WMS - Extended Configurability

DELMIA Apriso 2021 Technical Guide

Updated in Service Pack 1

DASSAULT SYSTEMES | The 3DEXPERIENCE® Company

Contents

Figures

1 Introduction

1.1 Overview

The Warehouse Management System uses Handlebars.js to create HTML templates. This solution applies to all Warehouse Operator applications. The guide describes how to configure two types of Handlebars.js templates which are used:

- In the Screens that display a **list of tiles** (for example, a list of Orders, Products, Containers with their properties)
- In the Screens that display **counters** and **pop-up windows**

1.2 Intended Audience

This guide is aimed at users with a good understanding of:

- DELMIA Apriso Process Builder and Screen Flows
- HTML, JavaScript, and Handlebars.js templating language

2 Technology Overview

2.1 Handlebars.js Templates

The Warehouse Management System solution makes extensive use of Handlebars.js to create HTML templates. A configuration scenario presented in this document requires you to have a good knowledge of this templating framework.

For more information, refer to the Handlebars website.

2.2 Json.NET

To display your own custom data in the Handlebars.js templates, you need to write user formulas in C# to modify the incoming JSON. The Json.NET library is used to add and reorganize data.

For more information on Json.NET, refer to the Newtonsoft documentation.

In order to use the Json.NET library, you must enable "Script extensions" in the User Formula editor.

3 Extension Points Used to Change UI Configuration

Every Warehouse Operator application contains two Extension Points which are used to change the app UI configuration (that is a list of tiles, counters, and pop-up windows):

1. **Templates Extension Point**.
2. **UI Configuration Extension Point**.

The naming convention for the listed Extension Points is:

- **[AppNameAbbreviation]_Templates_Extension** (e.g. INV_Templates_Extension)
- **[AppNameAbbreviation]_UI_Configuration_Extension** (e.g. INV_UI_Configuration_Extension_)

If you implement changes in one app, they are not going to affect any other app.

3.1 Templates Extension Point

The **Templates Extension Point** enables linking a new template to any Warehouse Operator app, which makes the template visible for the user. The Extension Point is located in every app module and is used with lists of tiles, counters, and pop-up windows.

3.1.1 How to Use the Templates Extension Point

You can use the **Templates Extension Point** to:

- Define a new template
- Link an existing Template Operation

Define a new template

To define a new template modify a Display Step in the **Templates Extension Point** in the following way:

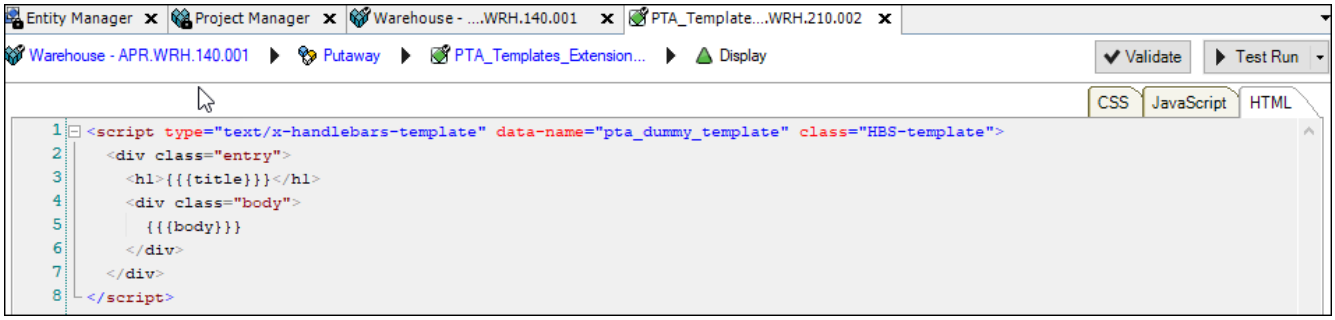

Figure 1 The Templates Extension Point – Display Step modification example

The `<script>` tag must contain:

1. `data-name` property (it acts as a template name).
2. `class="HBS-template"`.

