Guidance on GSP Customization

Contents

1 Introduction

1.1.1 Document Purpose

The Saviynt provide feasibility to do the customization of the create/update user request page as one of part of Third party access Governance (TPAG). Using this feature the create and user update request page can be customized to meet the client requirement apart from configuring the request form in Global configuration by using dynamic attributes.

Using this feature, following things can be achieved.

- Can access Saviynt internal data via analytics.
- Can do complex validation on the UI attributes.
- Drive the attributes based on the UI events.
- Changing the scoping of attributes based on user login like
    - Make certain attributes editable only if login user is admin or sponsor for end user it will be readonly attribute.
- Populate data to attribute in form based on UI events like
    - Onclick
    - Onchange
    - onFocus
    - OnMouseover and so on

purpose of this document is to provide guidance for doing GSP customization of following page.

- createupdateuserrequestfirststep.gsp

to achieve above mentioned use case.

1.1.2 Audience

The primary audience for this document is as follows:

- Technical team

It is expected that the person using this document has experience in SSM administration or background in administering Identity Governance platforms.

The person should have knowledge on following area.

- HTML
- JSP
- Javascript
- Jquery
- Ajax

2 Steps for GSP customizations

The following are steps involved in GSP customizations.

- Identify and download the OOTB page from Saviynt.
- Customise the GSP
- Assign feature access to Sav role.
- Upload the updated page in EIC

- Restart the pod

2.1.1 Manage CIdentify and download the OOTB page from Saviynt.

You can manage the GSP files from the **Views** tab on the **File Directory** page.

- Log on to EIC.
- On the top right of the Home page, click the **Applications** icon, and select **Admin**.
- Click the **Menu** icon.
- In the left pane, under **Settings**, click **File Directory**.
- The **File Directory** page is displayed.

Click the **Views** tab, and then perform any required action listed in the following table:

Action	Steps
Upload	To upload the GSP files: 1. Click the Upload icon. 2. In the File To Upload dialog box, do the following: i. Click Select next to the File box. ii. Select the GSP file to upload, and then click Open. The selected file is displayed in the File To Upload box. Note You must retain the original name of the GSP file before uploading it. For example, if you upload the technicalRuleEdit.gsp file then you must retain the original name of the file before uploading.

Action	Steps
	iii. Click **Upload**. If the upload is successful, the **File Uploaded Successfully** message appears. Note If a file with the same name exists, the version number of the newly uploaded file is numerically incremented. For example, if the Create.gsp file with version number 1 already exists, then version number 2 is assigned to the newly uploaded file. iv. Click **Close**. v. Restart the service for the change to take effect. For more information, see Restarting the Services.
Edit	3. To modify the GSP files: 4. Select the GSP file you want to modify, and then click the **Edit** icon. The **Create File** dialog box is displayed. 5. Perform the necessary changes and click **Submit**. 6. Click **Close**. 7. Restart the service for the change to take effect. For more information, see Restarting the Services.
Download	To download a GSP file to your local system: 1. Select the GSP file you want to download and click the **Download** icon. If the download is successful, the **File Downloaded Successfully** message is displayed at the top of the page.

Action	Steps
	2. To download a particular version of the selected GSP file: i. Click the **Versions** icon. ii. In the dialog box, select the version number of the GSP file you want to download and click **Download**. If the download is successful, the **File Downloaded Successfully** message appears at the top of the page.

2.1.2 Customise the GSP

This section will explain how to do the customization of GSP page with few samples code for following use cases.

- Accessing the Saviynt data and populate in custom field created from GSP.
- Custom Validation
- Create a derived field
- Populating/Setting the data for dynamic attributes created in Global config.

2.1.2.1 *Accessing the Saviynt data and populate in custom field created from GSP*

To achieve this usecase its spitted into two main parts

- Building the HTML element for showing it in UI
- Getting the internal Saviynt data via analytics using exposed API "**${request.contextPath}/customize/getdbdata**"

2.1.2.1.1 Building HTML elements

- To create the custom input type in Create or user update request form, the following code snippet shared in below sample should be added in the gsp page (createupdateuserrequestfirststep.gsp)

  To illustrate the above functionality the samples are shared below for following the HTML element to show it in UI.

  - Text Field
  - Drop down.

  By using the same way other HTML elements can be build as well.

2.1.2.1.1.1 How to create a text field

The following sample code shown below used create input type text field using the custom code. The customer can utilize the OOTB class used in in form to utilize the Css.

