ifSGNodeChildren

Table of Contents

Available since firmware version 7.0

The **ifSGNodeChildren** interface allows querying and manipulation of nodes in a SceneGraph node tree, such as creating new nodes, placing them at certain positions in the tree, and removing them. Note that if you want to use the methods on this interface to manipulate child nodes at the scene level, the nodes in question need to be wrapped in another element, typically a **Group** node. For example:

```
<?xml version="1.0" encoding="utf-8" ?>

<component name = "myScene" extends = "Scene" >

  <script type = "text/brightscript" >
    <![CDATA[

    sub init()
      m.myGroup = m.top.FindNode("myGroup")
      m.label = m.myGroup.getChild(0)
    end sub

    ]]>
  </script>

  <children>

    <Group id = "myGroup">
      <Label id = "myLabel" ... />
    </Group>

  </children>

</component>
```

In the example above, `m.label` will contain the `roSGNode` corresponding to the **Label** node after the `getChild()` call. On the other hand, the following will not work:

```
<?xml version="1.0" encoding="utf-8" ?>

<component name = "myScene" extends = "Scene" >

  <script type = "text/brightscript" >
    <![CDATA[

    sub init()
      m.label = m.top.getChild(0)
    end sub

    ]]>
  </script>

  <children>

    <Label id = "myLabel" ... />

  </children>

</component>
```

The **Scene** node children are hidden elements used by the SceneGraph framework. Thus, despite the fact that the **Label** node is in the scene **<children>** element, it will not be retrieved by `getChild()`.

> Always remember that removing or replacing a node in a SceneGraph node tree can cause that node to be destroyed entirely if there are no more references to it.

Implemented By

- roSGNode

Supported Methods

- appendChild(child as roSGNode)
- createChild(nodeType as String) as Object
- insertChild(child as roSGNode, index as Integer)
- removeChild(child as roSGNode)
- removeChildIndex(index as Integer)
- replaceChild(newChild as roSGNode, index as Integer)
- getChild(index as Integer)
- getParent() as roSGNode
- getChildCount() as Integer
- reparent(newParent as roSGNode, adjustTransform as Boolean) as Boolean
- appendChildren(child_nodes as Object) as Boolean

- insertChildren(child_nodes as Object, index as Integer) as Boolean
- removeChildren(child_nodes as Object) as Boolean
- removeChildrenIndex(num_children as Integer, index as Integer) as Boolean
- replaceChildren(child_nodes as Object, index as Integer) as Boolean
- getChildren(num_children as Integer, index as Integer) as Object
- createChildren(num_children as Integer, subtype as String) as Object
- getScene() as roSGNode
- update(aa as roAssociativeArray) as Void
- getAll() as Object
- getRoots() as Object
- getRootsMeta() as Object
- getAllMeta() as Object

Description of Methods

appendChild(child as roSGNode)

Adds a child node to the end of the subject node list of children so that it is traversed last (of those children) during render.

createChild(nodeType as String) as Object

Creates a child node of type *nodeType*, and adds the new node to the end of the subject node list of children.

insertChild(child as roSGNode, index as Integer)

Inserts a previously-created child node at the position *index* in the subject node list of children, so that this is the position that the new child node is traversed during render.

removeChild(child as roSGNode)

Finds a child node in the subject node list of children, and if found, remove it from the list of children. The match is made on the basis of actual object identity, that is, the value of the pointer to the child node.

removeChildIndex(index as Integer)

If the subject node has a child node in the *index* position, remove that child node from the subject node list of children.

replaceChild(newChild as roSGNode, index as Integer)

If the subject node has a child node in the *index* position, replace that child node with the *newChild* node in the subject node list of children, otherwise do nothing.

getChild(index as Integer)

If the subject node has a child node at the *index* position, return it, otherwise return invalid.

getParent() as roSGNode

If the subject node has been added to a parent node list of children, return the parent node, otherwise return invalid.

getChildCount() as Integer

Return the current number of children in the subject node list of children. This is always a non-negative number.

reparent(newParent as roSGNode, adjustTransform as Boolean) as Boolean

Parents the subject node to another node specified by *`newParent`*. If *`adjustTransform`* is true, the subject node transformation factor fields (translation/rotation/scale) are adjusted so that the node has the same transformation factors relative to the screen as it previously did. If *`adjustTransform`* is false, the subject node is simply parented to the new node without adjusting its transformation factor fields, in which case, the reparenting operation could cause the node to jump to a new position on the screen. Returns true if the subject node was successfully reparented, false otherwise.