Link a Template Operation

A **Template Operation** contains the template that returns HTML code based on the JSON argument. The Template Operation is located in the TemplatesHBS module.

The properties of the Template Operation are as follows:

- The Operation name: **[name]_HBS**
- The Operation subtype: **View**

You can provide more than one template but it is recommended that there is only one template per one Operation.

To create and link a Template Operation:

1. Configure a View Operation with one Step.
2. In the Step provide the template in the `<script>` tag. The `<script>` tag must contain:
   a. `data-name` property (it acts as a template name).
   b. `class="HBS-template"`.
3. Link the View Operation as a suboperation to the **Templates Extension Point**.

3.2 UI Configuration Extension Point

The **UI configuration Extension Point** allows you to edit the UI of any Warehouse Operator app by modifying JSON attributes. The default JSON configuration object is defined in the **[[AppNameAbbreviation]_UI_Configuration** Operation. This JSON object defines the configuration of all the elements that can be modified for each Screen.

3.2.1 How to Use the UI Configuration Extension Point

The **UI Configuration Extension Point** gets and returns a JSON object as a string. To change UI:

1. Parse JSON object via Json.net library.
2. Replace the attributes you want to change based on the JSON structure described below.

   To modify the "templateName" JSON attribute use the value of template `data-name` property that you defined or linked in the **Templates Extension Point**.

3.2.2 JSON Configuration Object Structure

Every Warehouse Operator app has its own JSON object that contains subobjects for every configurable Screen (that is, a Screen that uses Handlebars.js templates to display the list of Entity tiles). Each subobject can contain:

- **EndlessScroll** subobject for Screens with a list of tiles
- **CounterData** subobject for Screens with counters

- **[name]_Popup** subobject for Screens with pop-up windows

EndlessScroll

EndlessScroll subobject describes the details of the process responsible for how a list of tiles is displayed. The process involves the following Operations:

1. An **Ajax Arguments Initialize** Operation – a type of an Operation that returns the basic data for Ajax request. The Operation is located in the EndlessScroll module.

   The properties of the Ajax Arguments Initialize Operation are as follows:

   - The Operation name: **[ScreenName]_AJAX_args_[name]**
   - The Operation subtype: **Initialize**
2. An **Ajax Response Data** Operation – a type of an Operation that returns the full data you want to display (based on the basic data returned from the **Ajax Arguments Initialize** Operation). The Ajax Response Data Operation is located in the EndlessScroll module.

   The properties of the Ajax Response Data Operation are as follows:

   - The Operation name : **[ScreenName]_AJAX_resp_[name]**
   - The Operation subtype: **Data**

   The Outputs of both Operations must be in the form of a JSON object that can contain any of the Process Builder data type (scalar or list).

The process overview:

The **Ajax Arguments Initialize** Operation defines the arguments for the **Ajax Response Data** Operation.

The **Ajax Response Data** Operation based on the arguments from the **Ajax Arguments Initialize** Operation creates an argument for a template. This argument is a one level JSON with scalar/array attributes. On the client side this JSON is parsed and transformed (via JavaScript) to an object with “items” attribute: list of JSON subobjects (each per tile).

The "templateName"JSON attribute must contain the value of the defined or linked Template `data-name` property.

