Wizard Controls API

Wizard controls can be divided into two categories, simple and collection controls. With simple controls, you can set and get values by simply calling getControlValue and setControlValue API. With collection controls, you need to call getControlSelectedItems API to get selected items first before updating any of the item. Each selected item is represented by ItemControl. ItemControl can be used to set and get column values by calling getValue and setValue API.

For setting and getting the values from the controls, controlName can be passed to the API methods. When you have multiple controls with the same name, to identify a specific control a qualified path has to be passed in place of the controlName for the API methods.

QualifiedPath for controls used in the group control:

```
pageName/sectionName/groupName/controlName
```

QualifiedPath for controls used in the panel control:

```
pageName/sectionName/groupName/panelControlName/pageName/sectionName/groupName/
rolName
```

Example:

```
appStudioWizard.getControlValue("pageName/sectionName/groupName/controlName");
```

Wizard:

Wizard object can be accessed in the script with the keyword AppStudioWizard. It provides all the APIs to interact various controls in the Wizard.

getItemId

Get itemId of the item loaded in the Wizard.

```
var itemId = appStudioWizard.getItemId()
```

getItemType

Get itemType of the item loaded in the Wizard.

```
var itemTypeName = appStudioWizard.getItemType()
```

getItemConfigId

Get itemConfigId of the item loaded in the Wizard.

```
var itemConfigId = appStudioWizard.getItemConfigId()
```

isItemNew

Returns true if the item in the Wizard is never saved to server.

```
var isItemNew = appStudioWizard.isItemNew()
```

isItemEditable

Returns true if the item in the Wizard is editable (in edit mode).

```
var isItemEditable = appStudioWizard.isItemEditable()
```

Section and group controls in the page can have their own ItemType context. Following API returns true if the associated item of the layout control (Section or Group) is editable (in edit mode).

```
var isLayoutEditable = appStudioWizard.isItemEditable(layoutControlName)
```

isControlUpdated

Checks control with the given name is updated (has unsaved changes)

```
var isControlUpdated = appStudioWizard.isControlUpdated(controlName)
```

reload

Reload whole wizard with the latest item details from the server by discording all the local changes

```
appStudioWizard.reload();
```

reloadControl

Reload specific collection control data from the server by discording all the local changes

```
appStudioWizard.reloadControl(controlName);
```

showDialogFlyout

Shows the flyout given the flyout name.

```
appStudioWizard.showDialogFlyout(pageDialogName);
```

showInsertFlyout

Shows the associated InsertItem flyout on the collection controls

```
appStudioWizard.showInsertFlyout(controlName);
```

showWorkflowActivityFlyout

Shows WorkflowActivity flyout given the activityId, workflowId, itemId and tableControlName.

```
appStudioWizard.showWorkflowActivityFlyout(activityId, workflowId, itemId, tableControlName);
```

Aras Object

Aras object can be accessed in the script with the keyword aras. It provides all the APIs to interact with the Innovator on the client side. It can be used only for the templateViews used for showing the Wizard in the Innovator UI. It doesn't work if the templateView is used for showing UI in the Portal.

Simple Controls

Following API works with all simple controls like Text, List, Checkbox, RichText, Item and Items.

getControlValue

Get the value of the simple control based on its name.

```
var controlValue = appStudioWizard.getControlValue(controlName);
```

setControlValue

Set value for the simple control based on its name.

```
appStudioWizard.setControlValue(controlName, controlValue);
```

Collection Controls

The following API works with collection controls like Table, Worksheet, TreeTable, Structure, DashboardTable, DashboardWorksheet. Collection controls API can only be used when they are visible on the current page.

getControlSelectedItems

Gets list of selected items from a collection control given the collection control name.

```
var controlItems = appStudioWizard.getControlSelectedItems(collectionControlName);
```

setControlSelectedItems

Selects the items in the collection control given collection control name and the array of relationshipIds if the table shows relationship data. For dashboard table, dashboard worksheet, and reference items table, itemIds can be passed directly.

```
const arrayOfIds = ["131706F6B80D47A5A68ED00BAF210CC8"]
appStudioWizard.setControlSelectedItems(collectionControlName, arrayOfIds);
```

getControlItem

Gets item based on the relationshipId or itemId given the collection control name. RelationshipId should be passed if the table represents Relationships. ItemId should be passed if the table represents Dashboard or References.

```
var itemControl = appStudioWizard.getControlItem(collectionControlName, relationshipId/itemId);
```

