

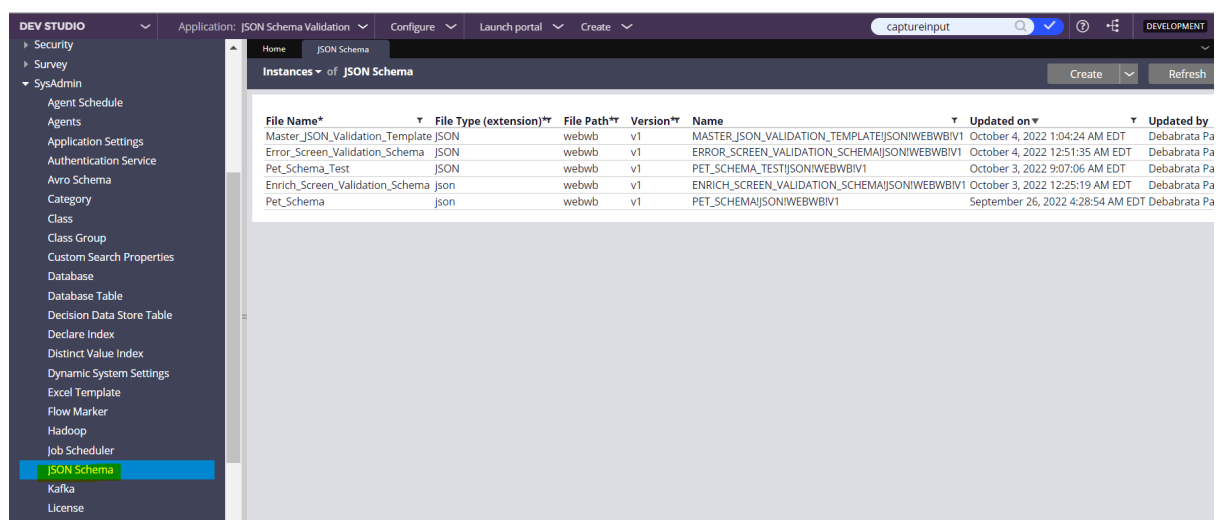
Implementation Guide

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.

Save JSON Schema As

?

Cancel

Create and open

JSON Schema short description *

Enrich Screen Validation Schema

File Name

Enrich_Screen_Validation_Schem

File Type (extension)

json

File Path

webwb

Version

v1

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

Main

History

JSON Schema Preview

JSONSchema

```
1 {
2   "properties": {
3     "ErrorList": {
4       "type": "array",
5       "properties": {
6         "IsSelected": {
7           "type": "boolean",
8           "enum": [
9             "true"
10          ]
11        },
12        "Comments": {
13          "type": "String",
14          "maximum": 500
15        }
16      },
17      "required": [
18        "IsSelected",
19        "Comments"
20      ]
21    }
22  }
23 }
```

Upload file

Download file

Option 2:

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

Dashboard

Welcome to **Pega Case Manager**

Learn to easily create and manage cases. To get started, watch an overview of the application, take a tour to familiarize yourself, or get self-directed guidance on Pega Community.

[Take a tour](#) [Open Pega Community](#)

Case stages for **JSON Schema Validation**

Enrich (69)	Open Validation Error (26)	Resolved (0)
Create	ProcessValidationError	

My Worklist (97)

Name	Case	Category
ProcessValidationError	J-8009	JSON Schema Validation
Capture Details	J-8008	JSON Schema Validation
Capture Details	J-8006	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.

New: JSON Schema Validation

Schema Name: **Master_JSON_Validation_Template** Version: **v1** Input JSON Class:

JSON Schema

```

1 {
2   "properties": {
3     "property1": {
4       "type": "string",
5       "minimum": 5,
6       "maximum": 103,
7       "enum": [
8         "enum1",
9         "enum2",
10        "enum3"
11      ],
12      "EditValidate": "EditValidateRuleName",
13      "ValidateRule": {
14        "ruleName": "ValidateRuleName",
15        "appliesTo": "ValidateRule appliesTo class"
16      }
17    },
18    "property2": {
19      "type": "string",
20      "mode": "array",
21      "minimum": 5,
22      "maximum": 103,
23      "minItems": 2,
24      "maxItems": 4,
25      "isUnique": "true",
26    }
  }

```

Input JSON:

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 .