appendChildren(child_nodes as Object) as Boolean

Available since firmware version 7.2

`child_nodes`	Array of child nodes

Appends the child nodes specified by *child_nodes* to the subject node. Returns true if the child nodes were successfully appended.

insertChildren(child_nodes as Object, index as Integer) as Boolean

Available since firmware version 7.2

`child_nodes`	Array of child nodes
index	Child tree position index

Inserts the child nodes specified by *child_nodes* to the subject node starting at the position specified by *index*. Returns true if the child nodes were successfully inserted.

removeChildren(child_nodes as Object) as Boolean

Available since firmware version 7.2

`child_nodes`	Array of child nodes

Removes the child nodes specified by *child_nodes* from the subject node. Returns true if the child nodes were successfully removed.

removeChildrenIndex(num_children as Integer, index as Integer) as Boolean

Available since firmware version 7.2

`num_children`	Number of child nodes
index	Child tree position index

Removes the number of child nodes specified by *num_children* from the subject node starting at the position specified by *index*. Returns true if the

child nodes were successfully removed.

replaceChildren(child_nodes as Object, index as Integer) as Boolean

Available since firmware version 7.2

child_nodes	Array of child nodes
index	Child tree position index

Replaces the child nodes in the subject node, starting at the position specified by *index*, with new child nodes specified by *child_nodes*. Returns true if the child nodes were successfully replaced.

> Starting from firmware version 8.1, when using replaceChildren() to update the content of each item in a markupGrid, if the developer supplies more items than there are in the original list (going from 4 items to 5), the 'extra' items are ignored and not added as children.

getChildren(num_children as Integer, index as Integer) as Object

Available since firmware version 7.2

num_children	Number of child nodes
index	Child tree position index

Retrieves the number of child nodes specified by *num_children* from the subject node, starting at the position specified by *index*. Returns an array of the child nodes retrieved. If num_children is -1, return all the children.

createChildren(num_children as Integer, subtype as String) as Object

Available since firmware version 7.2

num_children	Number of child nodes
subtype	Node type or extended type

Creates the number of children specified by *num_children* for the subject node, of the type or extended type specified by *subtype*. Returns an array of the child nodes created.

getScene() as roSGNode

Available since firmware version 7.6

Returns the node's root Scene. This returns a valid Scene even if the node is not parented.

update(aa as roAssociativeArray) as Void

Available since firmware version 7.6

This function takes the key-value pairs in the roAssociativeArray and maps the values to the respective field name in the calling node. This requires the field name to be present in the node prior to calling this function. If the field name is not present in the calling node, the value from the roAssociativeArray will not be mapped.

```
aa = {"a":"1", "b":"2", "c":"3", "d":"4"}

cn = createObject("roSGNode", "contentNode")
cn.addfield("a","string",false)
cn.addfield("b","string",false)
cn.addfield("c","string",false)

cn.update(aa)

'At this point, cn would contain the following:
<Component: roSGNode> =
{
    change: <Component: roAssociativeArray>
    focusable: false
    focusedChild: <Component: roInvalid>
    id: ""
    a: "1"
    b: "2"
    c: "3"
}
```

The following methods can be called on any subject node and return the same global results. They can be used in a development channel for debugging purposes, but should not be used in a production channel.

> These methods are similar to the debugger `sgnodes` commands. See Special SceneGraph Debugging Commands for information on the debugger `sgnodes` commands. Also please note that calling these functions from code should only be done for debugging purposes. Any calls to getAll(), getRoots(), getRootsMeta() and getAllMeta() should be removed from your production channels.

getAll() as Object

Available since firmware version 7.2

Returns an array with every existing node created by the currently running channel.

getRoots() as Object

Available since firmware version 7.2

Returns an array with every existing node without a parent created by the currently running channel. The existence of these unparented nodes means they are being kept alive by direct BrightScript references. These could be in variables local to a function, arrays, or associative arrays, including a component global `m` or an associative array field of a node.

getRootsMeta() as Object

Available since firmware version 7.2

Returns an array with every existing node without a parent created by the currently running channel. The existence of these unparented nodes means they are being kept alive by direct BrightScript references. These could be in variables local to a function, arrays, or associative arrays, including a component global `m` or an associative array field of a node. These unparented nodes are organized as an XML forest of trees.

getAllMeta() as Object

Available since firmware version 7.2

Returns a string with a dump of the same nodes as the `getAll()` method, organized as an XML forest of trees according to the usual parent-child node relationship. Cycles are handled with a reference entry in the tree rather than indefinite recursion.