- Sample field name – *Description and mapped to user customproperty29*

```
<div class="row">
                        <div class="col-md-4">
                           %{--Description--}%
                           <div class="form-group">
                              <label class="control-label tooltips" for="Description"><g:message
                                    code="Description" editable="false"/>


                              </label>
                              <input type="text" id="customproperty29" name="customproperty29"
class="form-control"
                                   value="${usersInstance?.customproperty29}" init-data= init-
data="${usersInstance?.customproperty29}" editable="false" readonly/>
                           </div>
                        </div>
</div>
```

- Sample field name – *Firstname*

```
<div class="col-md-6">
                           %{--First Name--}%
                           <div class="form-group">
                              <label class="control-label tooltips" for="firstname"><g:message
                                    code="FirstName.Default.label" editable="true"/><span
                                    style="color:red">*</span>
                                 <i href="javascript:;" data-html="true" class="tooltips fa fa-info-circle"
                                   data-placement="bottom"
                                   data-original-title="Allowed characters are a-z, A-Z, space, quote ('),
hyphen (-), dot (.) and commercial at (@)"></i>
                              </label>
```

```
                    <input type="text" id="firstname" name="firstname" class="form-control
UserMandatory"
                        value="${usersInstance?.firstname}" init-
data="${usersInstance?.firstname}" onkeyup='detectchange();'
                        pattern="[a-zA-Z @.'-]+" maxlength="64"
                        placeholder="<g:message code="FirstName.Default.label"
                                    editable="true"/>">
                  </div>
                </div>
```

2.1.2.1.1.2 How to create a Dropdown

- Sample field name –company name

```
<div class="col-md-6">
                  <div class="form-group">
                    %{--Company--}%
                    <label class="control-label"
                        for="Company"><g:message
                         code="Company" editable="false"/><span
                         style="color:red">*</span></label>


                    <select class="form-control select2_category UserMandatory"
id="customproperty20"
                         name="customproperty20"
                         data-placeholder="Select Company"
                                        editable="true"/>"
                         init-data="${usersInstance?.customproperty20}"
onchange="detectchangeselect();getCompany('getcompanyList')">
                      <option label=""

value="${usersInstance?.customproperty20}">${usersInstance?.customproperty20}</option>


                    </select>
                  </div>
                </div>
```

2.1.2.1.2 Accessing Saviynt Data

The Saviynt data can be accessed by creating the Run time analytics and invoking the Run time analytics from the custom GSP using the API "**${request.contextPath}/customize/getdbdata**" via Ajax call.

- Create Run Time analytics
- Invoking from GSP

2.1.2.1.2.1 Create Run Time analytics

Refer the below link for creating the Run time analytics.

https://docs.saviyntcloud.com/bundle/EIC-Admin-v23x/page/Content/Chapter17-EIC-Analytics/Managing-Analytics-v232-Earlier/Defining-Dynamic-Attributes-at-Run-Time.htm

To illustrate the functionality, assume this following query is used to get the list of company

Analaytics Name: *getcompanyList*

```
select c.CUSTOMERNAME as VALUE from customer c
```

2.1.2.1.2.2 Invoking Analytics from Custom GSP

The Saviynt exposed the following API ""**${request.contextPath}/customize/getdbdata"** using that Run time analytics can be triggered.

2.1.2.1.2.2.1 Signature and Description of getdbdata

The description and payload of API is

URL: ${request.contextPath}/customize/getdbdata

Type: POST

Parameters:{ "analyticsname" :<Name of the analytics>,

"attributes":{}

}

Samples for payload

- The following is sample for involving the Run time analytics that doesn't have any Run time parameters.

```
{
              "analyticsname": "getcompanyList",
              "attributes": {


              }
           }
```

- The following is sample for involving the Run time analytics that have any Run time parameters.