Schema Name

Master_JSON_Validation_Template

Version

v1

Input JSON Class

JSON Schema

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26

```

{
  "properties": {
    "property1": {
      "type": "string",
      "minimum": 5,
      "maximum": 103,
      "enum": [
        "enum1",
        "enum2",
        "enum3"
      ],
      "EditValidate": "EditValidateRuleName",
      "ValidateRule": {
        "ruleName": "ValidateRuleName",
        "appliesTo": "ValidateRule appliesTo class"
      }
    },
    "property2": {
      "type": "string",
      "mode": "array",
      "minimum": 5,
      "maximum": 103,
      "minItems": 2,
      "maxItems": 4,
      "isUnique": "true",

```

Input JSON

1

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

183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216

```

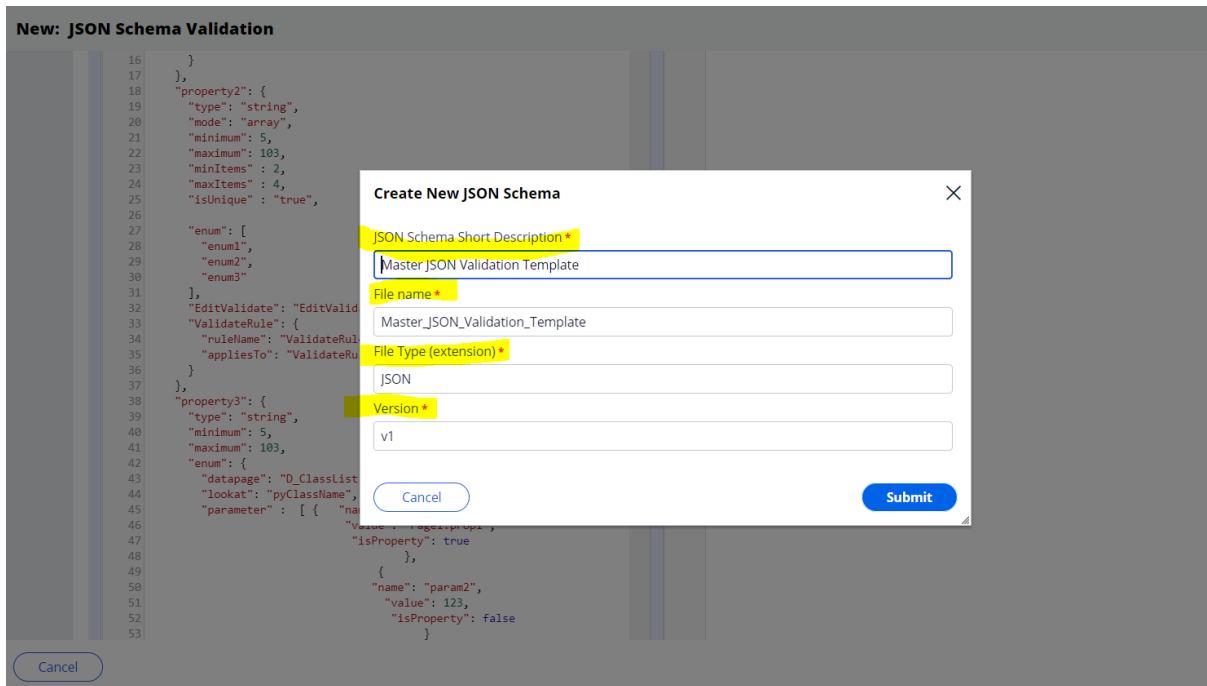
      "type": "object",
      "properties": {
        "property15": {
          "type": "array",
          "properties": {
            "property13": {
              "type": "string",
              "minimum": 5,
              "maximum": 103,
              "enum": [
                "enum1",
                "enum2",
                "enum3"
              ],
              "EditValidate": "EditValidateRuleName",
              "ValidateRule": {
                "ruleName": "ValidateRuleName",
                "appliesTo": "ValidateRule appliesTo class"
              }
            }
          },
          "required": [
            "property15"
          ]
        }
      },
      "required": [
        "property1",
        "property2",
        "property5"
      ]
    }
  }

