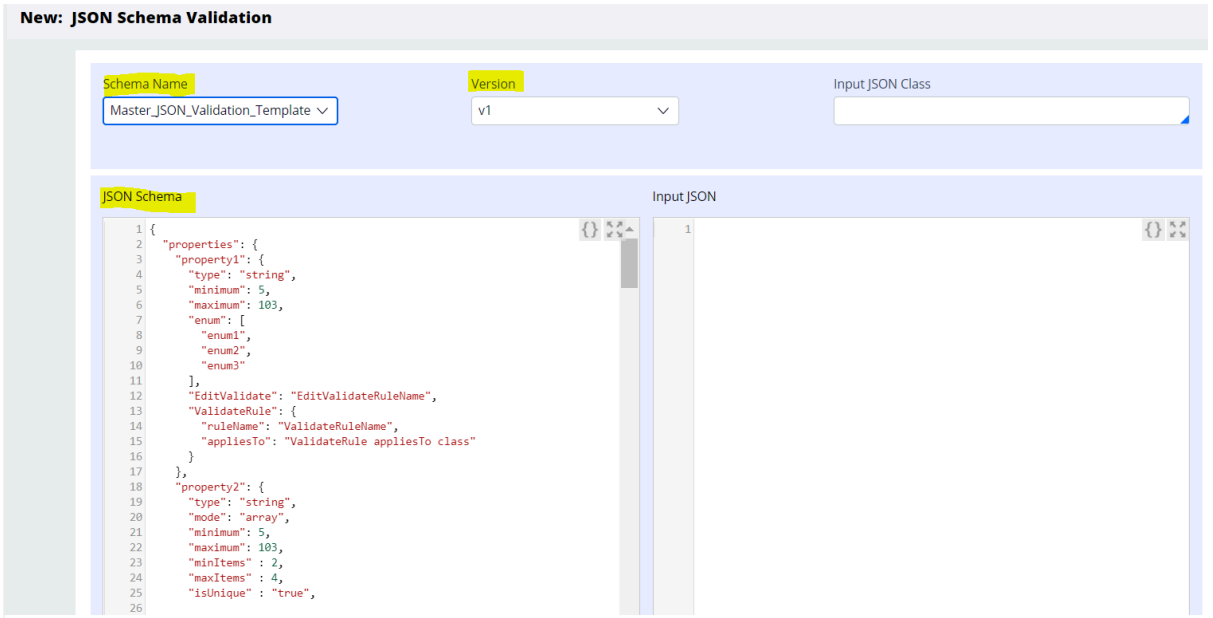
Then use upload file option to upload the file and click on save.

Option 2:

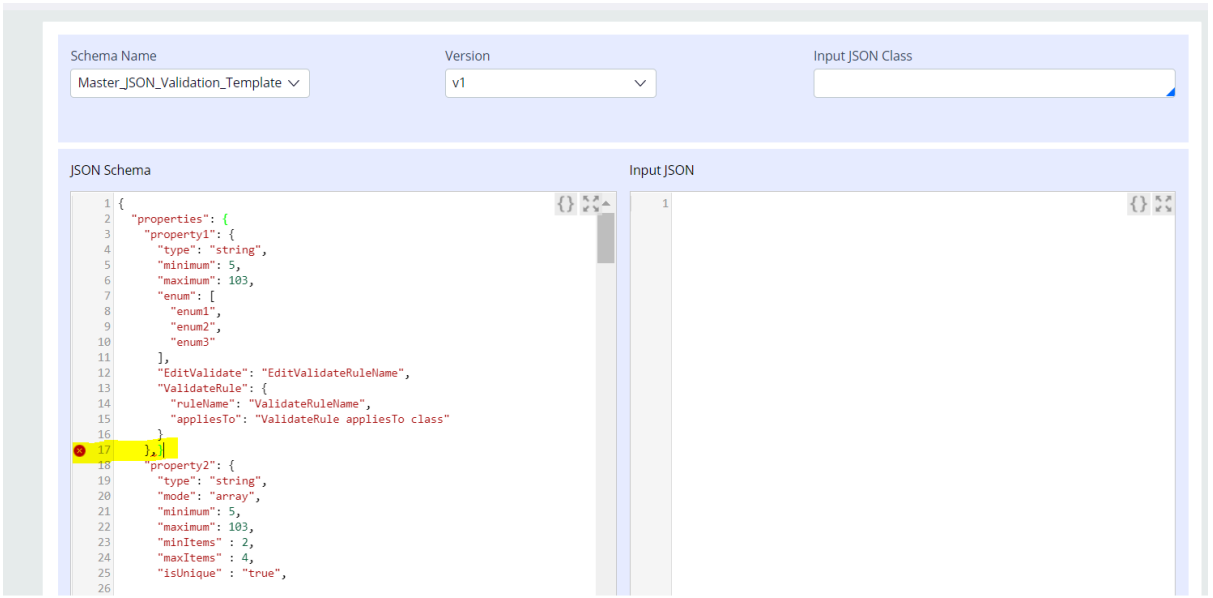
This is the recommended option. Create a instance of case type “JSON Schema Validation”.