getControlItemByRelatedId

Gets item based on relatedItemId given the collection control name. This will only work for the collection control that displays relationship(non-null) data.

```
var controlItems = appStudioWizard.getControlItemByRelatedId(collectionControlName, relatedItemId);
```

addControlItem

Adds an item in the collection control right after the passed relationshipId given the collection control name and returns the inserted item. This will only work for the table, worksheet and tree table that show relationship data. ItemData passed to the function should contain all the properties and their values.

```
const itemData = {
properties: {
```

```
address: "Sample Data",
},
relatedItem: {
properties: {
name: "Sample Item",
description: "Sample Desc",
},
},
};
```

```
var addedControlItem = appStudioWizard.addControlItem(collectionControlName, relationshipId, itemData);
```

insertControlItem

Inserts an item in the collection control right after the passed relationshipId given the collection control name and returns the inserted item. This will only work for the table, tree-table and worksheet that show relationship data.

```
appStudioWizard.insertControlItem(collectionControlName, relationshipId, insertItemId).then((insertedItemControl) => {
// accessing inserted ItemControl and setting its value
insertedItemControl.setValue(columnPropertyName, cellValue);
});
```

deleteControlItem

Deletes an item in the collection control given collection control name and relationshipId for relationship table. In case of data shown based on reference items or items, itemId can be passed.

```
appStudioWizard.deleteControlItem(collectionControlName, relationshipId/itemId);
```

setCommandDropdownOptions

Each collection control comes with its own toolbar with commands. If there is any dropdown command in the toolbar, options for that dropdown can be set using this function given the collection control name and the command name.

```
var options = [
{ label: 'optionLabel', value: 'optionValue' },
{ label: 'optionLabel2', value: 'optionValue2' },
];
```

```
appStudioWizard.setCommandDropdownOptions(collectionControlName, commandName, options);
```

filterCollectionItems

This API works only with Table, Worksheet and Report controls where expression filter is shown. By calling this function and passing the expessionString, items shown in the collection control will be filtered. This filtering happens on the server and the data in the control will be refreshed. This API can be used in the OnBeforeLoad event of the control to load the data based on the passed filter expression. It can also be used in any custom command onClick handler shown in the toolbar, but in that case reloadControl should be called right after this API to reload the data in the control based on the filter expression. For Report control, this API can only be used if the report is created based on the expression.

```
appStudioWizard.filterCollectionItems(collectionControlName, expressionString);
```

Simple Expression: Simple expression will have propertyName, Operator, and value. Type of the operator that can used inside the expression depends on property's data type. One can check supported operators for a property, by accessing the property filter in the properties flyout of that control.

```
"propertyName OPERATOR value"
```

Following operators can be used to create an expression based of the property's data type.

Contains, EqualTo, NotEqualTo, LessThan, GreaterThan, LessThanEqualTo, GreaterThanEqualTo, StartsWith, EndsWith, DoesNotContains, DoesNotStartsWith & DoesNotEndsWith.

Complex Expression: Complex expression is composed of many simple expressions with logical and grouping operators.

"expression LOGICAL_OPERATOR expression LOGICAL_OPERATOR expression"

Logical operators 'AND' and 'OR' can be used along with grouping operators '(' and ')' to a create complex expressions that contain simple expressions.

Sample Expression:

```
 const expressionString = "id NotEqualTo
131706F6B80D47A5A68ED00BAF210CC8"
 const expressionString = "name Contains app OR name EndsWith item"
 const expressionString = "classification EqualTo Item AND name StartsWith
AS_"
```

Control Item

Control item represents an item shown in the collection control at root or child level. Control item provides API to set and get property values. It also provides API to add and remove child items from the parent control item.

getItemId

Once you have controlItem that represents a row in collection controls, associated relationshipId/itemId can be obtained by calling this function.

```
 var itemId = controlItem.getItemId();
```

getRelatedId

Once you have controlItem that represents a row in collection controls, associated relatedId can be obtained by calling this function.

```
var relationshipId = controlItem.getRelatedId();
```

isItemNew

Once you have controlItem that represents a row in collection controls, associated isNew state can be obtained by calling this function. Return true if the item is newly added or inserted.

```
var isItemNew = controlItem.isItemNew()
```

getValue

Once you have controlItem that represents a row in collection controls, a particular cell value in that row can be obtained by passing column property name and cell value. If the controlItem represents relationship data, to get relatedItem property value, you need to pass columnPropertyName as “related.relatedItemPropertyName”.

```
var cellValue = controlItem.getValue(columnPropertyName);
```

