WEBCON

BPS 2023.1

Changes to SDK & API

1. Introduction

The following document contains a list of changes in WEBCON BPS SDK and API when compared to version 2022.1. These changes break code compatibility or change how existing code works.

2. Changes to SDK

2.1. Introducing asynchronous methods

Available SDK methods, both those in base classes and tool classes, have been changed to asynchronous versions.

2.1.1. Migrating methods that return no results.

Example of migrating methods that return no results, e.g. CustomAction

```
public override void Run(RunCustomActionParams args)
{
    try
    {
        var id = GetDocumentID();
        DoSomeLogic(id);
    }
    catch (Exception ex)
    {
        args.HasErrors = true;
        args.Message = ex.Message;
        args.LogMessage = ex.ToString();
    }
}
private int GetDocumentID()
{
    return 0;
}
```

If all methods used in the code are synchronous:

```
public override Task RunAsync(RunCustomActionParams args)
{
    try
    {
        var id = GetDocumentID();
        DoSomeLogic(id);

    }
    catch (Exception ex)
    {
        args.HasErrors = true;
        args.Message = ex.Message;
        args.LogMessage = ex.ToString();
    }
    return Task.CompletedTask;
}
```

```
private int GetDocumentID()
{
    return 0;
}
```

If the code includes some asynchronous methods:

```
public override async Task RunAsync(RunCustomActionParams args)
{
    try
    {
        var id = await GetDocumentIDAsync();
        DoSomeLogic(id);

    }
    catch (Exception ex)
    {
        args.HasErrors = true;
        args.Message = ex.Message;
        args.LogMessage = ex.ToString();
    }
}


private async Task<int> GetDocumentIDAsync()
{
    var id = await GetDataAsync();
    return id;

}
```

2.1.2. Migrating methods that return results

Example of migrating methods that return results, e.g. CustomBusiness Rule

```
public override EvaluationResult Evaluate(CustomBusinessRuleParams args)
{
    var result = GetData();
    return EvaluationResult.CreateStringResult(result);
}

private string GetData()
{
    return "";
}
```

If all methods used in the code are synchronous:

```
public override Task<EvaluationResult> EvaluateAsync(CustomBusinessRuleParams args)
        {
            var result = GetData();
            return  Task.FromResult(EvaluationResult.CreateStringResult(result));
        }

        private string GetData()
        {
            return "";
        }
```

If the code includes some asynchronous methods:

```
public override async Task<EvaluationResult> EvaluateAsync(CustomBusinessRuleParams
args)
        {
            var result = await GetDataAsync();
            return EvaluationResult.CreateStringResult(result);
        }

        private async Task<string> GetDataAsync()
        {
            var data = await ExetuteSqlCommanAsync();
            return data;
        }
```

2.2. Changing tool classes to instance classes

Definitions of tool classes that used to be static are now instance, these are:

- DataSourcesHelper
- SqlExecutionHelper
- TomeZoneHelper
- ConnectionsHelper

To create instances for these class, it is necessary to provide the context in which the plugin works.
It is available in most methods in the execution parameters in the Context property, e.g.:

```
public override void Run(RunCustomActionParams args)
{
        var context = args.Context;
}
```

Additionally, in CustomDataSource and FormFieldExtension classes it is available in the base class, e.g.:

```
public class FormFieldExtension2: FormFieldExtension<PluginConfiguration>
{
    public override object OnDBValueSet(object valueToSet)
    {
        var context = base.Context;
        ...
    }
}
```

2.3. Creation methods

For BpsTransactionScope and NewAttachmentData classes, the public constructor has been replaced with a different creation method, accordingly:

BPSTransactionScope.Create (BaseContext context)

DocumentAttachmentsManager. GetNewAttachment (string name, byte[] content)

2.4. Changes to culture in converting field and column values on the form

Plugins implemented the SetValue(object value) and SetValue(object value, CultureInfo culture) methods. In the new SDK version, these two methods have been replaced with one SetValue(object value, CultureInfo culture = null). This new implementation will change the conversion of string data if the culture is not given for **dateTime** and **decimal** fields.