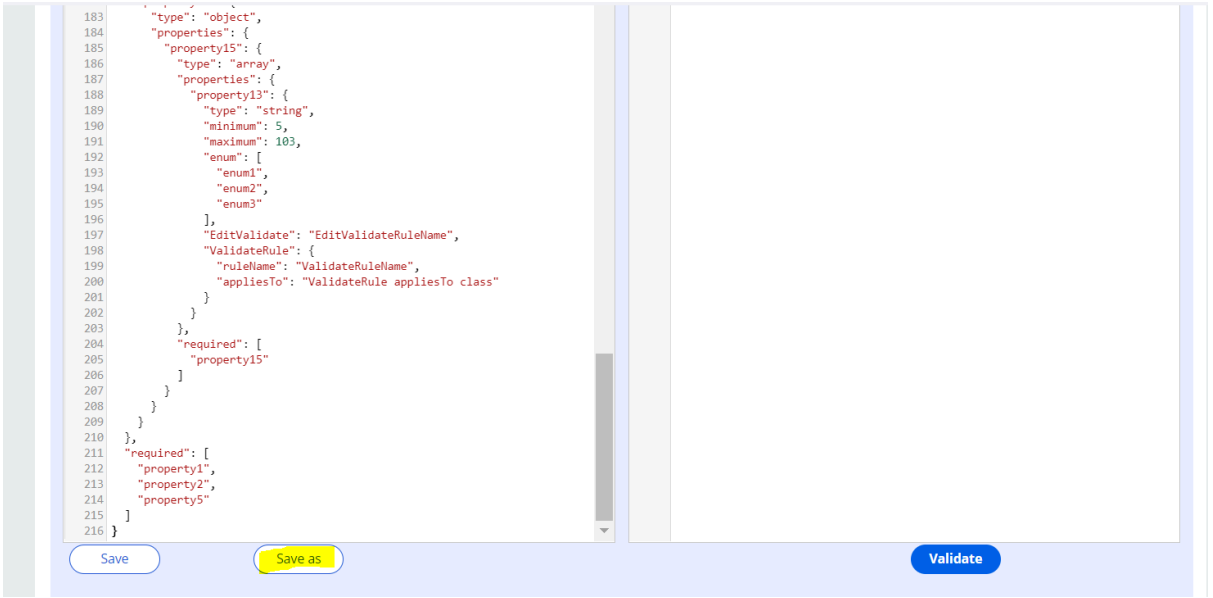
Select a existing schema name and version which suit your purpose most or Select "**Master_JSON_Validation_Template**" as schema name and version as v1 . "Master_JSON_Validation_Template" contains all the json schema validation syntax . The schema will be populated in JSON Schema box automatically.

