TargetGroup

Available since firmware version 7.5

Table of Contents

Extends: Group

Description

The **TargetGroup** node class associates a set of rectangular regions that children of the group will occupy. Like MarkupList, the TargetGroup has a content field containing the data for each item and an itemComponentName field that specifies an RSG component that will be used to render a content item. It also has a targetSet field that contains a TargetSet that define a set of rectangular targets where children of the TargetGroup will be rendered.

The TargetGroup node is typically used to create a scrolling list (or row) of items where the focused item occupies more space than the other items.

For example, a TargetGroup could be used to create a full screen vertical scrolling list of item where the focused item is larger than the other items in the list. As the list items scroll, the appearance of the item moving into the focus region would be dynamically adjusted to fill the larger focus region. Simultaneously, the appearance of the item leaving the focus region would be dynamically adjust to return to the unfocused size. To set up this use case, you might set the targetSet field to a TargetSet node that specifies nine rectangles. The first rectangle would be specified to have the width and height of an unfocused item and be positioned so that it's bottom is above the top of the screen. The last rectangle would be specified to have the width and height of an unfocused item and be positioned so that's top is below the bottom of the screen. The remaining seven rectangles would define the rectangular regions of the onscreen items. Suppose the design calls for the focus item to be centered vertically at the center of the screen. To do that, you would specify the 5th rectangle to be larger than the other eight and position it so that it is centered vertically, you would specify the remaining rectangles to form a column of rectangular regions where the top three and bottom three visible items would be located.

The second step of setting up this use case would be to implement an RSG component that will be used to render each item. The TargetGroup node manages the creation of the items for the visible components, associates each with a ContentNode, and updates fields of the item component with information such as the current width and height of the item and the focus status of the item.

The TargetGroup's jumpToItem field is set to identify which content item is to be located at the TargetSet's targetRects field target rectangle identified by the TargetSet's focusindex field.

The final step of setting up this use case would be to create a VerticalList component that extends TargetGroup, sets up the TargetGroup's TargetSet node, and as the user presses up and down buttons on the remote, sets the TargetGroup's animateToTargetItem field to the prior or next index. Setting the animateToTargetItem field causes the displayed items to smoothly animate from their current target region to another target region, such that the specified index ends up at the TargetSet's target rectangle that is identified by the TargetSet's focusIndex field.

The above use case specifies the most common use case for the TargetGroup node, but only hints at the possible uses. For example, you could create your own RSG components with various custom behaviors. There might include:

- A list where all the items are small when the list does not have the focus, but when the list receives the focus, all of the items smoothly adjust their size and position so that the focus item is largest, the items on either side of the focus item are slightly larger than the unfocused size and the remaining items remain the same size as the unfocused items. To do this, you would create two TargetSet's in your RSG component, one that defines the regions when the list is unfocused and one that defines the regions when the list is focused. Initially, the TargetGroup's targetSet field would be set to the unfocused TargetSet node. Then, when the list is focused, the targetSet's animateToTargetSet field would be set to the focused TargetSet node, causing all of the target regions to smoothly animate to their new size and position, taking along the associated item component's with them.
- A horizontal scrolling list of items where the focused region floats across the screen as the user presses left/right until the focus region reaches the edge of the display, at which point the focus region remains stationary and the items scroll left or right. This would require the use of several TargetSet nodes (one for each possible position of the focus region). Initally, the TargetGroup's targetSet field would be set

to one of these TargetSet's. Then while the focus region is not at one of the edges, key presses would set the animateToTargetSet field to animate the focus region to its next location. Once the focus region reaches an edge, another key press in the same direction would set the animateToTargetIndex field to cause the items to scroll so that the next content item occupies the focus region.

- A list where when an item is selected, all of the items fly off the screen while the selected item zooms up and moves to the center of the screen. To set up this use case, you would specify a TargetSet for when the list items are onscreen and another TargetSet for the onscreen location of the focused item and the offscreen locations where each items will disappear.
- A circular arrangement of a fixed number of items with the item at the 6 o'clock position being larger and having the focus. Note that in this case, no offscreen targets would be specified

Fields

Field	Type	Default	Use
itemComponentName	string	""	Specifies the name of a XML component for the group items. An instance of this component is created on demand for each visible item of the group. The XML component must define a specific interface as detailed in **TargetGroup XML Component** below.
content	**ContentNode**	none	Specifies the content for the group. See **Data Bindings** below for more details.
targetSet	**TargetSet**	invalid	Specifies the TargetSet to use to define the target regions of the items in the group. When set or modified, the target regions are immediately adjusted to use the new values.
defaultTargetSetFocusIndex	int	0	For TargetSet's that do not specify a focusIndex, this value will be used as the index of the TargetSet where the focused item is located. If a TargetSet specifies any value for the focusIndex, that value will be used instead of defaultTargetSetFocusIndex.
wrap	Boolean	false	Specifies whether the content items wraparound at the end of the TargetGroup to fill all of the targets rectangles.
duration	Time	0.3 seconds	Specifies the time, in seconds, to perform the animation when the animateToItem or animateToTargetSet fields are set.
showTargetRects	Boolean	false	Specifies whether the current target rectangles (as defined in the read-only currTargetSet field's TargetSet) are drawn or not. Typically this would only be set to true while debugging a channel, although in some use cases its possible that you might want to display the current target rectangles. The rectangles are drawn using the color in the targetSet's TargetSet node's color field.
currFocusItemIndex	float	-1.0	**Read-Only** As the TargetGroup animation occurs, this field is constantly updated to represent the index of the ContentNode currently occupying the focus target region. When currFocusItemIndex is an integer value, the specified ContentNode occupies the focus target. When currFocusItemIndex has a fractional part, the value indicates that an animation is in process. For example, a value of 5.7 would indicate that items 5 and 6 are currently overlapping the focus region, with item 6 occupying 70% and item 5 the other 30%.
currTargetSet	**TargetSet**	invalid	**Read-Only** As the TargetGroup animation occurs that is initiated by setting the animateToTargetSet field, currTargetSet contains the current values of the target regions as the animation proceeds from the initial TargetSet's targets to the new TargetSet's targets.
itemSelected	integer	0	**Read-Only** When a group item is selected, set to the index of the selected item.
itemFocused	integer	0	**Read-Only** When a group item gains the key focus, set to the index of the focused item.
itemUnfocused	integer	0	**Read-Only** When a group item loses the key focus, set to the index of the unfocused item.
jumpToItem	integer	0	**Write-Only** When set to a valid item index, causes the group to immediately update so that the specified index moves to the target region specified by the TargetSet's focusIndex.

