Tree

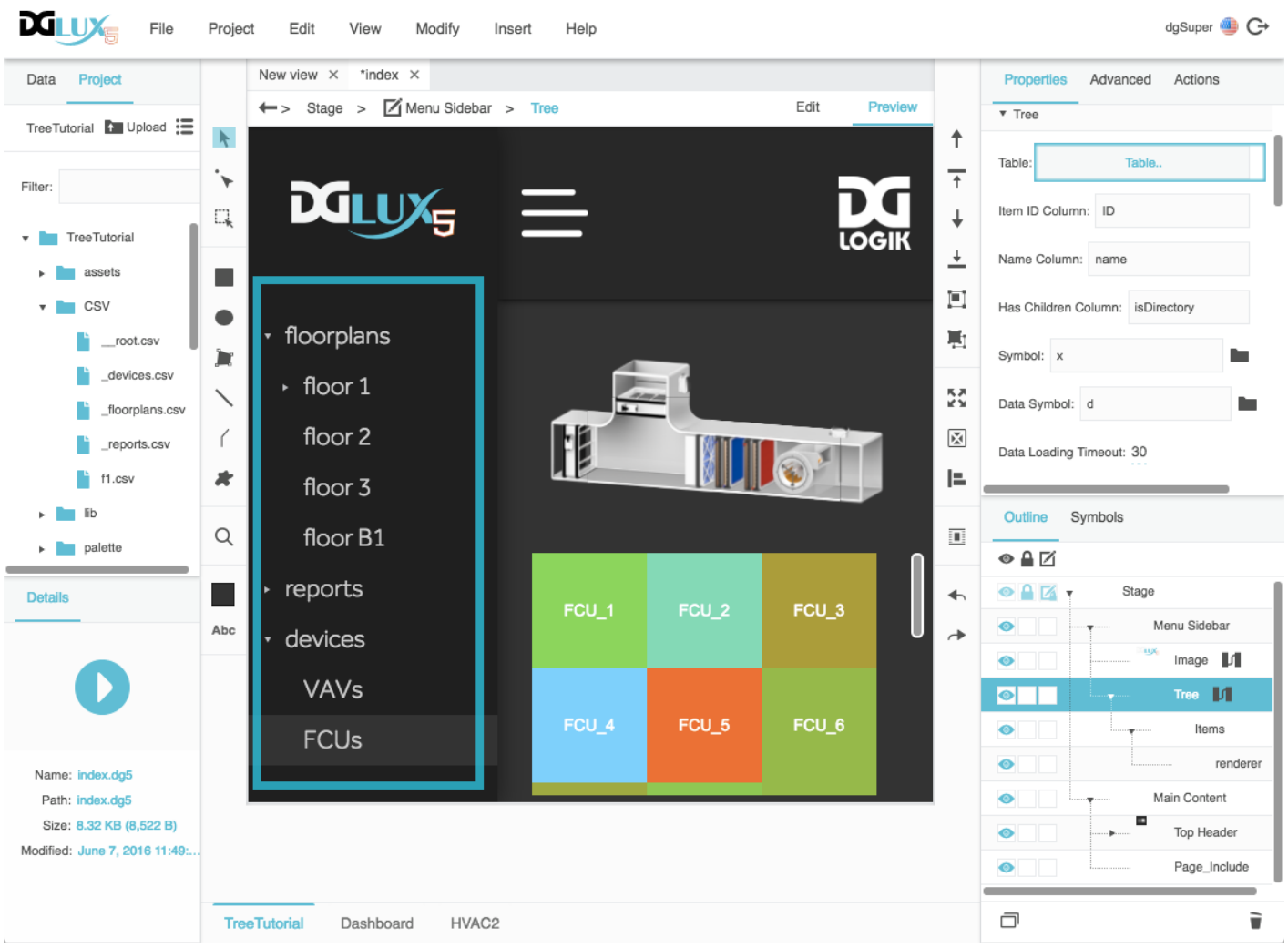
The tree component loads source tables and uses these tables to render a visual tree. Each source table describes the children of some tree node.

The tree uses "lazy loading," which means that a child table is loaded only when the relevant tree node is expanded. In order to enable this lazy loading, you must build a special dataflow symbol. See About the Dataflow Symbol.

Note

The terms *tree node* and *tree item* are closely related. A node is a tree data unit. An item is the rendered visual entity for a node.

For a detailed reference of properties that affect trees, see Common Properties and Tree Properties.



The tree component in DGLux5

About the Source Tables

The top-level source table is loaded in dataflow and can describe the top-level tree nodes. For example, to create the tree in the image above, the top-level source table could contain one row each for `floorplans`, `reports`, and `devices`.

Note

Alternatively, the top-level source table can describe a root node that you do not want to appear in your tree. In such a case, set the **Show Root** property to FALSE.

You bind the top-level source table to the **Table** property of the tree. All other source tables are loaded lazily.

All source tables must use the same column names for columns that are used to load tables, to identify tree items for selection, or to render tree items. These columns can include:

- **Name**—By default, this name becomes the string displayed in the node. This behavior can be overridden by a visual symbol.
- **ID**—This value is required and must be unique within a table. If a tree uses a property that requires unique IDs across tables, such as the **Select Node** property, this value must be unique across all source tables.
- **Whether the item has children**—This boolean value is optional. If it is not defined, the user can attempt to expand any node, even if no children exist.
- **Dataflow symbol properties**—This can be any column that is used to define lazy loading behavior, such as a path. See About the Dataflow Symbol.
- **Visual symbol properties**—This can be any column that is passed to the items renderer for the visual symbol. See About the Visual Symbol.

About the Dataflow Symbol

The tree requires a special dataflow symbol for lazy loading. You must create this dataflow symbol and use the symbol name as the **Data Symbol** property of the tree.

The dataflow symbol must have two tabledata parameters. These parameters must be named `input` and `output`. If any other names are used, the tree might not work at all. These names are case sensitive. The `input` and `output` parameters represent any particular tree node and the children of that node, respectively. You must define the relationship between `input` and `output` within the symbol.