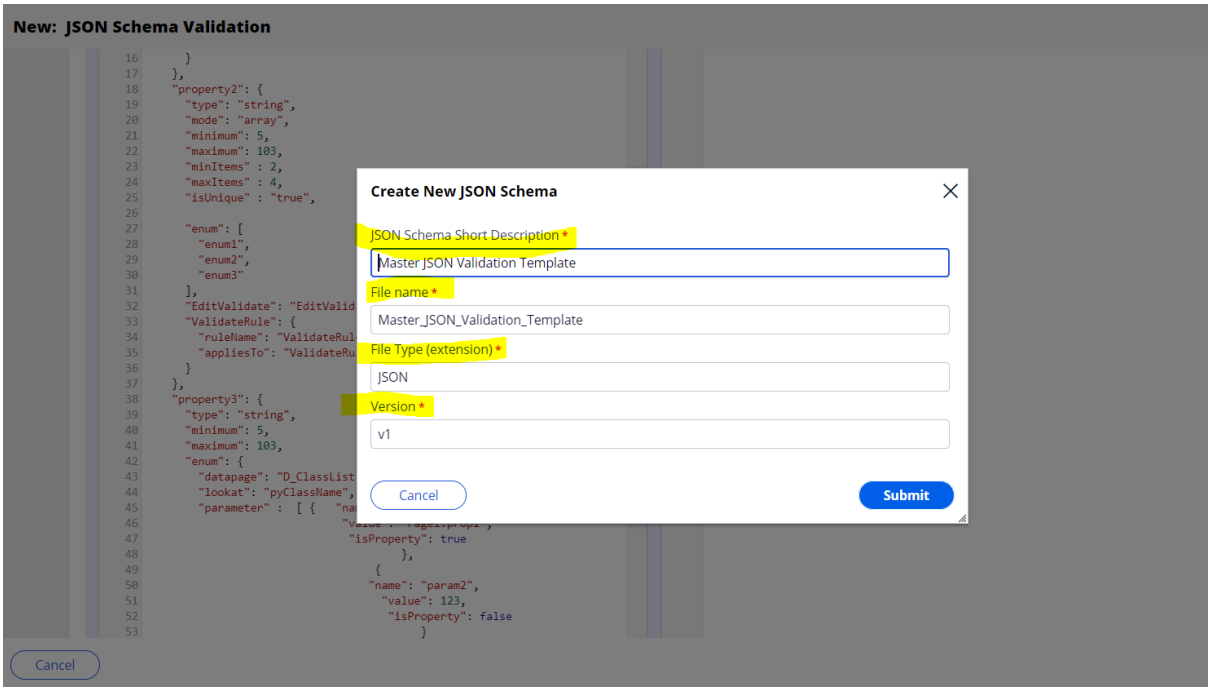


Changes the schema as per you need. PEGA code editor for JSON is very cool. It will throw the error in real time inline if there are any syntax error. Make sure there are no syntax error before save as the modified schema .