- In the previous version, setting a value without providing a culture for DateTimeField and DecimalField resulted in an attempt to convert in the current culture. A failed conversion would initiate a second attempt using InvariantCulture.
- For DecimalCell item list columns, setting a value without providing a culture resulted in an attempt to convert in the current culture. A failed conversion would result in a null value being set.
- For new SDK, if the culture is not provided and if the conversion in the current culture fails, the SDKFormatException error will be shown. To convert the value in InvariantCulture, provide the culture in the argument of method SetValue(…).

2.5. Changes to attachment groups

In AttachmentsGroup class constructors, it is necessary to provide a non-empty DisplayName. Attempting to create a category with an empty DisplayName will return the SDKArgumentException error.

2.6. Removal of GenerateDocx method

SDK.Tools was used to implement the GenerateDocx method. This method is no longer available in the new SDK. It can be substituted with a regular WORD document generation action or a custom component for generating docx files.

2.7. Privileges via SecurityManager

Privileges can be changed via SecurityManager by using IncreaseSecurity and ReduceSecurity methods. They were found under one ChangeSecurityParams class in the old SDK version. In the new version, it is now two separate classes: IncreaseParams and ReduceParams.

By default, privileges are reduced to read-only after path transition. The increase in privilege level can be made permanent using the IsPermanent property in the IncreaseParams class.

2.8. Default path via DocumentsManager

The PathID property implemented in StartNewWorkFlowParams and MoveDocumentToNextStepParams classes is nulled in the new SDK. Using DocumentsManager without providing a PathID will cause it to use the default path set for the given step.

2.9. Changes in DocumentData class

The Signature property implemented in the DocumentData class has been renamed toInstanceNumber.

2.10. Changes to variable evaluation for bool fields

The values to which **bool** type fields are compared have been changed. If the value in the plugin configuration is assigned to the **string** property, it used to be compared to “True” and “False”. It will now be “true” and “false”.

If the plugin compares values and takes into account letter case, it will be necessary to migrate the code.

Previously:

```
[ConfigEditableText(DisplayName = "UpdadeData", DefaultText = "")]
public string NeedUpdate { get; set; }
….

if (Configuration.NeedUpdate == "True")
…
```

Currently:

```
[ConfigEditableText(DisplayName = "UpdadeData")]
public bool? NeedUpdate { get; set; }
 ….

if (Configuration.NeedUpdate == true)
…
```

3. Changes to BPS API

3.1. New version of BPS Rest API

- A new version of BPS Rest API, known as version 5.0, has been added.
- Rest API version 3.0 has been designated as outdated, and it is recommended to no longer use it. It will be removed in the next version of WEBCON BPS.
- Rest API version 2.0 has been removed.

3.2. Model change in choice fields in elements endpoints

In REST API 5.0 an beta, the model for a choice field with multiple choices has been changed from:

```
{
  "value": {
    "choices": [
      {
        "id": "1",
        "name": "First Value"
      }
    ],
    "other": "Typed in value"
  }
}
```

To:

```
{
  "value": [
    {
      "id": "1",
      "name": "First Value"
    },
    {
      "id": "__Other__",
      "name": "Typed in value"
    }
  ]
}
```

The “other” property has been removed. A custom value typed in by the user is now part of the table with the other values. It’s unique ID will always be __Other__. The behavior of the “svalue” field has not been changed.

3.3. Changes to handling of read-only fields in JavaScript mode

Get/elements endpoints with the expand=formLayout flag, and get/formlayout in all API versions will now return whether the field is in read-only with JavaScript enabled, in addition to the standard read-only mode.

In all API versions, post and patch/elements endpoints now allow fields to be edited when they are in “read-only but editable by JavaScript” mode.

3.4. Changes to naming Item list models in oData API

In the beta version of oData API, the way in which model names for Item lists are built has been changed. From now, each model will have the List_ prefix. For example, an Item list called Subelements with the Id of 988 that was previously named Subelements_988 will now be named List_Subelements_988 instead.

This means exiting queries need to be reconfigured.