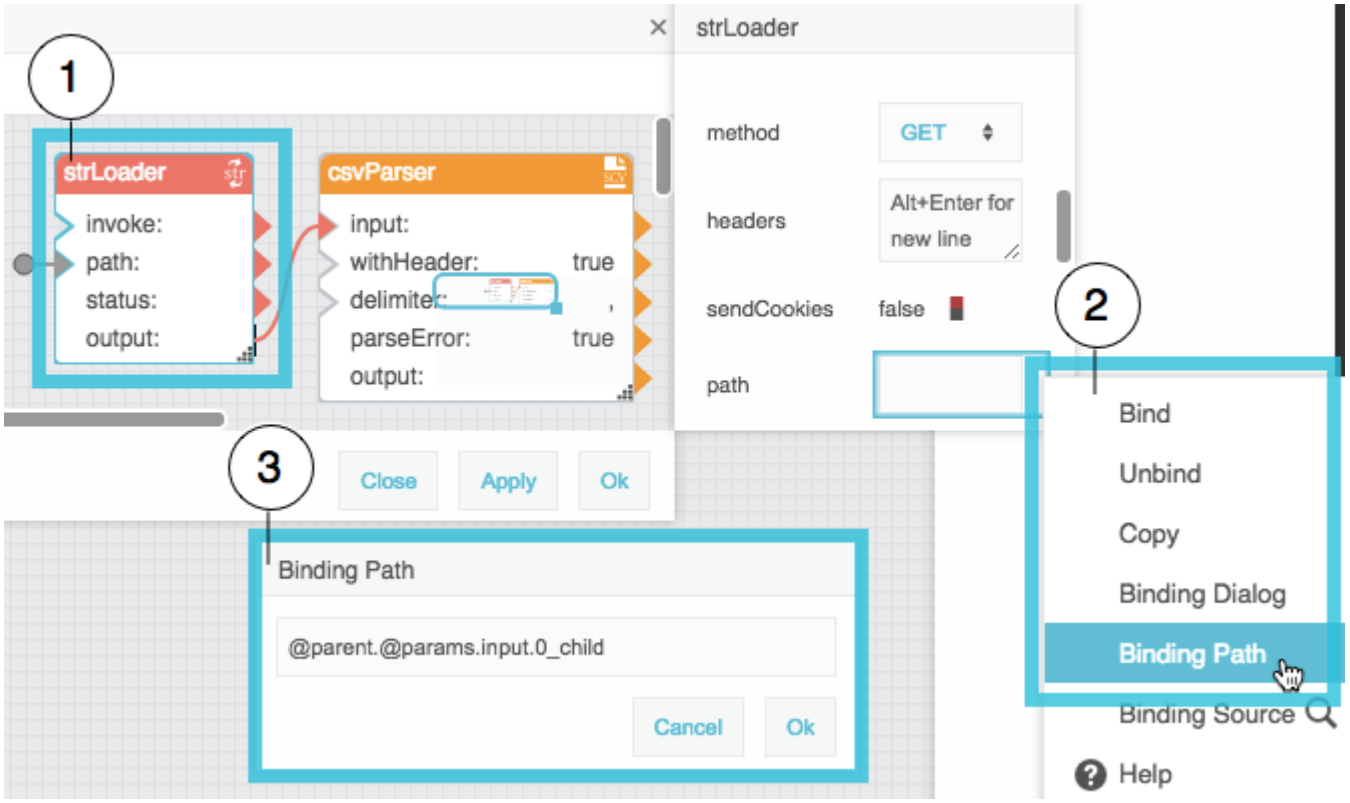
The following are some example methods for creating the dataflow symbol. For more examples, see the tree tutorials.

- **Example 1: Tables are parsed from CSV files on the server.**

  Any table that has children must include a column that contains the paths to the child CSV files.

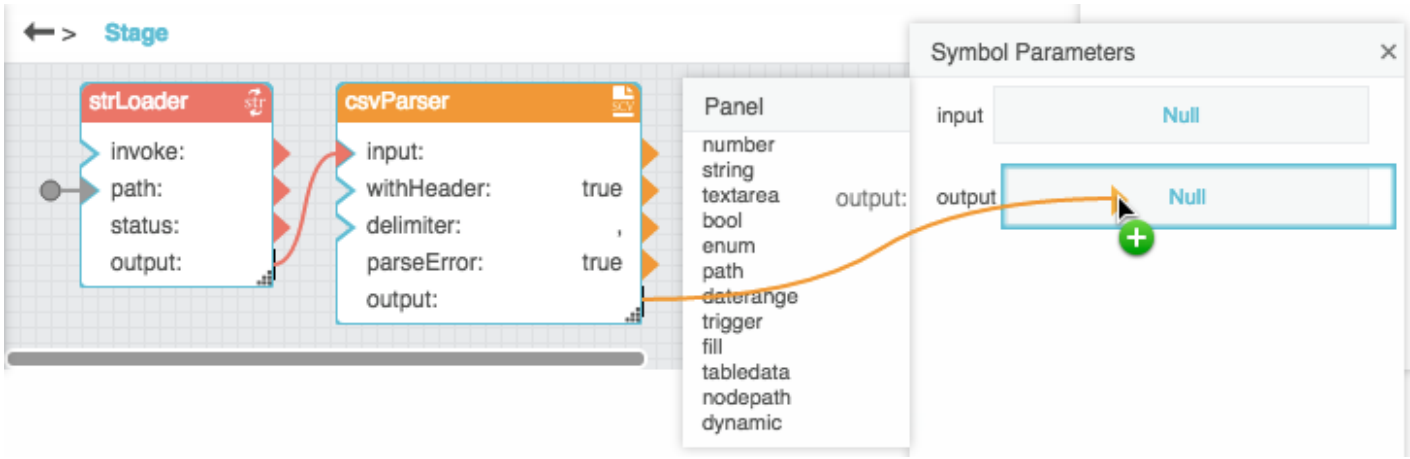
  1. Inside the dataflow symbol, add a String Loader block.
  2. Hover over the String Loader's **path** property, click the blue dot, and choose **Binding Path**.
  3. Set the binding path as `@parent.@params.input.0_child`, where `child` is the name of the table column that contains CSV file paths.
  4. Join this String Loader block to a CSV Parser.
  5. Bind the CSV Parser's output table to the symbol's **output** parameter.
- **Example 2: Tables reflect data structure or project directory structure.**
  1. Use a List Node or Get Children block for data nodes and metrics, or use List Files for DGLux5 project files.
  2. Hover over the **path** property, click the blue dot, and select **Binding Path**.
  3. Set the binding path as `@parent.@params.input.0_<pathColumn>`.
  4. Bind the block output table to the symbol's **output** parameter.
- **Example 3: Queries.**
  1. Inside the dataflow symbol, add a Concatenate block that creates a query string.
  2. For the **input *n*** property that represents the path, set the binding path as `@parent.@params.input.0_<pathColumn>`.
  3. Join the Concatenate block to the block that runs the query. This query block might be Invoke Action or another block, depending on your data source.
  4. Bind the query block's output to the symbol's **output** parameter.