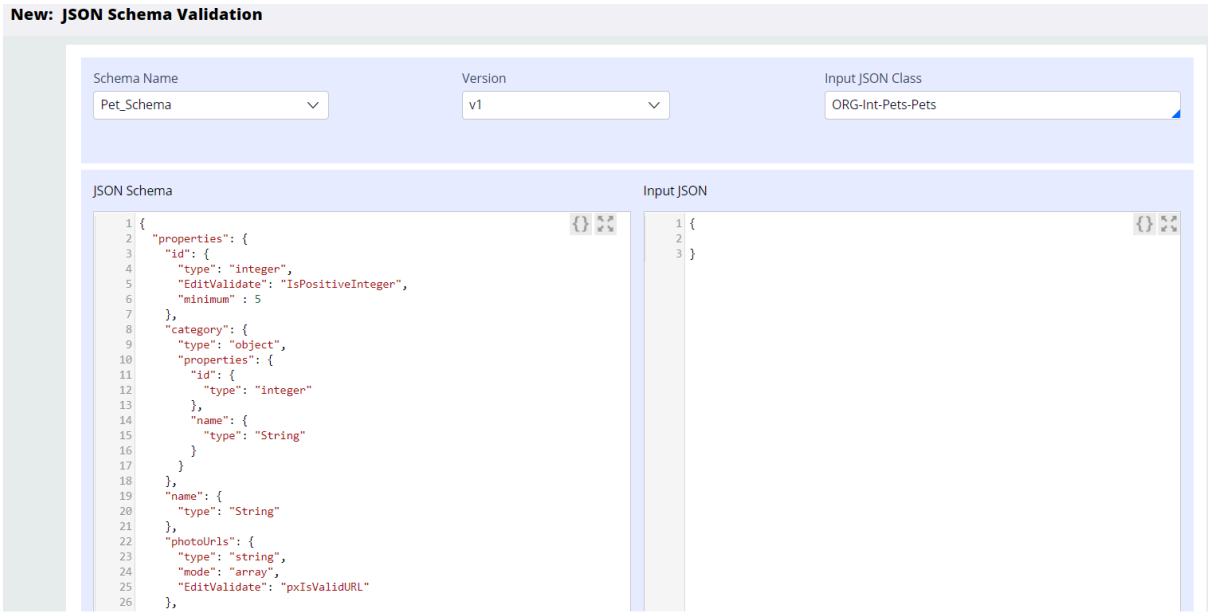


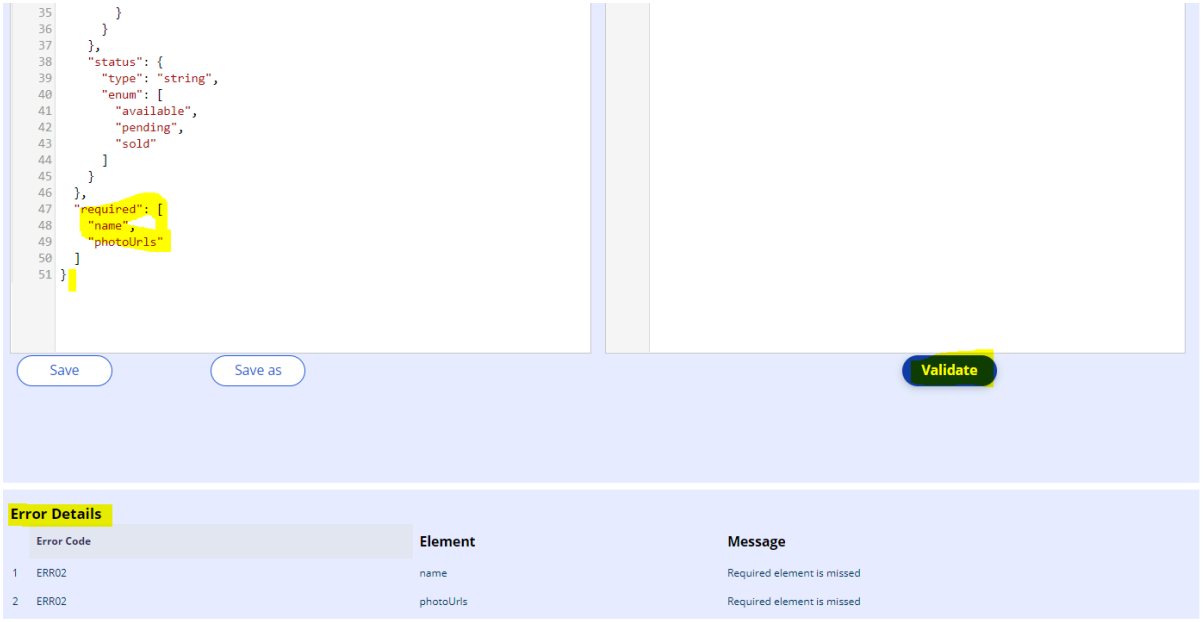
After you finish editing, please click on “Save As” button provide appropriate details to save the new JSON Schema data instance.





After that select the newly created schema from the “Schema Name” dropdown and Select version. Schema definition will be populated in the “JSON Schema” code editor box. Now to unit test the newly created schema provide different input JSON in the “input JSON” code editor panel and click on the validate button. If any error is there during validation, it will display the error below.





While unit testing if you want to modify the schema, edit the details in “JSON Schema” code editor panel and click on the save button. Then continue unit testing.

After all completion of unit testing click on save button .Now your JSON Schema is ready to use in run time.