JSON configuration object structure for EndlessScroll:

```
{
  "[Screen_Name]": {
    "EndlessScroll": {
      "AJAX_args": "[ScreenName]_AJAX_args_[name]",
      "AJAX_resp": {
        "ItemsGroups": {
          "[Entity_Name]": {
            "ExtendFnName": "[ScreenName]_AJAX_resp_[name]",
            "templateName": "[name]"
          }
        }
      }
    }
  }
}
```

CounterData and Popup

CounterData and **[name]_Popup** subobjects make it possible to directly modify data (using **ExtendFnName** Operation) and template name (using **templateName**) which are both passed to a counter or pop-up View. There are no JSON modifications on the client side.

```
{
  "[Screen_Name]": {
    "CounterData": {
      "ExtendFnName": "[PB_Operation_Name]",
      "templateName": "[name]",
    },
    "[Popup_Name]_Popup": {
      "ExtendFnName": "[PB_Operation_Name]"
      "templateName": "[name]",
    }
  }
}
```

4 Configuration Scenario

How to Modify an Entity Tile

This scenario illustrates how to modify a tile in the INV- 20 Screen in the Inventory Operations application. The scenario is applicable to all the Screens across all the Warehouse Operator applications.

1. Link the **WRH_Echo_HBS** and **WRH_List_Echo_HBS** Template Operations to the **INV_ Templates_Extension** Extension Point.

   The **WRH_Echo_HBS** and **WRH_List_Echo_HBS** Template Operations have been created to simplify the debugging process. They convert arguments to text and they are both located in the TemplatesHBS module.

2. In the **INV_UI_Configuration_Extension** Extension Point use the User Formula editor to define the code that changes the value of the **"templateName"** JSON attribute located in the EndlessScroll subobject to **"list_echo"**:

   ```
   var obj = Newtonsoft.Json.Linq.JObject.Parse(JSON);

   obj["INV-20"]["EndlessScroll"]["AJAX_resp"]["ItemsGroups"]["Containers"]["templateName"]
   = "list_echo";

   NEW_JSON = obj.ToString();
   ```

   As a result , the INV-20 Screen will display arguments of the template:

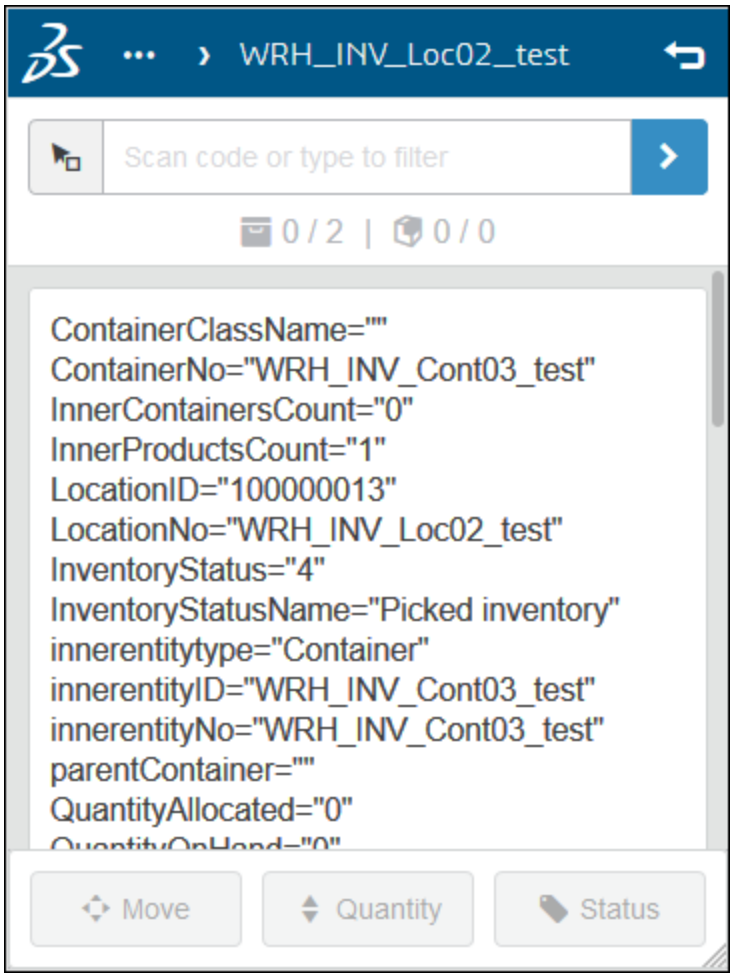

Figure 2 INV-20 Screen – list of tiles: template change

3. Create a new **Ajax Response Data** Operation called **AJAX_resp_test** in the Warehouse PB Project (for configuration details, see Ajax Reponse Data Operation). The Operation should add additional arguments (**EP_Scalar_data** and **EP_Array_data**) to the template.