The following image demonstrates how to set the binding path in the dataflow symbol.



1	Select block.	2	Click the blue dot and choose Binding Path.
3	Type binding path.		

The following image demonstrates how to bind the dataflow symbol output to the `output` parameter.



Important

Whenever the column names change, for example if your query changes, you must re-create the dataflow symbol.

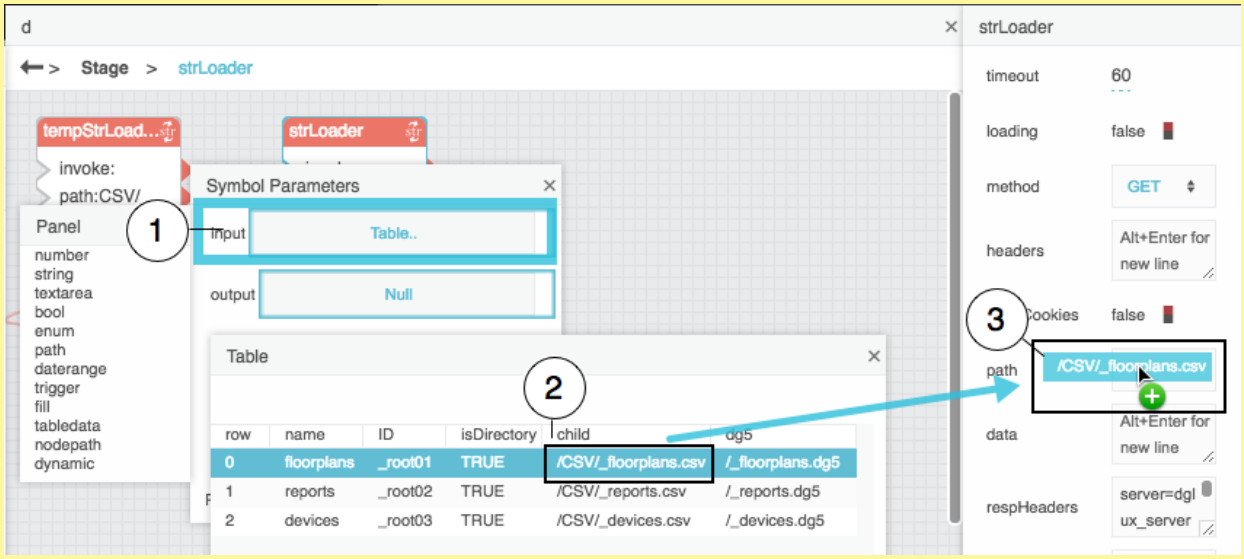
Tip

To avoid typing a binding path manually, you can use the following shortcut:

1. Temporarily bind a table with the same column names as the source tables to the `input` parameter.
2. Drag the table cell from row 0 and the relevant column.
3. Drop the cell on the relevant property.

   This causes the table cell at this location to be set as the binding path.
4. Delete the binding to the `input` parameter.

The following image demonstrates this shortcut.



1	Open input table.	2	Drag the cell from relevant column, top row.
3	Drop on relevant dataflow property.	4	(Not shown) Delete the binding to `input`.

See also: Page Model.

About the Visual Symbol

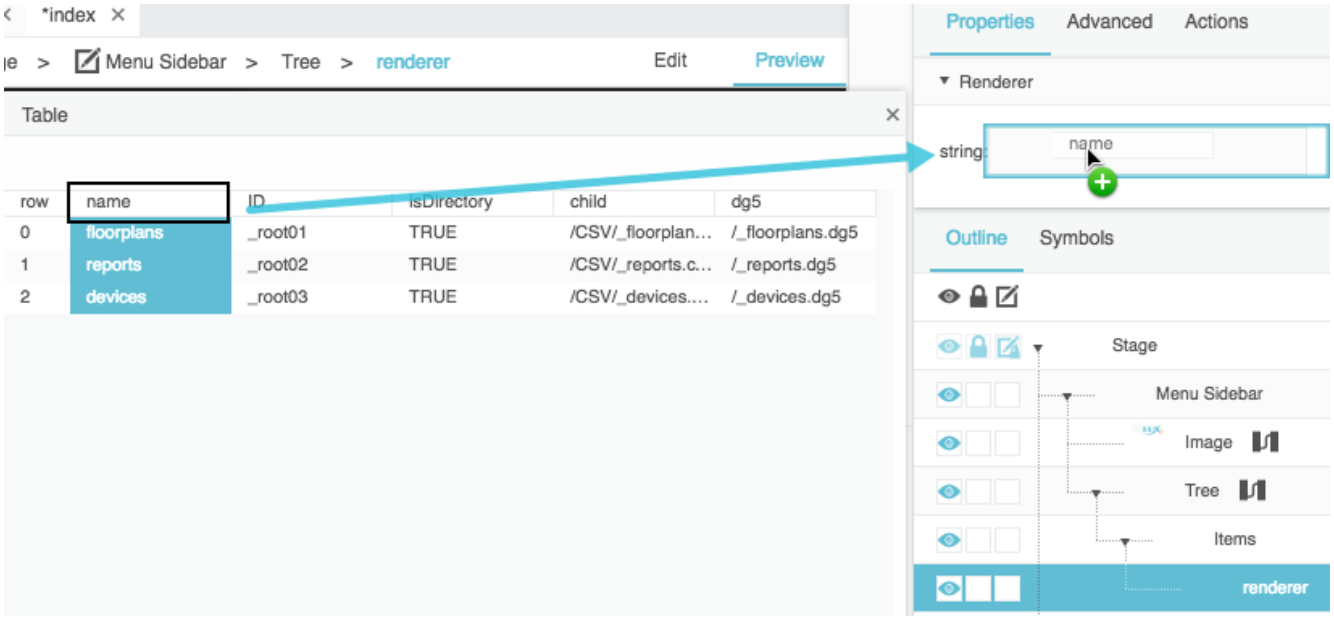
Optionally, the tree component can render an instance of a visual symbol for each item in the tree.

