Contents

DOWNLOAD: Example of a project

Lists (inertial lists) contain the preset number (whole, positive)of items.

List Types

- **List** uses one template for all its items, all items are of one type and you can see several items at once.
- **Static List** - list items - popups; all are different and you can see only one item which is aligned by the list center at a time.

Creating Lists

The process of list creation can be divided into three stages:

- Creating a **list** type item
- Creating and assigning a list template - a **popup** created previously
- Fillin the list

To create a **list** item use the command:

IR.CreateItem(IR.ITEM_LISTBOX, List_Name, Left, Top, Width, Height)

- **IR.ITEM_LISTBOX** - tell the command to create the **list** item
- **List_Name** - the name of the created list
- **Left** - the indent from the left page border
- **Top** - the indent from the top page border
- **Width** - item width
- **Height** - item height

Example of creating a **list** item at the application launch:

```
IR.AddListener(IR.EVENT_START,0,function()  // Event is activated at the
```

```
application launch
{
	IR.CreateItem(IR.ITEM_LISTBOX,"Item 1",277,10,441,751);    // Create a
"list" type item, it will appear on the page displayed at the launch
});
```

List Template

List template is a popup which sets the number, sequence and positioning of items on the list.

The template can be created in GUI Editor by visual positioning of items or in a script. You can see more information about how to create popups in scripts here.

To assign the template to a list use the command:

List_Name.Template = "Popup_Name"

- **List_Name** - the name of the variable which contains the list identifier
- **Template** - the list property responsible for the template
- **Popup_Name** - the name of the popup to be used as the template

Example of assigning of the template to a **list** type item at the application launch:

```
IR.AddListener(IR.EVENT_START,0,function()  // Event is activated at the
application launch
{
	IR.CreateItem(IR.ITEM_LISTBOX,"Item 1",277,10,441,751);    // Create a
"list" type item, it will appear on the page displayed at the launch

	list = IR.GetItem("Page 1").GetItem("Item 1");     // Receive the list
identifier and save it into the variable with the name "list".

	list.Template = "Popup 1";   // Assign the template to the created list
});
```

List Direction

The orientation of list items define the list direction:

- **Vertical** - list items are scrolled one by one from the top downward
- **Horizontal** - list items are scrolled one by one from the left to the right

Filling Lists

After the list is created and the popup template is selected you can start filling the list.

To fill lists use the command:

List_Name.CreateItem(element, sub_element, {Property_1: Value, .. ,Property_N: Value});

- **List_Name** - the name of the variable containing the list identifier
- **element** - the number of a list item beginning with 1
- **sub_element** - the number of sub element of the list item beginning with 0. Zero is an initial popup.
- **Property_1: Value, Property_N: Value** - any available properties of sub element for example .Width. The property can be written N-times separated by commas, N - the number of available properties for sub element.
- **Value** - the value the property is going to take, for example for the .Width property write 1024.

Example of filling a list at the application launch:

```
IR.AddListener(IR.EVENT_START,0,function()  // Event is activated at the
application launch
{
     list = IR.GetItem("Page 1").GetItem("Item 1");// Receive the list
identifier and save it in the variable with the name "list".

     list.CreateItem(1, 1, {Text: "TV Bedroom"});  // Create item 1 and set
the "Text" property  "TV Bedroom" for sub element 1
     list.CreateItem(2, 1, {Text: "Projector"});   // Create item 2 and set
the "Text"  property "Projector" for sub element 1
     list.CreateItem(2, 2, {Image: 1});            // Set the "Image"
property which identifier is 1 for sub element 2
     list.CreateItem(3, 1, {Text: "TV Hall"});     // Create  item 3 and set
the "Text" property  "TV Hall" for sub element 1
     list.CreateItem(3, 2, {Image: 2});            // Set the "Image"
property which identifier is 2 for sub element 3
});
```

Changing List Properties

Changing of the list is performed by creating the list items with new value of the changed property from scratch.