setValue

Once you have itemControl that represents a row in collection controls, a particular cell value in that row can be set by passing column property name and cell value. If the controlItem represents relationship data, to set relatedItem property value, you need to pass columnPropertyName as “related.relatedItemPropertyName”.

```
itemControl.setValue(columnPropertyName, cellValue);
```

getChildItems

Once you have controlItem that represents a row in worksheet, treeTable, structure, graph controls, its child items can be retrieved by passed child relationshipName.

```
var childControlItems =
controlItem.getChildItems(childRelationshipName);
```

getChildItem

Once you have controlItem that represents a row in worksheet, treeTable, structure, graph controls, its child item can be retrieved by passing child relationshipName and child relationshipId

```
var childControlItem = controlItem.getChildItem(childRelationshipName, childRelationshipId);
```

getChildItemByRelatedId

Once you have controlItem that represents a row in collection controls, childControlItem can be obtained given that child relationshipName, child relatedId.

```
var childControlItems = controlItem.getChildItemByRelatedId( childRelationshipName, childRelatedItemId);
```

addChildItem

Once you have controlItem that represents a row in collection controls, childItem can be added given that child relationshipName, child relationshipId, and itemData. It also returns newly added item in the form of childControlItem.

```
const itemData = {
properties: {
address: "Sample Data",
},
relatedItem: {
properties: {
name: "Sample Item",
description: "Sample Desc",
},
},
};
```

```
var childControlItem = controlItem.addChildItem(childRelationshipName, childRelationshipId, itemData);
```

insertChildItem

Once you have controlItem that represents a row in collection controls, childControlItem can be inserted given that child relationshipName, child relationshipId, and insertItemId. It also returns newly

added item in the form of childControlItem.

```
controlItem.insertChildItem(childRelationshipName, childRelationshipId, insertItemId).then((insertedChildItemControl) => {
// accessing inserted ItemControl and setting its value
insertedChildItemControl.setValue(columnPropertyName, cellValue);
});
```

deleteChildItem

Once you have controlItem that represents a row in collection controls, childItem can be deleted from the controlItem given childRelationshiName and childRelationshipId.

```
controlItem.deleteChildItem(childRelationshipName, childRelationshipId);
```

Layout Control: Group & Section

Below APIs are supported for Group & Section controls when their Associated Type setting is set to **Type**.

setLayoutControlContext

Set itemId as context for a layout control based on its control name to refresh the control with passed itemId.

```
appStudioWizard.setLayoutControlContext(sectionControlName, itemId);
```

getLayoutControlContext

Get itemId associated with a layout control based on its name.

```
var itemId = appStudioWizard.getLayoutControlContext(sectionControlName);
```

List Control

setListOptions

Set list options for a List control by its control name.

```
const options = { optionValue: "OptionLabel", optionValue2:
```

```
"OptionLabel2" };
 appStudioWizard.setListOptions(listControlName, options);
```

setCollectionListOptions

Collection controls can have column that represents a list. Table, Worksheet, TreeTable, DashboardTable, DashboardWorksheet can have a column of type list**.** Following API sets list options for a collection control by passing its list column name.

```
 const options = {optionValue: "OptionLabel", optionValue2:
"OptionLabel2"};
 appStudioWizard.setCollectionListOptions(collectionControlName,
columnName, options);
```

setCollectionChildListOptions

Set list options for nested table for a collection control by passing its list column name and relationship name.

```
 const options = {optionValue: "OptionLabel", optionValue2:
"OptionLabel2"};
 appStudioWizard.setCollectionChildListOptions(collectionControlName,
relationshipName, columnName, options);
```

Report Control

setReportParameters

This API can be used if the report is created based on QueryDefinition or ServerMethod to filter the report data dynamically. It accepts the JavaScript object that contains properties, those will be passed as parameters to the QueryDefinition or ServerMethod. This API can be used in the OnBeforeLoad event of the control to load the data based on the passed filter parameters. It can also be used in any custom command onClick handler shown in the toolbar, but in that case reloadControl should be called right after this API to reload the data in the report based on the filter parameters.

```
 appStudioWizard.setReportParameters(reportControlName, {classification:
"Component"});
```

Frame Control

getElement

Gets DOM element of the control based on the control name. This is mainly used to get frame DOM element for the Frame control.

```
var htmlFrameElement = appStudioWizard.getElement(frameControlName);
if (htmlFrameElement) htmlFrameElement.src = "siteURL";
```

CADViewer Control:

setControlValue

Sets cadDocumentId on the CADViewer control bv its control name to load the cad model.

```
appStudioWizard.setControlValue(viewerControlName, "cadDocumentId");
```