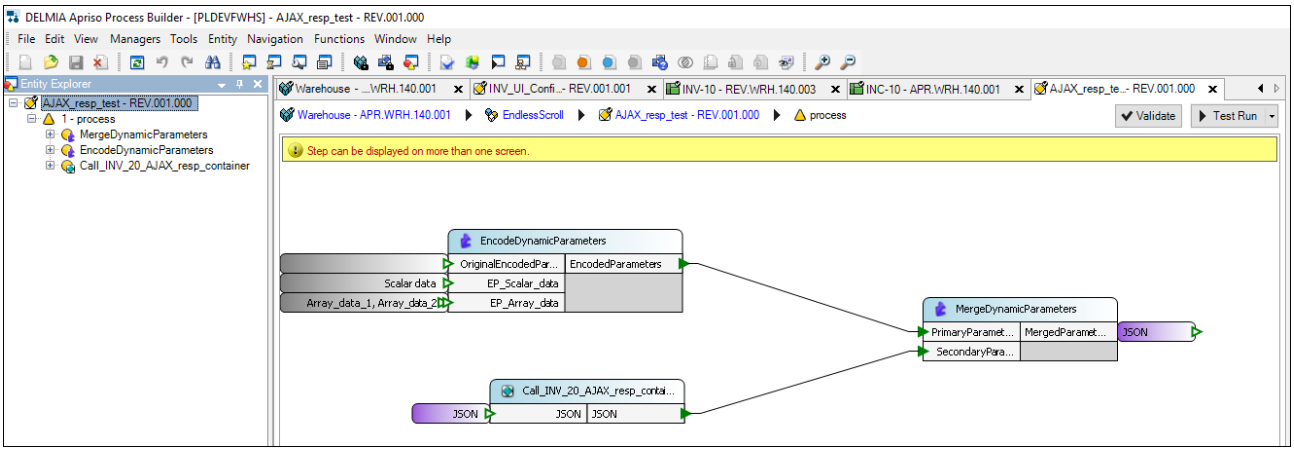

Figure 3 Ajax Response Data Operation with additional arguments

4. In the **INV_UI_Configuration_Extension** Extension Point use the User Formula editor to define the code that changes the value of the **"ExtendFnName"** located in the EndlessScroll subobject to "**AJAX_resp_test**":

```
var obj = Newtonsoft.Json.Linq.JObject.Parse(JSON);

obj["INV-20"]["EndlessScroll"]["AJAX_resp"]["ItemsGroups"]["Containers"]["templateName"]
= "list_echo";
obj["INV-20"]["EndlessScroll"]["AJAX_resp"]["ItemsGroups"]["Containers"]["ExtendFnName"]
= "AJAX_resp_test";

NEW_JSON = obj.ToString();
```

The result will be the same as in step 2, but there will be two additional arguments (**EP_Scalar_data** and **EP_Array_data**) of the template. Note that the array will be divided across the tiles.