```

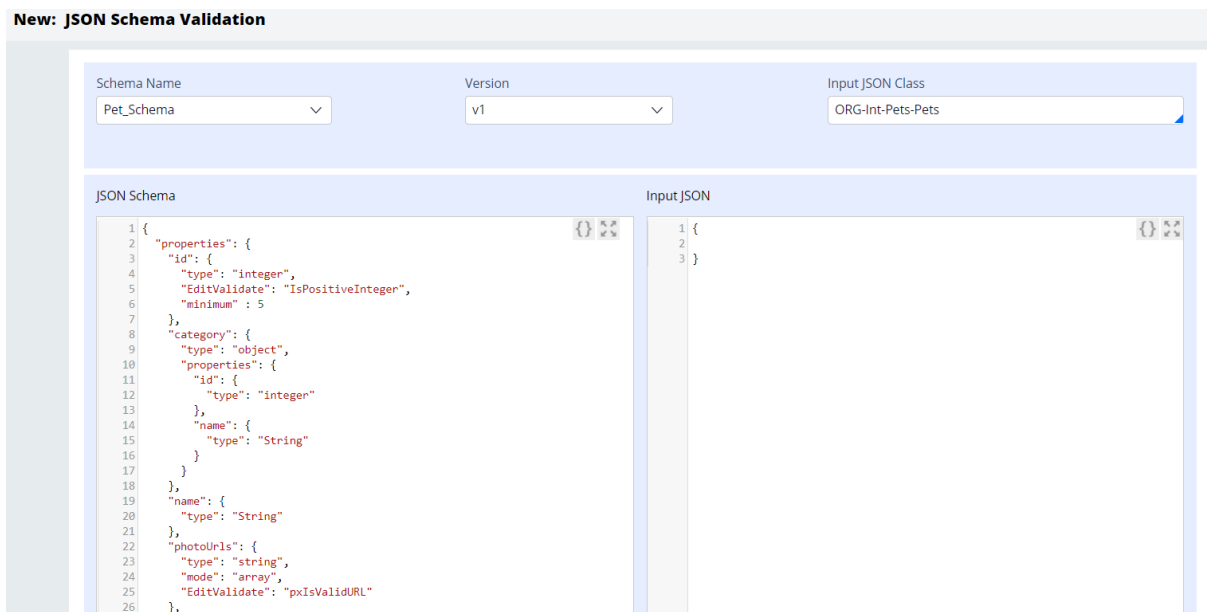
Save

Save as

Validate



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.



```
35     }
36   },
37 },
38 "status": {
39   "type": "string",
40   "enum": [
41     "available",
42     "pending",
43     "sold"
44   ]
45 },
46 },
47 "required": [
48   "name",
49   "photoUrls"
50 ]
51 }
```

Save Save as Validate

Error Details		
Error Code	Element	Message
1 ERROR2	name	Required element is missed
2 ERROR2	photoUrls	Required element is missed

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.

```
35     type: "string"
36   }
37 },
38 "status": {
39   "type": "string",
40   "enum": [
41     "available",
42     "pending",
43     "sold"
44   ]
45 },
46 },
47 "required": [
48   "name",
49   "photoUrls"
50 ]
51 }
```

Save Save as Validate

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

Reference of JSON Schema for Runtime Validation :

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.

- "PageToValidate"** ☐ Pass pega top-level page name where the message will display
- "InputJSONClass"** ☐ Pass the top level page class.
- "isPageToValidate"** ☐ check this checkbox for PEGA screen validation.
- "SchemaJSON"** ☐ Name of the JSON schema rule for the screen validation
- "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.