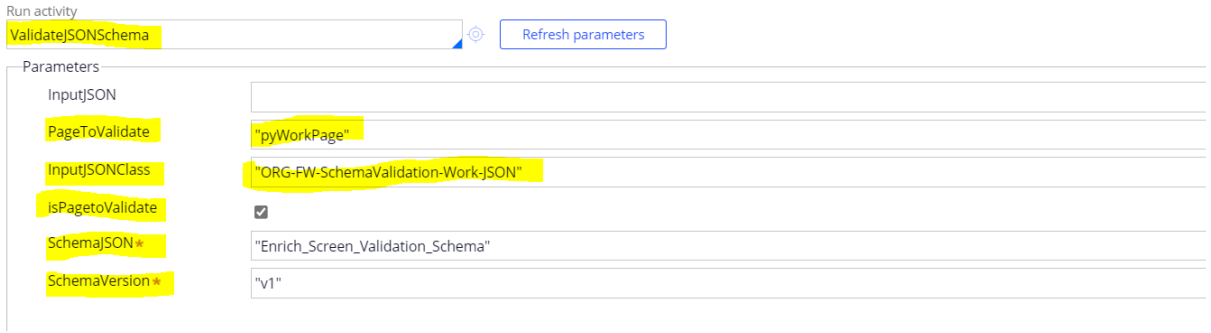
<u>Reference of JSON Schema for Runtime Validation</u> :

PEGA screen Validation

To validate Pega screen using JSON schema validator call **activity** “**ValidateJSONSchema**” from the post processing activity and pass the below highlighted parameter.

a. “**PageToValidate**” □ Pass pega top-level page name where the message will display
b. “**InputJSONClass**” □ Pass the top level page class.
c. “**isPageToValidate**” □ check this checkbox for PEGA screen validation.
d. “**SchemaJSON**” □ Name of the JSON schema rule for the screen validation
e. “**SchemaVersion**” □ version of schema name will be used for screen validation

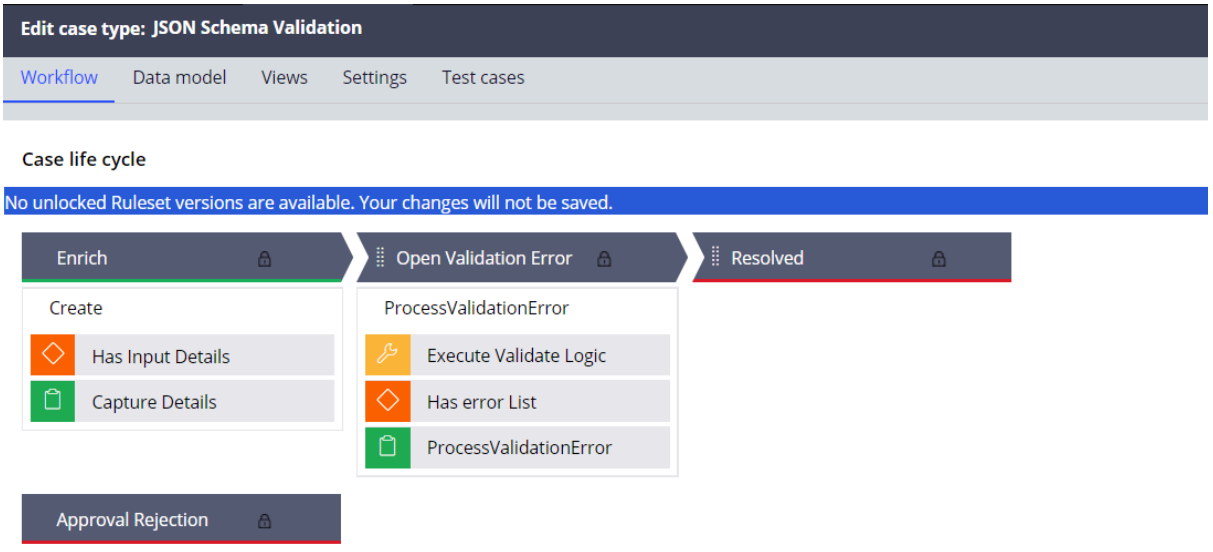
We are ready to validate the screen using the screen using JSON schema validator tool.



To validate request/response for PEGA connector/services.

There are two options for request/response for PEGA connector/services.

Option 1: **By injecting “JSON Schema Validation” case dependencies**.