Within the symbol, Horizontal layout is strongly recommended. Symbol contents might not be displayed at all in Vertical or Absolute layout.

If you use a visual symbol, you can bind source table columns to symbol parameters in the tree items renderer. To do so:

1. In the Tree properties, invoke the value of the **Table** property to open the top-level source table.
2. Expand **Tree** > **Items** > **renderer** in the Outline.

3. Drag column headers from the source table to properties in the Property Inspector, as shown in the following image.



The symbol parameters also must be bound to properties inside the symbol.

For an example, see the tree tutorials.

Important

Whenever the column names change, for example if your query changes, you must re-bind the changed table columns to the renderer.

Tip

If the visual symbol has a default parameter, this parameter is automatically used as the **name** property of the tree. This shortcut works only if the dataflow symbol has been defined. To set a default symbol parameter, right-click the parameter in the parameters list in symbol editing mode and choose **Make Default**. See Symbol.

About Selection

This section configures how to configure tree selection, and it describes the **Selected Items** string and the **Selected Items Data** table.

How to Configure Selection

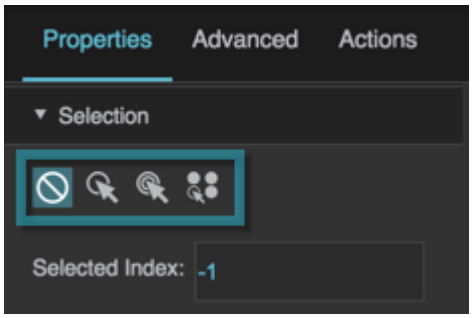
You can configure the selection behavior allowed by the tree. For example, the tree can allow one item to be selected, or it can allow multiple items to be selected. Selection behaviors are described here.

To configure tree selection:

1. In the Outline, select **Tree**.
2. Expand the **Selection** properties.

3. Choose a selection behavior.

   The following image shows the selection behaviors. See also: Selection Properties.



Selected Items and Selected Items Data

The **Selected Items** property is a comma-separated list of IDs of the selected items in the tree.

The **Selected Items Data** property is a table containing the source table rows for all currently selected tree items. This property allows you to pass data for selected items through the tree component to other parts of the project. For example, the tree tutorials include an example of using the selected items to drive a repeater. You can also perform operations on this table in the dataflow, using Table Operations dataflow blocks or the Script dataflow block. See also: Scripting and Syntax.

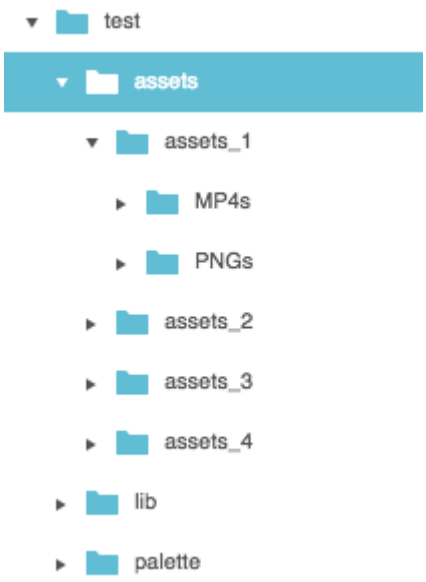
For more details about both of these properties, see Selection Properties.

Tutorial: Create a Simple Tree to List Files

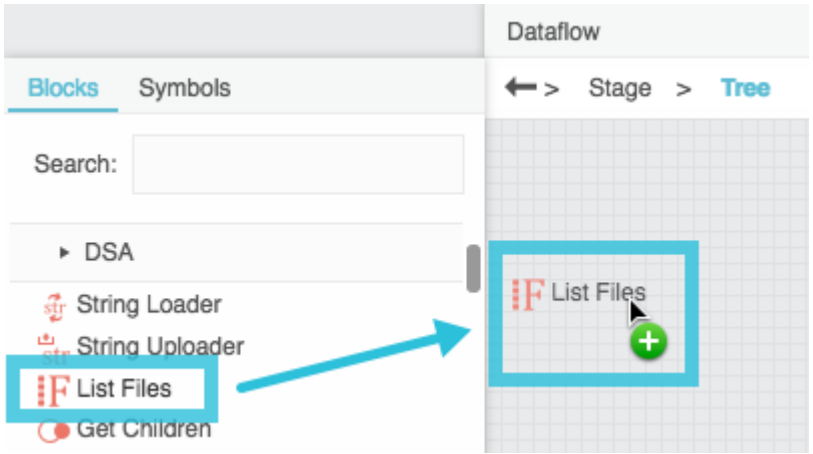
These steps create a tree that displays the directory structure of this DGLux5 project.

1. In a new project, add the following sub-folders in the `assets` folder:
    1. In the `assets` directory, add `assets_1`, `assets_2`, `assets_3`, and `assets_4`.
    2. In the `assets_1` directory, add `MP4s` and `PNGs`.

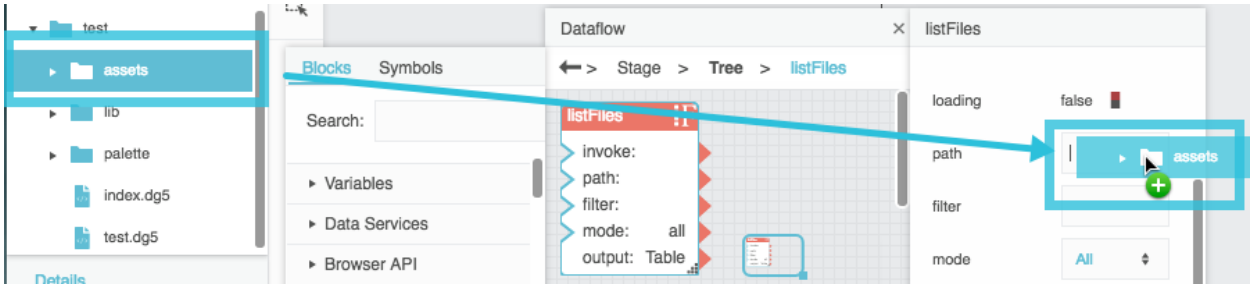
       The following image demonstrates the finished directory structure.