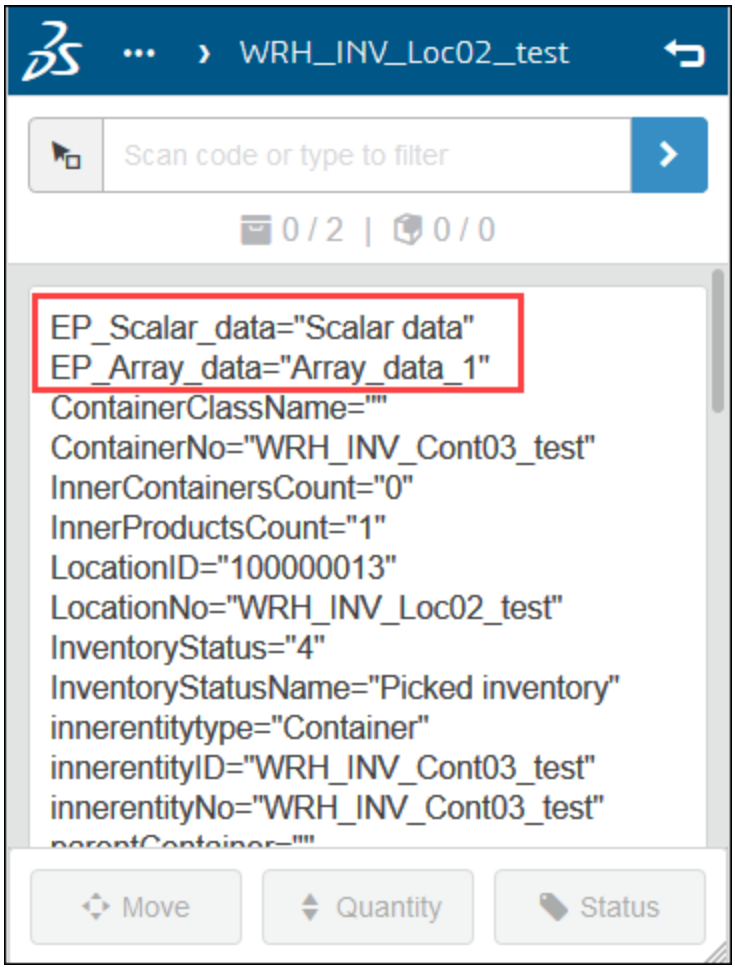

Figure 4 INV-20 Screen – list of tiles: template and data change

5 References

Internal Documentation

1. ***WMS - Conceptual Design***

   Provides the conceptual design of Warehouse Management System that includes a configuration specification, functional description, and Process flow in the respective business areas.

2. ***WMS - Technical Design***

   Describes the technical design of the Warehouse Management System (WMS). The document includes descriptions of Screen Flows, Standard Operations, Extension Points, System Parameters, Determinations, etc.

3. ***WMS - Counting Detailed Design***

   Describes the detailed design of Counting for Warehouse Management System. The document includes descriptions of various Process aspects such as the prerequisites, assumptions, Process flow description, and Process results.

4. ***WMS - Inventory Operations Detailed Design***

   Describes the detailed design of the Inventory Operations for Warehouse Management System. The document includes descriptions of various Process aspects such as the prerequisites, assumptions, Process flow description, and Process results.

5. ***WMS - Put Away Detailed Design***

   Describes the detailed design of Put Away for Warehouse Management System. The document includes descriptions of various Process aspects such as the prerequisites, assumptions, Process flow description, and Process results.

6. ***WMS - Receiving Detailed Design***

   Describes the detailed design of Receiving for Warehouse Management System. The document includes descriptions of various Process aspects such as the prerequisites, assumptions, Process flow description, and Process results.

7. ***WMS - Shipping Detailed Design***

   Describes the detailed design of Shipping for Warehouse Management System. The document includes descriptions of various Process aspects such as the prerequisites, assumptions, Process flow description, and Process results.

8. ***Process Builder Help***

   Provides an overview of DELMIA Apriso Process Builder (PB) and information on installing and using the application. This Help describes the user interface elements, entity maintenance, available Business Controls, and management of Processes, Operations, and Screen Flows.

3DS Support Knowledge Base

If you have any additional questions or doubts not addressed in our documentation, feel free to visit the **3DS Support Knowledge Base** at https://support.3ds.com/knowledge-base/.