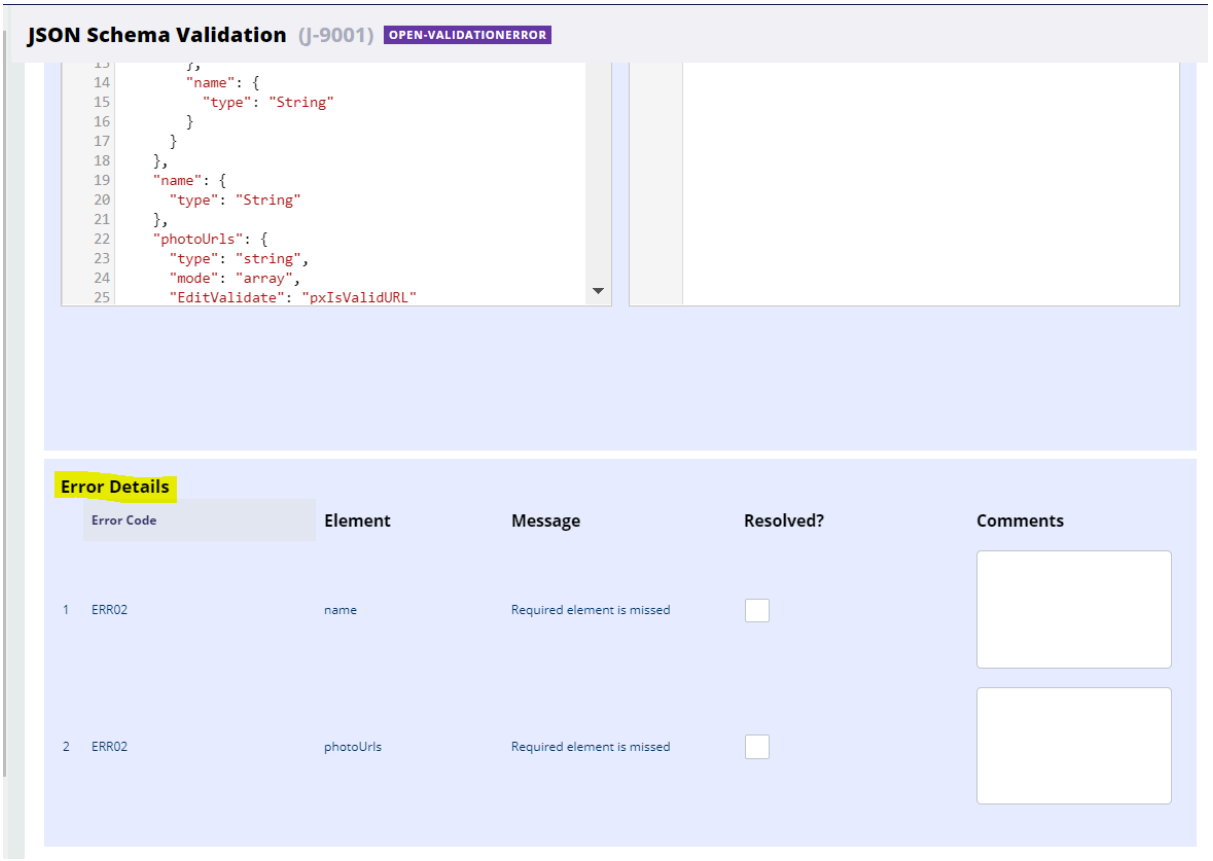
From the process flow where PEGA connector/service rule getting invoked, create a child case of case type “**JSON Schema Validation**” . Propagate Set the below properties values to the the child case.

a. **SchemaName** □ Name of the JSON schema rule for the request/response validation

b. **SchemaVersion** □ version of schema name will be used for request/response validation
c. **InputJSONClass** □ Top-Level page class of request/response
d. **InputJSON** □ Request/response JSON

If the input JSON is correct against the JSON schema provided the "JSON Schema Validation" child case will be resolved successfully .

