Collection Controls

The following API functions with collection controls such as Table, Worksheet, TreeTable, Structure, DashboardTable, and DashboardWorksheet. These collection control APIs can only be used when the controls 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

This API selects items in a collection control by specifying the collection control name along with an array of relationship IDs when the table displays relationship data. For dashboard tables, dashboard worksheets, and reference items tables, you can pass item IDs directly instead.

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

getControlItem

Retrieves an item from a collection control by specifying the control name and either the relationshipId or itemId. Use relationshipId when the table represents relationships, and itemId when the table represents dashboard data or references.

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

getControlItems

Retrieves all items from a collection control by specifying the control name. This method returns all items currently bound to the control.

```
var items = appStudioWizard.getControlItems(collectionControlName);
```

getControlItemByRelatedId

Retrieves an item from a collection control using the relatedItemId and the control name. It works only for collection controls that display non-null relationship data.

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

addControlItem

This API adds a new item to a collection control—such as a table, worksheet, or tree table—that displays relationship data. The new item is inserted immediately after the specified relationshipId within the collection. The function requires the collection control name and an ItemData object containing all relevant properties and their values. Upon successful insertion, the newly added item is returned.

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

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

addControlItems

Adds multiple items to a dynamic table control by passing an array of itemData objects, each containing all necessary properties and values. This method only supports dynamic table controls that display relationship data and returns an array of all inserted items.

```
const itemDataArray = [
{
properties: {
name: 'Sample Item 1',
description: 'Sample Desc 1',
},
},
{
properties: {
name: 'Sample Item 2',
description: 'Sample Desc 2',
},
},
]
```

```
var addedControlItem =
appStudioWizard.addedControlItems(collectionControlName, itemDataArray);
```

insertControlItem

This API inserts an item into a collection control—specifically a table, tree table, or worksheet that displays relationship data—immediately after the specified relationshipId. It requires the collection control name and returns the inserted item.

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

deleteControlItem

This API deletes an item from a collection control. For relationship tables, you provide the collection control name and the relationshipId of the item to delete. For tables displaying reference items or items, you pass the itemId instead.

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

setCommandDropdownOptions

Each collection control includes its own toolbar with various commands. If a command in the toolbar is a dropdown, you can set the options for that dropdown by using this function, providing 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 is applicable only to Table, Worksheet, and Report controls that support expression-based filtering. By invoking this function and passing an expressionString, the items displayed in the control will be filtered accordingly. The filtering is executed on the server, and the control's data is refreshed to reflect the updated results.

You can use this API in the OnBeforeLoad event of the control to load data dynamically based on the specified filter expression. Alternatively, it can be used in a custom toolbar command's onClick handler. However, when used this way, you must also call reloadControl immediately afterward to ensure the data in the control is refreshed according to the new filter. For Report controls, this API is only applicable if the report was created using an expression-based query.

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

Simple Expression: A simple expression consists of a property name, an operator, and a value. The type of operator that can be used in the expression depends on the data type of the selected property. To view the list of supported operators for a specific property, you can access the Property Filter section in the Properties flyout of the respective 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_"
```

setInsertFilterExpression

This API allows you to set a **default filter expression** inside the **InsertItem flyout** when it is opened to select an item for insertion. It is supported in **Table**, **Worksheet**, and **TreeTable** controls. You can optionally pass a **boolean parameter enforce**, which, if set to true, will hide the configured filter expression from the user.

// creating a custom expression string

```
let expressionString = "classification EqualTo Item AND id NotEqualTo 8D58C2D8DC1649279E64CCF5BD2FF7A7";
```

*// Refer to **filterCollectionItems** section on how to build your expression string*

```
appStudioWizard.setInsertFilterExpression(controlName, expressionString, enforce, typeName);
```

setControlParameters

This API can be used on Structure, Graph, and Report controls to pass custom parameters to the underlying QueryDefinition or ServerMethod. It accepts a JavaScript object containing key-value pairs that represent the parameters to be passed. These parameters will be sent to the server and used during data retrieval. You can use this API in the OnBeforeLoad event of the control to ensure the data is loaded based on the supplied custom parameters. Alternatively, it can be used in the onClick handler of a custom toolbar command. In that case, you must call the reloadControl API immediately after, to refresh the control's data using the updated parameters.

```
appStudioWizard.setControlParameters(controlName, {classification:"Component"});
```

getStructureActiveTab

Gets the typeName of current active tab of structure control given the structure control name and selected node itemId.

```
appStudioWizard.getStructureActiveTab(controlName, itemId);
```

setStructureActiveTab

Sets the currently active tab in the structure control based on the structure control name, the selected node's itemId, and the tab's typeName. For regular relationships, the typeName is a combination of the relationship name and the related ItemType name, as illustrated in the example below. In the case of null relationships, the typeName consists of only the relationship name. When the tab displays an item from an ItemProperty or a reference item, the typeName should be the ItemType name alone.

```
const typeName = “relationshipName(relatedItemTypeName)”;
appStudioWizard.setStructureActiveTab(controlName, itemId, typeName);
```