Run activity

ValidateJSONSchema Refresh parameters

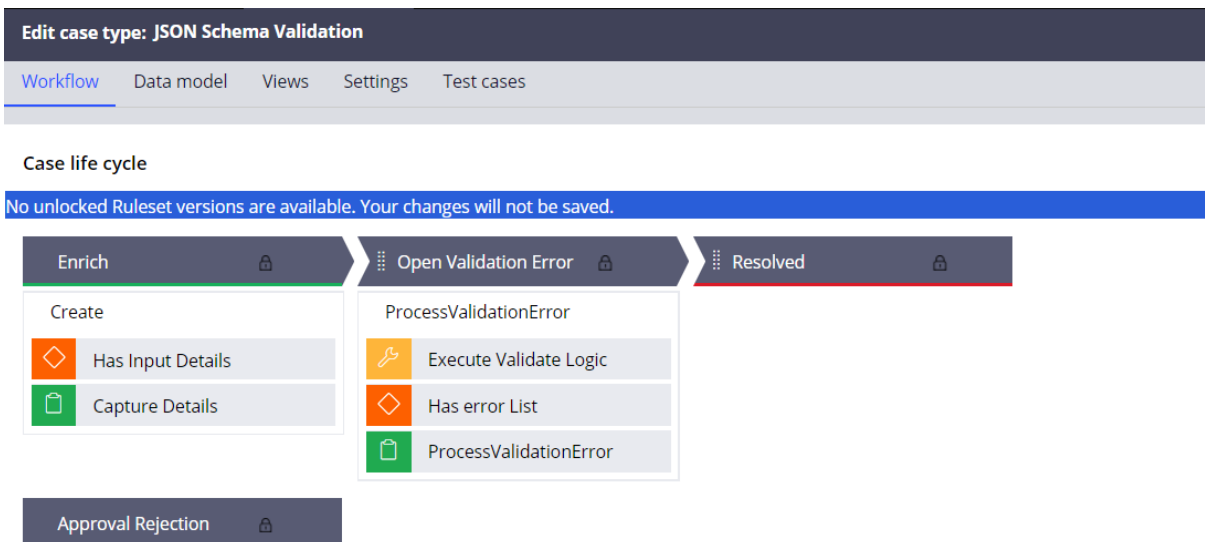
Parameters

InputJSON	
PageToValidate	"pyWorkPage"
InputJSONClass	"ORG-FW-SchemaValidation-Work-JSON"
isPageToValidate	☒
SchemaJSON *	"Enrich_Screen_Validation_Schema"
SchemaVersion *	"v1"

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.

- 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.

JSON Schema Validation (J-9001)
OPEN-VALIDATIONERROR

```

14      "name": {
15        "type": "String"
16      }
17    },
18  },
19  "name": {
20    "type": "String"
21  },
22  "photoUrls": {
23    "type": "string",
24    "mode": "array",
25    "EditValidate": "pxIsValidURL"

```

Error Details

Error Code	Element	Message	Resolved?	Comments
1 ERR02	name	Required element is missed	☐	
2 ERR02	photoUrls	Required element is missed	☐	

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

Activity: ValidateJSONSchema [Available]

ORG-FW-SchemaValidation-Work-JSON ~ ID: ValidateJSONSchema RS: SchemaValFW-01-01-01

Save as Actions Private edit

Steps Parameters Pages & Classes Security Test cases Specifications History

Parameters

Name	Description	Data type	Required	In/Out	Default value	Smartprompt type	Validate as
1 InputJSON		String	No	In			
2 PageToValidate		String	No	In			
3 InputJSONClass		String	No	In			
4 isPagetoValidate		Boolean	No	In			
5 SchemaJSON		String	Yes	In			
6 SchemaVersion		String	Yes	In			

. 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.

D_pzSSOAttributes (Data-Admin-AuthSer

D_SkinList (Code-Pega-List)

Declare_CaseTree (Rule-Obj-Class)

Declare_pyDisplay (Pega-UI-RunTime-Di

Declare_pzRecentsCache (Code-Pega-List

Declare_RuleTypeMenu (Data-RuleForm-

DynamicJS (System-Static-Content)

JSONSchemaPage (Data-SchemaVal-JSON

myParamPage (PegaGadget-ActionArea)

newAssignPage (Assign-Worklist)

offlineClientSettings (Embed-Data-Offline

pyActionInfo (Pega-UI-RunTime-Display)

pyAutoCleanupProcess (Rule-Test-Unit-C

pyDocuments (classless)

pyFlowData (Code-Flow-Navigation)

pyPortal (Data-Portal)

pyReportParameters_temp (Code-Pega-L

pyReportParamPage (Embed-QueryInput

pyWorkPage (ORG-FW-SchemaValidation

ErrorList

ErrorList2(SingleValue-Text)

ErrorList2(SingleValue-Text)

JSONSchemaDetails (Data-SchemaVal-

pxCoveredInsKeys

pxFlow

pxStageHistory

pxSystemUpdateDetailsList

pyAttachmentCategories

pyCaseRelatedContent [Refers to D_p

pyCaseUpdateInfo (Embed-CaseUpdat

pyCreateAdhocRelationship (Link-Assc

pyFields

pyGetCasePredictions (Rule-Decision-!

pyRelatedWorkList

pyRelatedData

Clipboard page: pyWorkPage.ErrorList(1)

Edit Refresh Actions

Property	Value
Comments	
IsExistingError	true
IsSelected	
pyObjClass	SingleValue-Text
pyErrorCode	ERR02
pyLabel	Required element is missed
pyValue	name

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.

Field Value: ERR01 [Available]
CL: @baseclass ID: pyMessageLabel • ERR01 RS: SchemaValFW:01-01-01
Save as Actions Private edit

Localized label Specifications History

Translate from
ERR01

To
Element definition is missing from schema definition

Extra whitespaces will be collapsed to single space by browser

Field Value Parameters

Parameter	Default
No items	

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" }
```