  Analytics Name: *getcompanyListbasedonorg*
  Query: *select c.CUSTOMERNAME as value from customer c where c.CUSTOMPROPERTY1 = '${* org *}'*

```
{
          "analyticsname": "getcompanyListbasedonorg",
          "attributes": {
                        "org":"saviynt"
          }
        }
```

2.1.2.1.2.2.2 How to define function and make ajax call to fetch data from analytics

```
  function getCompany(analyticsName) {


    console.log("Analytics name ::" + analyticsName);



    var data = {
      "analyticsname": analyticsName,
      "attributes": {
              }
    };
// Making an Ajax call to fetch the data from Analytics
    $.ajax({
      async: false,
      url: "${request.contextPath}/customize/getdbdata",
      type: "post",
      dataType: "html",
      data: {q: JSON.stringify(data)},
      success: function (returnData) {
        var data = JSON.parse(returnData)
        var jsonObjMap = data['result']


        if (jsonObjMap != undefined) {
```

```
        var singleEntry = jsonObjMap[0]
        var targetPropertyArray = targetProp.split(',')
        for (var i = 0; i < targetPropertyArray.length; i++) {
          var key = ''
          var targetPropertyData = targetPropertyArray[i].toLowerCase()

          if (targetPropertyData.indexOf('_') > 0) {
            key = targetPropertyData.split('_').join('');
          } else {
            key = targetPropertyData
          }
          var value = singleEntry[key.toUpperCase()]
          if (value != null && value != '') {
            var $selectFields1 = "";
              var documentData = "";
              if (targetPropertyData == "VALUE") {
                documentData = document.getElementById("customproperty20");
              }

              if (documentData.type == 'select-one') {
                $("#" + documentData).val(value.trim()).attr("selected", "selected");
              } else {
                documentData.value = value;
              }
          }
        }
      }
    },
    error: function (e) {
      alert(e);
      alert("Company Data retrieve error");
    }
  });
}
```

2.1.2.2 *Custom Validation*

The below is sample for doing the validation for throwing the alert for mandatory check. The logic is simple by getting all fields based on class name (custom classname) with simple JS can throw the error msg.

```
function MandatoryFieldsCheck() {
    var validate = true;
    var msg = '';
    $(".UserMandatory").each(function () {
      console.log("Attribute Name :: " + $(this).attr("name"))

      if ('' + $(this).attr("name") != 'undefined') {
        console.log("Attribute value :: " + $(this).val())
        if ($(this).val() == '' || $(this).val() == null) {
          console.log("exception is on :: " + $(this).attr("name"))
          $(this).css("border", "1px solid red");
          validate = false;
          if (msg == '') {
            msg = "Data is missing in one or more mandatory fields.";
          }
        }
        else {
          if (msg == '') {
            $(this).css("border", "1px solid #e5e5e5");
          }
        }
      }
    })
    return {msg: msg, validate: validate}
  }
```

2.1.2.3 *Create a derived field.*

The derived field is nothing but getting the value of child attribute based on other attributes. Example here populating the company code based on company name.

HTML code for Building UI

```
 <div class="col-md-6">
                          <div class="form-group">
                             %{--Company Code--}%
                             <label class="control-label" for="Company Code"><g:message
                                   code="Company Code" editable="true"/><span
                                   style="color:red">*</span></label>
                             <input type="text" id="customproperty19" name="customproperty19"
                                  class="form-control UserMandatory"
value="${usersInstance?.customproperty19}"
                                  init-data="${usersInstance?.customproperty19}"
onkeyup='detectchange();' placeholder="<g:message code="Company Code"
                                                   editable="true"/>" readonly>


                          </div>
                       </div>
```

JQ to set the value based company Name

```
  function getCompany(analyticsName) {


     console.log("Analytics name ::" + analyticsName);



     var data = {
        "analyticsname": analyticsName,
        "attributes": {
                   }
     };
// Making an Ajax call to fetch the data from Analytics
     $.ajax({
        async: false,
        url: "${request.contextPath}/customize/getdbdata",
        type: "post",
        dataType: "html",
        data: {q: JSON.stringify(data)},
        success: function (returnData) {
           var data = JSON.parse(returnData)
```

```
         var jsonObjMap = data['result']

         if (jsonObjMap != undefined) {
            var singleEntry = jsonObjMap[0]
            var targetPropertyArray = targetProp.split(',')
            for (var i = 0; i < targetPropertyArray.length; i++) {
               var key = ''
               var targetPropertyData = targetPropertyArray[i].toLowerCase()

               if (targetPropertyData.indexOf('_') > 0) {
                  key = targetPropertyData.split('_').join('');
               } else {
                  key = targetPropertyData
               }
               var value = singleEntry[key.toUpperCase()]
               if (value != null && value != '') {
                  var $selectFields1 = "";
                     var documentData = "";
                     if (targetPropertyData == "CompanyCode") {
                        documentData = document.getElementById("customproperty19");
                     }

                     if (documentData.type == 'select-one') {
                        $("#" + documentData).val(value.trim()).attr("selected", "selected");
                     } else {
                        documentData.value = value;
                     }
               }
            }
         }
      },
      error: function (e) {
         alert(e);
         alert("Company Data retrieve error");
      }
```

```
        });
    }
```

Note:

The highlighted in yellow indicates that using JS the ID is get and value is set to company code attribute based on company name.