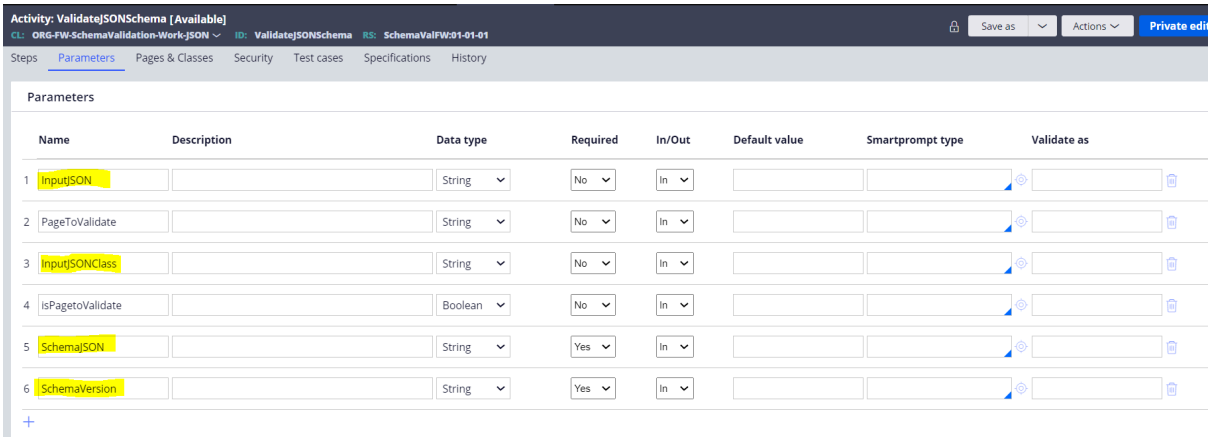
If there are any error the "JSON Schema Validation" child case will be moved to an assignment with "Open-ValidationError" status for further investigation. With this feature you can leverage all the PEGA case features and can be implemented as per the organization need.



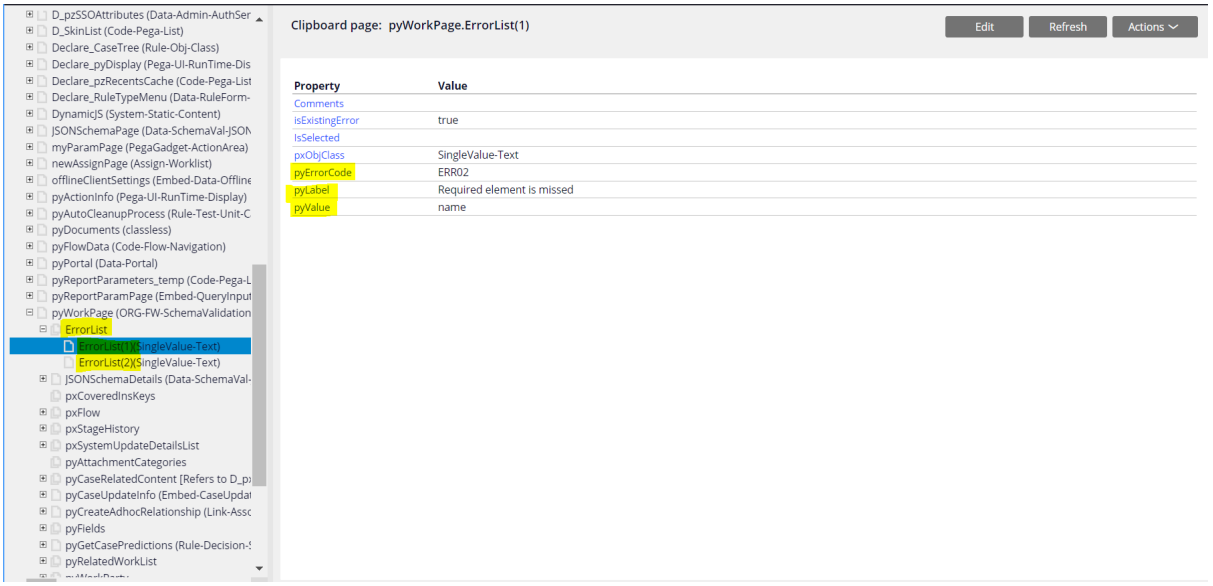
Option 2: **Without Case dependency injection**.