2. Right-click in the Outline or Document window, and select **Insert** > **Components** > **Tree**.
3. Add the top-level source table:
    1. In the dataflow model for the tree, add a List Files block, as shown in the following image.

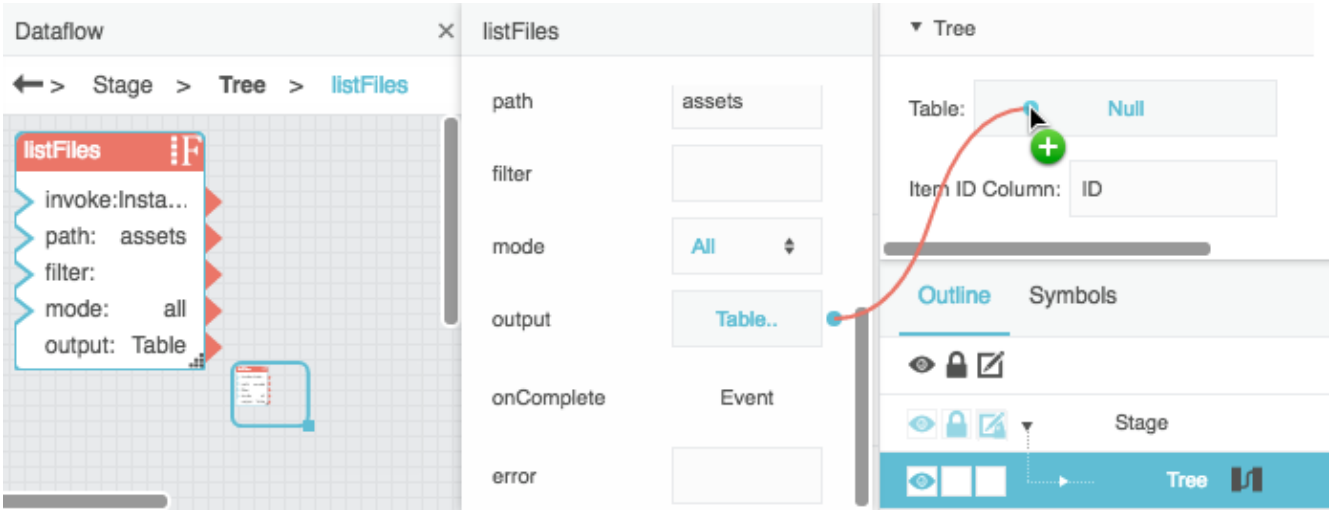


2. Drag the `assets` folder from the Project Panel, and drop it on the **path** property of the block, as shown in the following image.



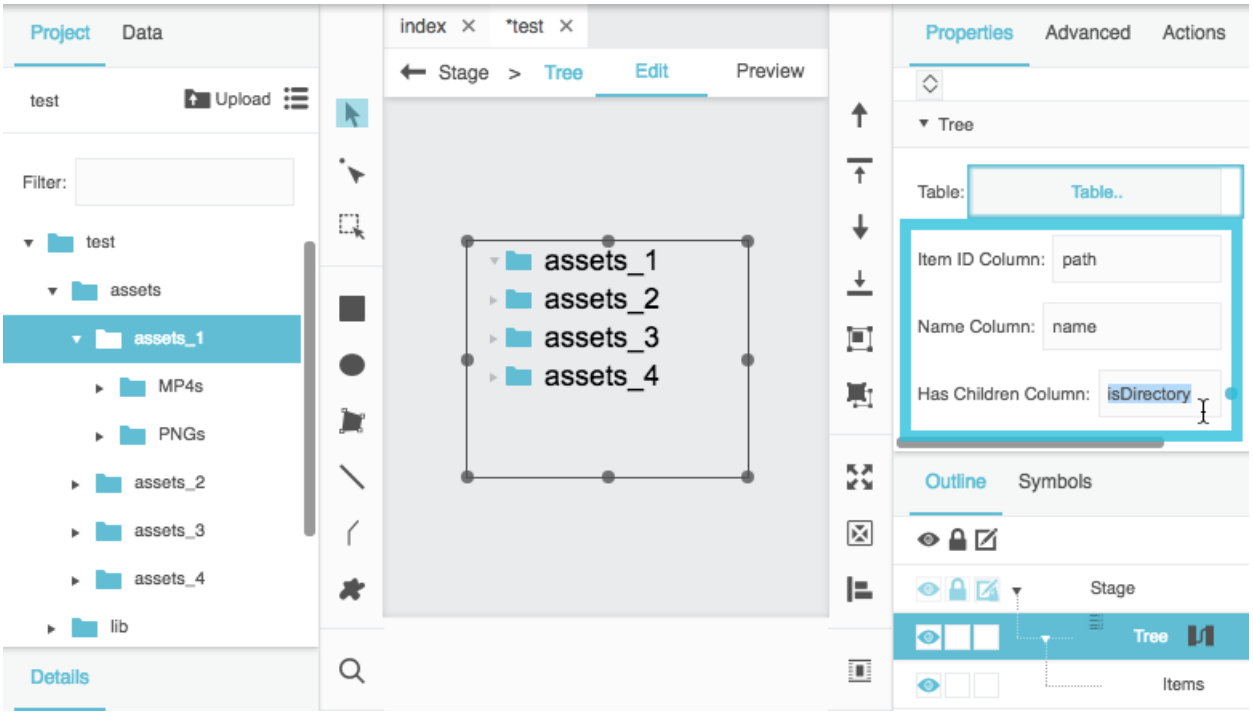
3. In the block properties, click **Invoke**.