2.1.2.4 *Populating/Setting the data for dynamic attributes created in Global config*

Here the dynamic attributes are created in Global configuration. Only datas are populated from custom GSP by invoking the Analytics based on UI event and set the value based on "ID" of the HTML attribute.

How to call the analytics from GSP and setting the value to attributes are explained in previous section so please refer the same for doing it.

In the below table will explain how the ID will be generated for dynamic attributes created from Global configuration.

Attribute Type	**Attribute Name**	**ID value**	**Generated ID**	**Comments**	**Sample code**
ENUM	department	selectEnum<DYNAMIC ATTRIBUTE NAME>	selectEnumDEPARTMENT	The name of the dynamic attributes will be prefixed with **selectEnum**	$("#selectEnumDEPARTMENT").select2("val", "");
Multi Select from List	department	selectMultilist<DYNAMIC ATTRIBUTE NAME>	selectMultilistDEPARTMENT	The name of the dynamic attributes will be prefixed with **selectMultilist**	$("# selectMultilistDEPARTMENT").select2("val", "");

String	department	<DYNAMIC ATTRIBUTE NAME>	DEPARTMENT	The generated ID will be same as name as dynamic attributes	document.getElementById(" DEPARTMENT ").value="dep1"

Based on ID of HTML element you set any property that is applicable for that HTML element. Like

- readonly,
- Hide
- show
- Disable
- Style and so on

As well do event-based trigger like

- Onclick
- Onfocus
- Onchange and so on

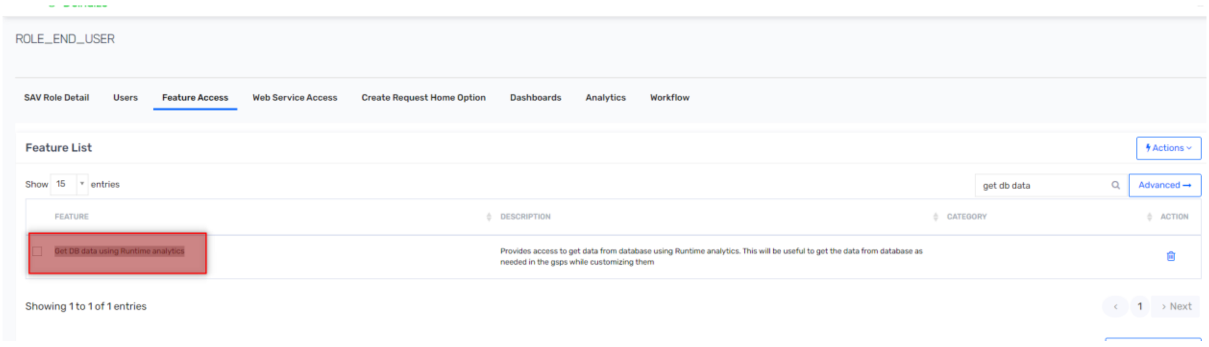
Note:

The property and events might be applicable based on the input type of HTML element.

2.1.3 Assign feature access to Sav role.

The permission required for accessing the API "GETDBDATA" is managed by feature in sav role.

The following features "Get DB data using Runtime analytics "needs to be added in Sav role to use the GETDBDATA


ROLE_END_USER

SAV Role Detail | Users | Feature Access | Web Service Access | Create Request Home Option | Dashboards | Analytics | Workflow

Feature List

Show 15 entries — get db data

FEATURE	DESCRIPTION	CATEGORY	ACTION
Get DB data using Runtime analytics	Provides access to get data from database using Runtime analytics. This will be useful to get the data from database as needed in the gsps while customizing them		

Showing 1 to 1 of 1 entries

2.1.4 Reference

https://docs.saviyntcloud.com/bundle/EIC-Admin-v23x/page/Content/Chapter06-EIC-Configurations/Configuring-File-Directories.htm https://docs.saviyntcloud.com/bundle/EIC-Admin-v23x/page/Content/Chapter17-EIC-Analytics/Managing-Analytics-v232-Earlier/Defining-Dynamic-Attributes-at-Run-Time.htm