Example of changing a list at button pressing:

```
IR.AddListener(IR.EVENT_START,0,function()  // Event is activated at the
application launch
{
     list = IR.GetItem("Page 1").GetItem("Item 1");     // Receive the list
```

```
identifier and save it in the variable with the name "list".
     button = IR.GetItem("Page 1").GetItem("Item 2");   // Receive the button
identifier and save it in the variable with the name "button".

     list.CreateItem(1, 1, {Text: "TV Bedroom"}); // Create item 1 and set
the "Text" property  "TV Bedroom" for sub element 1

     list.CreateItem(2, 1, {Text: "Projector"});  // Create item 2 and set
the "Text"  property "Projector" for sub element 1
     list.CreateItem(2, 2, {Image: 1});           // Set the "Image" property
which identifier is 1 for sub element 2

     list.CreateItem(3, 1, {Text: "TV Hall"});     // Create item 3 and set
the "Text" property  "TV Hall" for sub element 1
     list.CreateItem(3, 2, {Image: 2});           // Set the "Image" property
which identifier is 2 for sub element 3


     // As an example of changes in the list let's change positions of item 2
and item 3 at button pressing
     IR.AddListener(IR.EVENT_ITEM_PRESS, button, function()  // Event is
activated at button pressing
     {
      list.CreateItem(2, 1, {Text: "TV Hall"});  // Change values of item 2
properties to the values of item 3 properties
      list.CreateItem(2, 2, {Image: 2});

      list.CreateItem(3, 1, {Text: "Projector"});  // Change values of item 3
properties to the values of item 3 properties
      list.CreateItem(3, 2, {Image: 1});
     });
});
```

Clearing Lists

List clearing removes all list items. It is used when you don't need the content of the list any more. To clear lists use **List_Name.Clear()**, where

- **List_Name** - the name of the variable containing the list identifier.

Example of clearing a list at button pressing:

```
IR.AddListener(IR.EVENT_START,0,function()  // Event is activated at the
application launch
{
     list = IR.GetItem("Page 1").GetItem("Item 1");     // Receive the list
identifier and save it in the variable with the name "list".
     button = IR.GetItem("Page 1").GetItem("Item 2");   // Receive the button
identifier and save it in the variable with the name "button".
```

```
     list.CreateItem(1, 1, {Text: "TV Bedroom"}); // Create item 1 and set
the "Text" property  "TV Bedroom" for sub element 1

     list.CreateItem(2, 1, {Text: "Projector"});  // Create item 2 and set
the "Text"  property "Projector" for sub element 1


     list.CreateItem(3, 1, {Text: "TV Hall"});     // Create item 3 and set
the "Text" property  "TV Hall" for sub element 1

     // As an example of clearing the list let's clear the list at button
pressing:
     IR.AddListener(IR.EVENT_ITEM_PRESS, button, function()  // Event is
activated at button pressing
     {
        list.Clear(); // Clearing the list
     });
});
```

Deleting Lists

If you're required to delete one or several items from the list use:

List_Name.DeleteItem(ItemNumber)

- **List_Name** - the name of the variable containing the list identifier.
- **ItemNumber** - the number of the list item (beginning with 1).

Example of deleting list items at pressing:

```
IR.AddListener(IR.EVENT_START,0,function()  // Event is activated at the
application launch
{
     list = IR.GetItem("Page 1").GetItem("Item 1");     // Receive the list
identifier and save it in the variable with the name "list".
     button = IR.GetItem("Page 1").GetItem("Item 2");   // Receive the button
identifier and save it in the variable with the name "button".

     list.CreateItem(1, 1, {Text: "TV Bedroom"}); // Create item 1 and set
the "Text" property  "TV Bedroom" for sub item 1

     list.CreateItem(2, 1, {Text: "Projector"});  // Create item 2 and set
the "Text"  property "Projector" for sub item 1

     list.CreateItem(3, 1, {Text: "TV Hall"});     // Create item 3 and set
the "Text" property  "TV Hall" for sub item 1

     // Delete list items at presing on the button
     IR.AddListener(IR.EVENT_ITEM_PRESS, button, function()  // Event is
```

```
activated at button pressing
     {
        list.DeleteItem(3); // Delete list item 3
        list.DeleteItem(2); // Delete list item 2
     });
});
```

DOWNLOAD: Example of a project