4. Bind the List File block's **output** property to the tree's **Table** property, as shown in the following image.



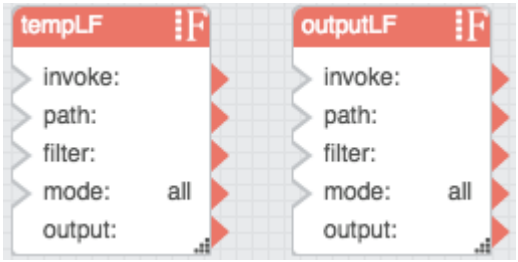
5. In the Property Inspector, type these column names:
   - Type `path` for **Item ID Column** (replace `ID`).
   - Type `name` for **Item Name Column**.
   - Type `isDirectory` for **Has Children Column**.

   The top-level folders now appear in the tree component, as shown in the following image.

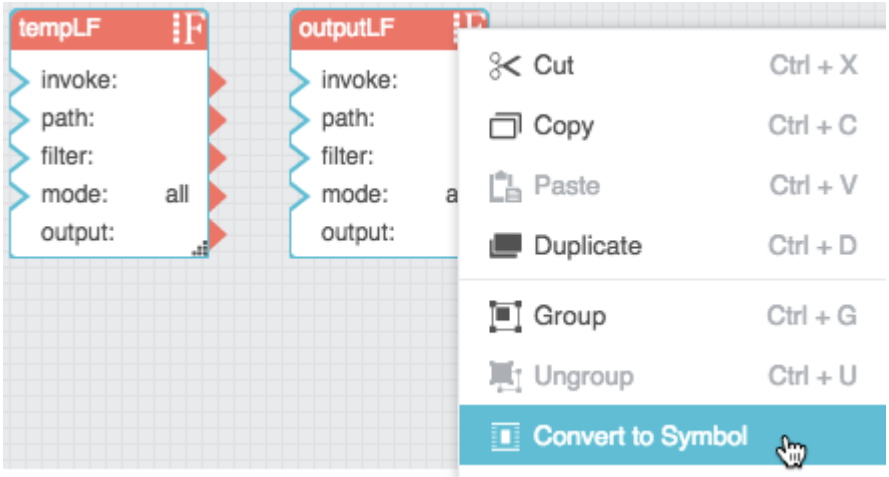


6. Create a dataflow symbol:
    1. In dataflow for the Stage, add two List Files blocks.

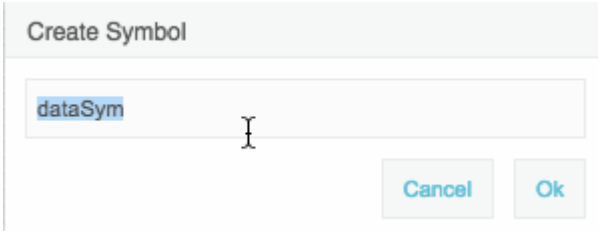
    2. Name the List Files blocks `tempLF` and `outputLF`, as shown in the following image.



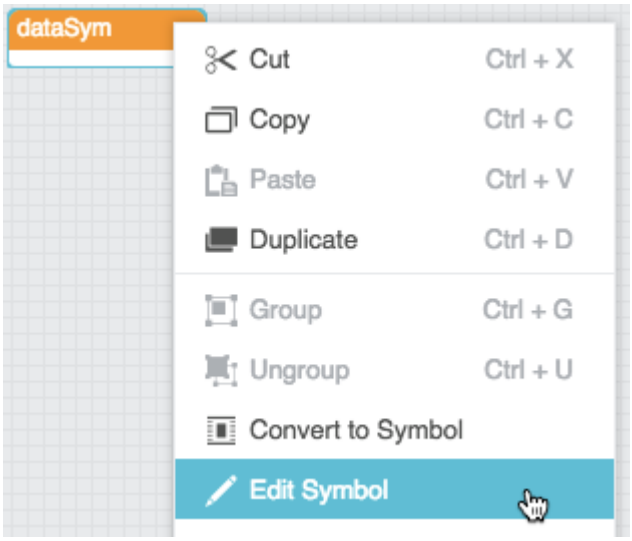
    3. Select the two dataflow blocks, right-click, as shown in the following image, and select **Convert to Symbol**.



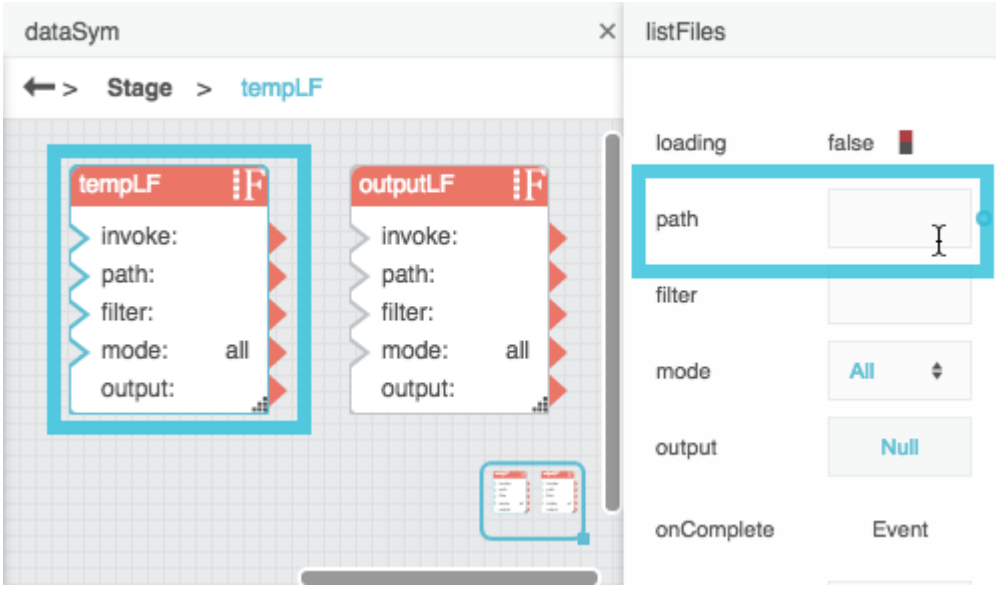
4. Name the symbol `dataSym`, and click OK, as shown in the following image.



