Integration of Form Field Extensions with WEBCON BPS is possible through the JS API, this article will go over available functions.
To be able to use the JS API you need to do a few things first:

1. Create two variables: __WEBCON_PORTAL_ADDRESS__ that should take the address of the server where the portal is located and __WEBCON_EXTERNAL_CONTROL_ID__ that should take id of the External form field where the page will be placed.

2. Include webcon.min.js file which is available for download here in iife folder.

Form appearance and behavior

webcon.window.setHeight(height) – this function is used to change the External form field height.

```
webcon.window.setHeight(500);
//or
webcon.window.setHeight('500px');
```

webcon.fields.showField(fieldName) and webcon.fields.hideField(fieldName) - these functions are used to hide/show fields on the form.

```
webcon.fields.hideField('AttText9');
webcon.fields.showField('AttText9');
```

webcon.fields.showSubelementColumn(subelementsControlId, columnName)

webcon.fields.hideSubelementColumn(subelementsControlId, columnName) – these functions are used to show/hide a particular column of an item list.

```
webcon.fields.showSubelementColumn(30, 'DET_Att1')
webcon.fields.hideSubelementColumn(30, 'DET_Att1')
```

Other functions use to hide/show form elements:

webcon.fields.showSubelementControl(subelementsControlId)
webcon.fields.hideSubelementControl(subelementsControlId)
webcon.fields.showTabPanel(id)
webcon.fields.hideTabPanel(id)
webcon.fields.showTab(id)
webcon.fields.hideTab(id)
webcon.fields.selectTab(id)

webcon.fields.hideGroup(id)
webcon.fields.showGroup(id)
webcon.fields.expandGroup(id)
webcon.fields.collapseGroup(id)
webcon.fields.showChart(id)
webcon.fields.hideChart(id)
webcon.fields.showSqlGrid(id)
webcon.fields.showSqlRow(id)
webcon.fields.hideSqlGrid(id)
webcon.fields.hideSqlRow(id)
webcon.document.hideComment()
webcon.document.showComment()
webcon.document.showSecurities()
webcon.document.showOcrTrainButtons()
webcon.document.hideOcrTrainButtons()

webcon.document.checkIfAttExists(regex, callback) - this function is used to check if an attachment with a specific name exists attached to the instance. The first parameter of the function is a regular expression that specifies the name of the attachment to be checked for. Second parameter is the callback function to which the result will be passed. The result is true or false.

In these examples, the result (true or false) will be logged in the console, depending on whether the attachment being searched for exists:

```
webcon.document.checkIfAttExists('^DocumentName[.]docx$', logInfo)
```

with the callback function:

```
function logInfo(value){
  console.log(value);
}
```

The callback function can of course also be provided in this way:

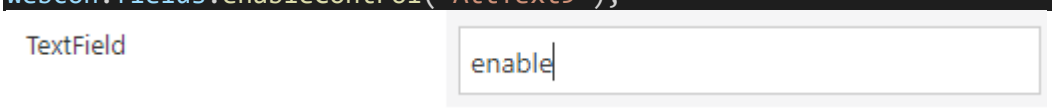
```
 webcon.document.checkIfAttExists('^DocumentName[.]docx$', function(value){
        console.log(value);
      })
```

Using this method, we can also check if a file with the given extension exists, If there is any attachment with the extension .pdf, the function returns true.

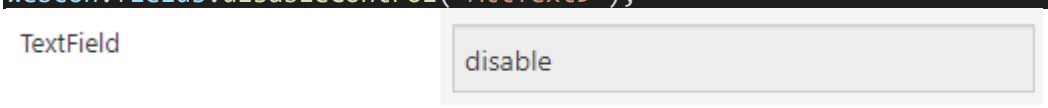
```
 webcon.document.checkIfAttExists('[.]pdf$', function(value){
        console.log(value);
      })
```

webcon.fields.enableControl(fieldName) and **webcon.fields.disableControl(fieldName) –** these functions are used to change whether the field is enabled or disabled.

```
webcon.fields.enableControl('AttText9');
```




```
webcon.fields.disableControl('AttText9');
```



Other functions use to edable/disable form elements:

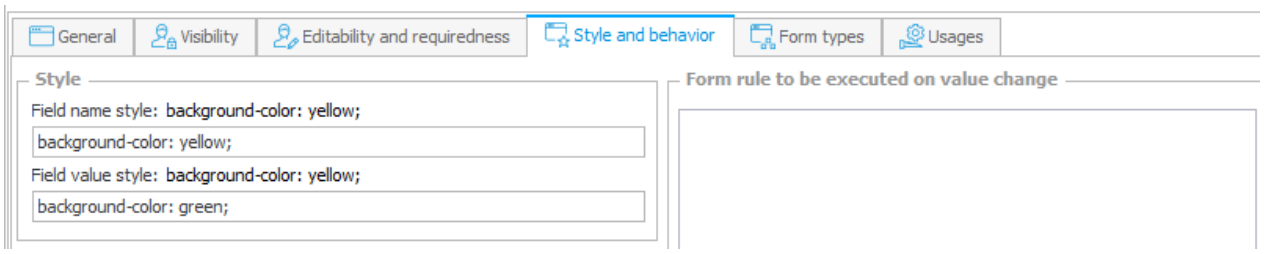
webcon.document.enableRootPanel()
webcon.document.disableRootPanel()

webcon.fields.isFormFieldVisible(fieldName, id, callback) – the function returns true or false depending on whether the field is visible or not

```
webcon.fields.isFormFieldVisible('AttText9', 85, function(value){
          console.log(value);
      });
```

webcon.fields.setControlStyle(fieldName, style) and webcon.fields.setLabelStyle(fieldName, style) –

With these functions you can set the same options as here in Designer Studio.




```
webcon.fields.setLabelStyle('AttText9', 'background-color: yellow')
webcon.fields.setControlStyle('AttText9', 'background-color: green');
```

Other functions for setting the style of form elements:

webcon.fields.setLabelStyle(fieldName, style)

Other functions to change the appearance of form elements

webcon.fields.selectTab(id)
webcon.fields.expandGroup(id)
webcon.fields.collapseGroup(id)

webcon.events.onHasValueValidation(callback) – the callback function must return true or false. If true is returned then the control has a value.

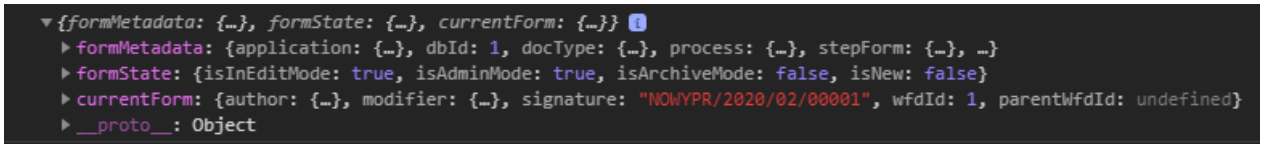
```
window.webcon.events.onHasValueValidation(function() {
    var hasValue = true;
    //logic defining whether a control has a value
    return hasValue;
    });
```

Workflow instance data and form metadata

webcon.document.getFormInfo(callback) – The callback function will be passed JSON with information about current form.

```
      webcon.document.getFormInfo(function(info){
        console.log(info);
      });
```

Information logged in the console



webcon.document.getContext(callback) - The callback function will be passed JSON with Portal language, userLogin and userName.

```
webcon.document.getContext(function(info){
        console.log(info);
      });
```

webcon.fields.getMode(fieldName, callback) - The callback function will be passed text 'Standard' or 'Custom' depending on field mode.

```
webcon.fields.getMode('AttText9',function(info){
        console.log(info);
      });
```

webcon.fields.setValue(fieldName, value) and
webcon.fields.getValue(fieldname, callback) - these functions are used to get/set form field values.

```
webcon.fields.setValue("AttText9", textValue);
```

```
webcon.fields.getValue("AttText9", function(attValue) {
        console.log(attValue);
      }):
```

webcon.fields.setTypedValue(fieldName, value) and

webcon.fields.getTypedValue(fieldName, callback) - these functions are used to set/get typed fields like date or decimal.

```
webcon.fields.setTypedValue('AttDateTime1',
{ value: new Date(2010, 0, 30, 13, 45)};

webcon.fields.setTypedValue('AttDecimal1',  { value: 10.23 });
```

```
webcon.fields.getTypedValue('AttDateTime1', function(value){
       console.log(value);
      });
```

webcon.fields.setSubValue(subelementsControlId, rowNumber, columnName, value, forceOnReadonly) and
webcon.fields.getSubValue(subelementsControlId, rowNumber, columnName, callback)