Just all the activity "**ValidateJSONSchema**" from the appropriate place and pass below parameter.

a. "**InputJSON**" □ Request/Response JSON
b. "**InputJSONClass**" □ Top level class of Request/Response
c. "**SchemaJSON**" □ Name of the JSON schema rule for the request/response validation
d. "**SchemaVersion**" □ version of schema name will be used for request/response validation

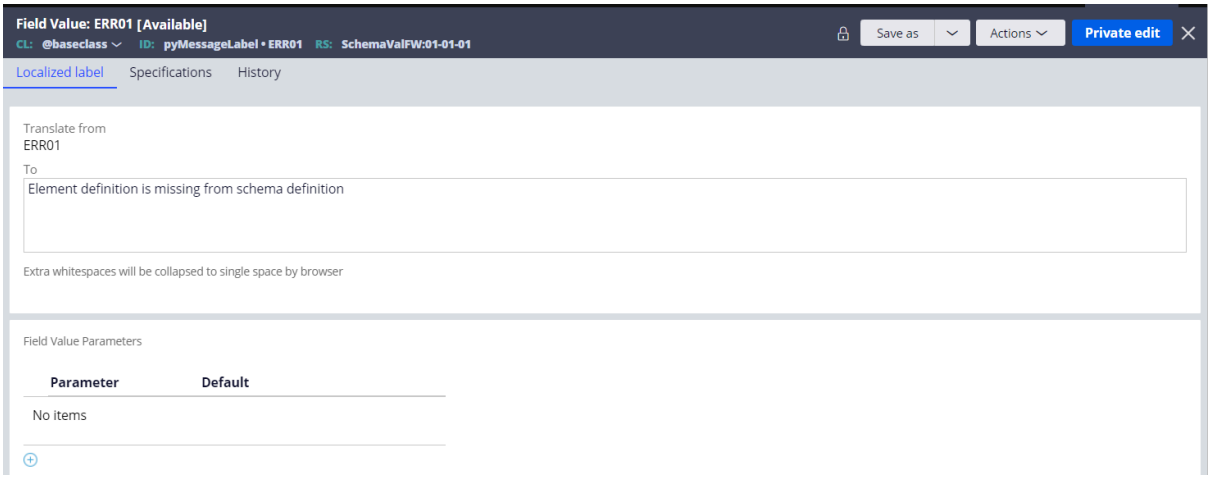


. If there are any error during schema validation. In the runtime primary page of "**ValidateJSONSchema**" will contain one **page list** called "**ErrorList**". Please Process the "ErrorList" Pagelist as per your need.



Message Localization:

All the messages are build using "**pyMessageLebel" field value** for localization as per organization standard based on error code. Below is one of the examples.



JSON Schema Validation API

I have built one API using PEGA service to expose the solution to the external world without depending on installing PEGA code. This API will take all the mandatory input to validate the JSON. Then it will create a “PEGA JSON Validation” case .

If any mandatory input is missing, the case will be routed to “Open-Verification” status to capture schema details.

If any validation error is there in the input JSON case will be routed to “Open-ValidationError” Status error resolution.

If there are no errors, the case will be resolved and status will be success.

API URI :

http://localhost:8081/prweb/api/JSONSchemaValidation/v1/JSONSchemaValidation

{Server-URL}/api/JSONSchemaValidation/v1/JSONSchemaValidation

Input JSON of the API:

{

"SchemaName" : "PEGA JSON Schema data Instance Name}}",

"SchemaVersion" : "PEGA JSON Schema data Instance version",

"InputJSONClass" : "Top level class og input JSON",

"InputJSONDetails" : { Input JSON which needs to be validated }

}

Example of Request JSON of the API

{

"SchemaName" : "Pet_Schema_Test",

"SchemaVersion" : "v1",

"InputJSONClass" : "ORG-Int-Pets-Pets",

"InputJSONDetails" : {"name" :
"Debabrata","photoUrls":["https://www.google.com/imgres","https://www.google.com/imgres"]}

}

Example of Response

Error Response:

{ "CaseID":"J-11071" ,"pxObjClass":"ORG-FW-SchemaValidation-Work-JSON" ,"pyStatusWork":"Open-ValidationError"
,"Status":"Fail" ,"ErrorList":[{ "ElementName":"photoUrls" ,"ErrorCode":"ERR16" ,"message":"List has duplicate items"
,"pxObjClass":"SingleValue-Text" }] }

Success Response:

{ "CaseID":"J-11072" ,"pxObjClass":"ORG-FW-SchemaValidation-Work-JSON"
,"pyStatusWork":"Resolved-Completed" ,"Status":"Success" }