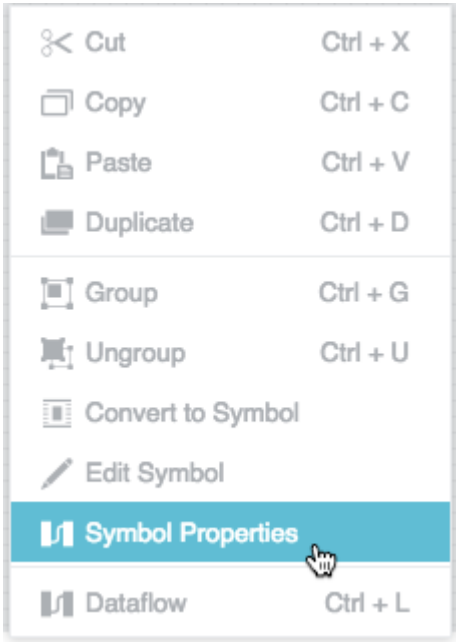
5. Right-click the new symbol block that appears, and select **Edit Symbol**, as shown in the following image.



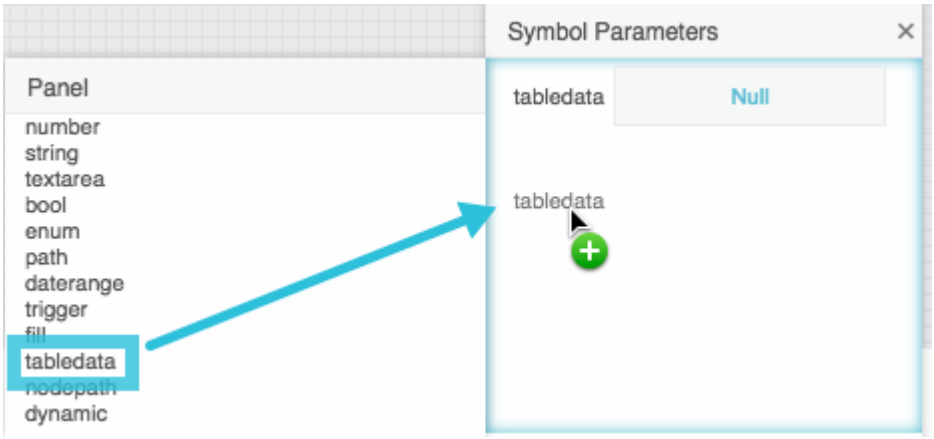
6. Click in the **path** property field of `tempLF`, and press Return, as shown in the following image.



7. Right-click the dataflow window, and choose **Symbol Properties**, as shown in the following image.



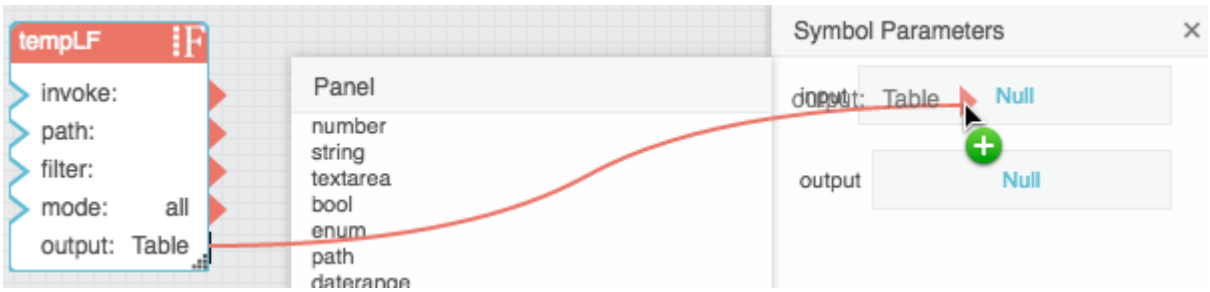
8. Add two tabledata parameters, as shown in the following image.



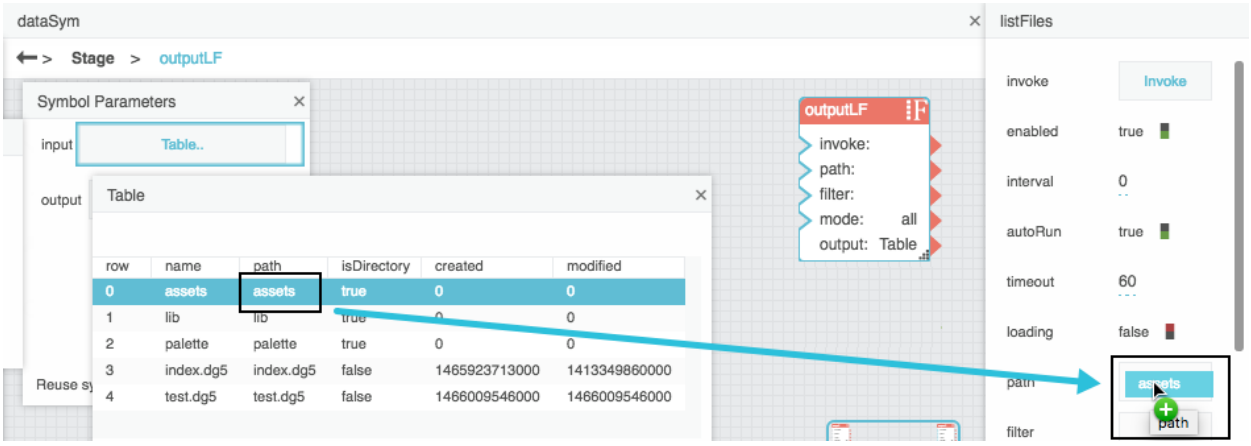
9. Name the new parameters `input` and `output`, as shown in the following image.

Important: If you do not use these exact names, the symbol might not collect data for the tree. Names are case sensitive.

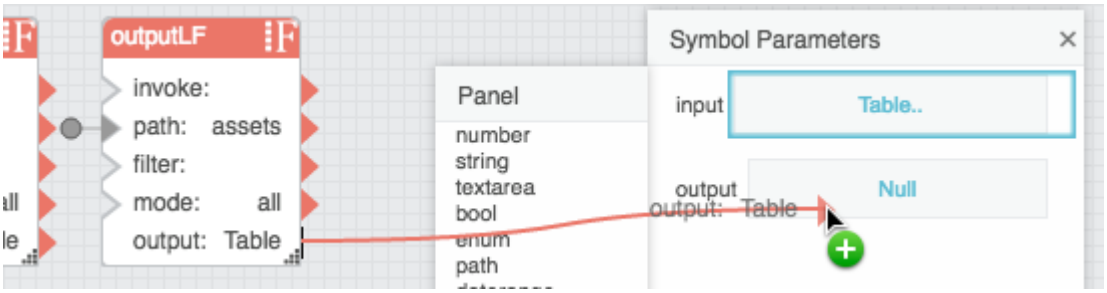
10. Bind the output table of `tempLF` to the `input` parameter, as shown in the following image.