Values can only be set in rows that already exist. Columns set to readonly can be modified after passing true to the **forceOnReadonly** parameter. Rows are numbered starting at 1.

```
webcon.fields.setSubValue(30, 1, 'DET_Att1', 'value to set', true);
```

```
webcon.fields.getSubValue(30, 1, 'DET_Att1', function(value){
        console.log(value);
    });
```

webcon.fields.setSubTypedValue(subelementsControlId, rowNumber, columnName, value) and

webcon.fields.getSubTypedValue(subelementsControlId, rowNumber, columnName, callback) – these functions are used to set/get typed item list values. Rows are numbered starting at 1.

```
webcon.fields.setSubTypedValue(32, 1, 'DET_Value1',  { value: 10.23 });
```

```
webcon.fields.getSubTypedValue(32, 1, 'DET_Value1', function(value){
        console.log(value);
    });
```

webcon.fields.subelementCountRows(subelemId, callback)

```
webcon.fields.subelementCountRows(30, function(value){
    console.log(value);
  });
```

webcon.fields.deleteSubelementRows(subelemId, list) – delete rows passed in this parameter from an item list

```
webcon.fields.deleteSubelementRows(30,[2,3])
```

Other functions for accessing item list data

webcon.fields.subelementHasRows(subelemId, callback)
webcon.fields.subelementInitialization(subelemId)
webcon.fields.subelementDelete(subelemId) – delete all item list rows

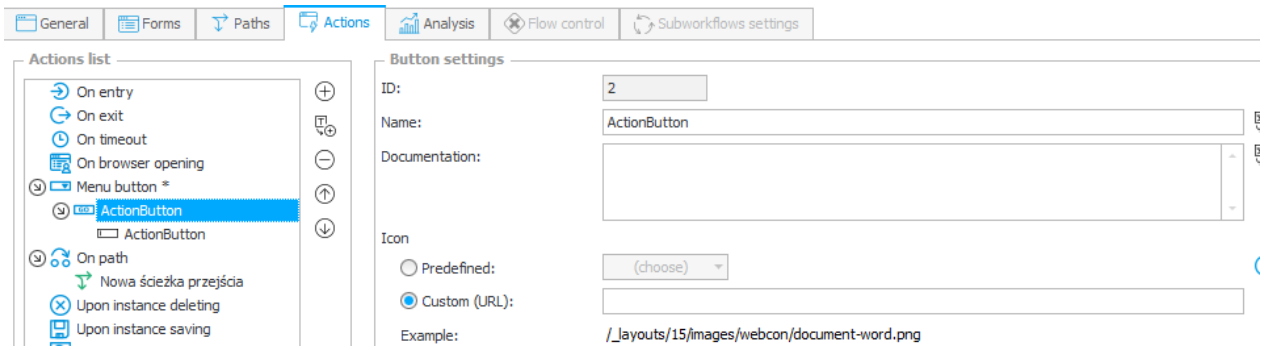
webcon.document.getSelfConfiguration(callback) - The callback function will be passed JSON with current form field configuration

```
webcon.document.getSelfConfiguration(function(configuration){
        console.log(configuration);
    });
```

Other functions to get field configuration

webcon.document.getControlConfiguration(fieldName, callback)

webcon.document.invokeMenuAction(actionButtonId) - Using this function, you can trigger an action set to trigger on 'Menu button' by entering the button id.




```
webcon.document.invokeMenuAction(2);
```

Data type conversion

webcon.document.stringToDate(strDate, callback)

```
webcon.document.stringToDate('10/10/2020', function(date){
       console.log(date); //Sat Oct 10 2020 00:00:00 GMT+0200

     })
```

webcon.document.dateToString(dateObject, format, callback) – take a Date object and pass to the callback function to receive a string representation.

```
var date = new Date(2020, 0, 10);
  webcon.document.dateToString(date, "pl", function(value){
```

```
        console.log(value);//2020-01-10
      })
  webcon.document.dateToString(date, "en", function(value){
        console.log(value);//1/10/2020
      })
```

webcon.document.convertFloatToString(value, precision, callback)

```
webcon.document.convertFloatToString(10.1010, 2, function(value){
        console.log(value);//10,10
      });
```

webcon.document.convertStringToFloat(value, callback)

```
webcon.document.convertStringToFloat("10,1010", function(value){
        console.log(value);//10.101
      });
```

webcon.document.getPairName(idNameString, callback) and **webcon.document.getPairID(idNameString, callback)** – The callback function will be passed only the ID or name.

```
 webcon.document.getPairName('35#Name', function(value){
        console.log(value); //Name
      })

 webcon.document.getPairID('35#Name', function(value){
        console.log(value); //35
      })
```