animateToItem	integer	0	**Write-Only** When set to a valid item index, causes the group to quickly scroll so that the specified index moves into the to the target region specified by the TargetSet's focusIndex.
animateToTargetSet	**TargetSet**	invalid	**Write-Only** When set to a valid TargetSet, causes the group to quickly animate so that the target regions of the initial TargetSet node are smoothly interpolated to the corresponding target regions of the new TargetSet node. If the two TargetSet's focusIndex fields are different, the focusIndex is also animated from the old to the new value. Note that if the number of rectangles in the new TargetSet's targetRects field does not match the number of rectangles in the current TargetSet's targetRects field, the results are undefined.

TargetGroup XML Component

The **TargetGroup** node `itemComponentName` field value should be set to the name of an XML component used to display each item in the grid. An instance of this component is created for each target region specified in the TargetSet's targetRects field.

If the XML component contains interface fields that match the names shown in the table below, those fields will be updated by the **TargetGroup** no de. This allows the XML component to alter the item appearance based on changes to these interface fields.

Note that the fields are updated in the order presented in the table below. Any layout scripting you write based on these fields should be done in that order to avoid updating your layout based on a field that has not been updated yet.

Field Name	Field Type	Description
currTarget	float	**Read-Only** Set to index of the current TargetSet's targetRect that should contain the item. If currTarget is an integer value, the item's currRect field will be the value of currTarget'th item in the TargetGroup node's currTargetSet field's targetRect's array. If currTarget is not an integer, it indicates that the item is animating from one targetRect index to another. For example, if the value is 5.7, the item is between the rectangles at index 5 and 6 of the TargetGroup node's currTargetSet's targetRect field. The item if 70% occupying the rectangle at index 6 and 30% occupying the rectangle at index 5.
currRect	rectangle	**Read-Only** Set to the rectangle that the item should occupy. The rectangle values can be accessed either as an associative array with "x", "y", "width" and "height" elements or as an array of four float's containing the x, y, width and height values of the rectangles. Note that the item will be automatically translated so that its origin is at the (x,y) location of this rectangle relative to the origin of the TargetGroup node. Typically, the width and height of currRect is used to dynamically adjust the size of the item as it animates from one target to another.
index	integer	**Read-Only** Set to the index of this item in the data model.
groupHasFocus	Boolean	**Read-Only** Set to true if the **TargetGroup** node has focus, false otherwise.
itemContent	**ContentNode**	Contains the data to be displayed by the group item. The relationship between data in the **ContentNode** node and the visual elements of the grid item is determined by the markup and scripts in the item XML component. Typically, an observer callback function of the `itemContent` field is used to update the grid item when the content changes.

focusPercent	float	**Read-Only** The fractional value, from 0.0 to 1.0, of a time delay after focus has moved from one item to the next. The fractional value increases incrementally from 0.0 to 1.0 for the newly-focused item, while simultaneously decreasing from 1.0 to 0.0 for the previously-focused item. This value can be used as a timing key to smoothly animate the appearance of the focused item as well as the previously-focused item, to indicate the movement of focus to the user.
itemHasFocus	Boolean	*Available since firmware version 8.* **Read-Only** Indicates whether the item component currently is the TargetGroup's focused item. When scrolling starts, the itemHasFocus field for the currently focused item is set to false. When scrolling ends, the itemHasFocus field for the newly focused item is set to true. During the scrolling animation, all itemHasFocus fields are set to false. Only one item component of any TargetGroup should have itemHasFocus set to true. If the TargetGroup does not focus, all itemHasFocus fields of their item components should be set to false.

Data Bindings

A **TargetGroup** node should have a single **ContentNode** node as the root node in its content field. One child **ContentNode** node should be added to the root node for each item in the group (these child nodes can be thought of as *item nodes*). Item nodes should contain the data required by the **TargetGroup** node's XML component.

The specific data fields in the **ContentNode** should match the values referenced by the **TargetGroup** node's XML component.

Sample Channels

Sample	Description
FloatingFocusWrap.zip	TargetGroup example demonstrating floating focus and wrapping from the first and last items.
TwoRowFixedFocus.zip	A sample demonstrating an unfocused TargetList when more than one TargetList is present.