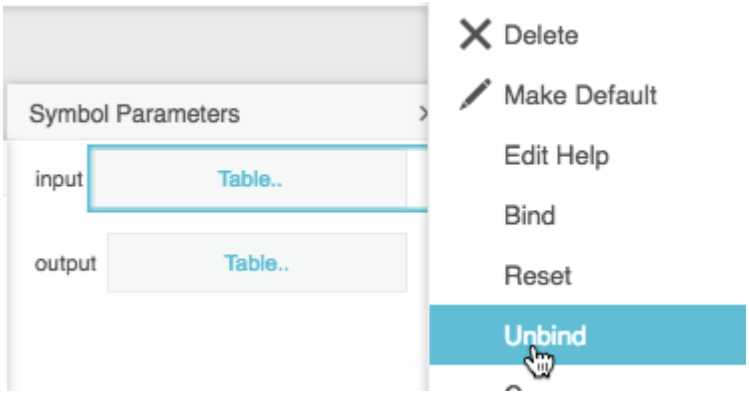
11. Open the `input` parameter table, and drag the cell from row 0, column **path**, to the **path** property of `outputLF`, as shown in the following image.



12. Bind the **output** property of `outputLF` to the `output` parameter, as shown in the following image.



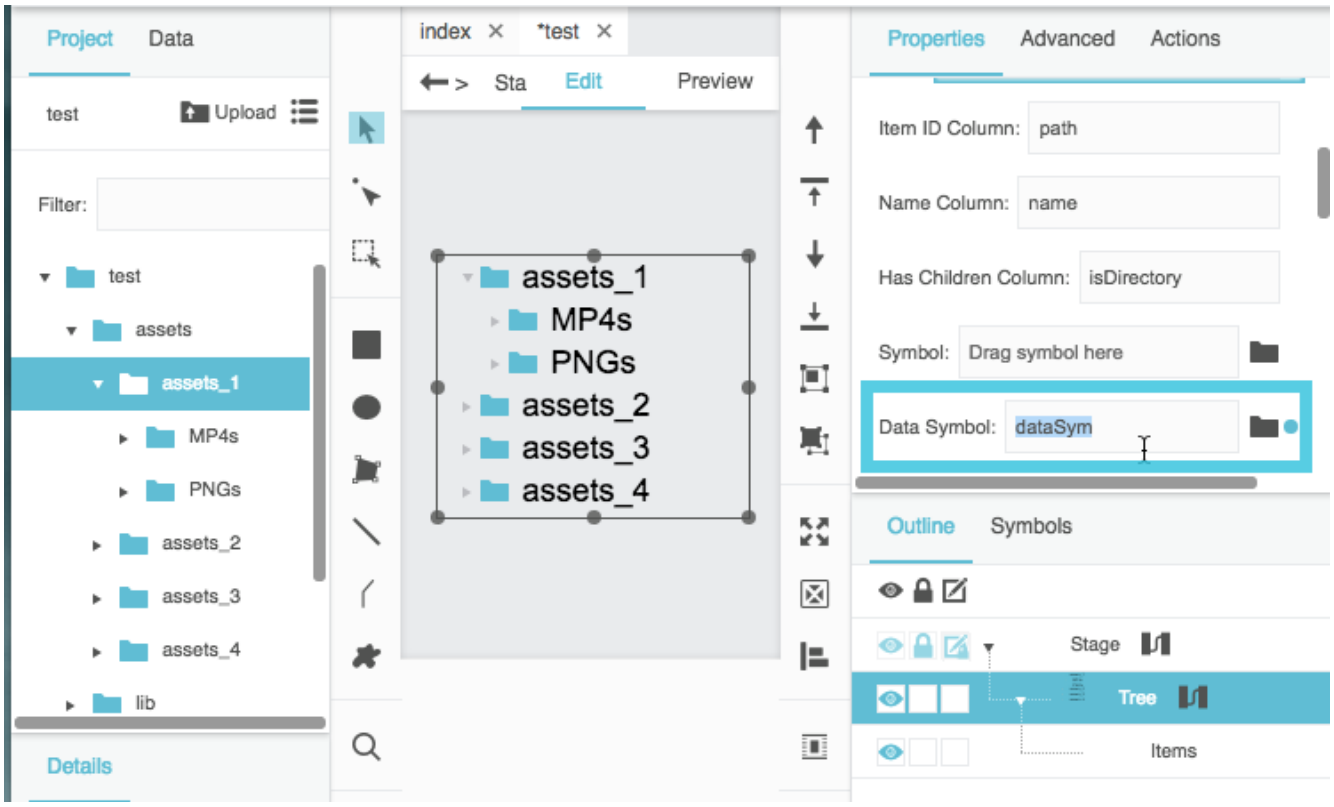
13. Unbind the `input` symbol parameter, as shown in the following image.



The symbol parameter values will become null.

14. Delete the `tempLF` block.
15. Click **OK** to save changes and exit dataflow symbol editing mode.
16. Delete the instance of the dataflow symbol from the Stage dataflow.

7. Select the tree in the Outline, and in the Property Inspector, type `dataSym` for the **Data Symbol** property, as shown in the following image.



All descendants of the root now can appear in the tree component.

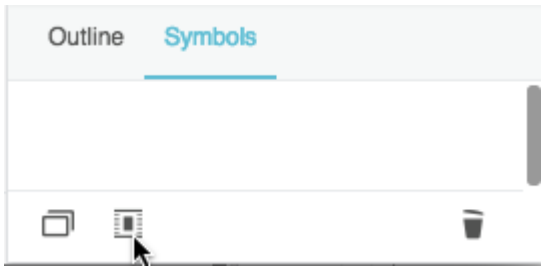
Tutorial: Render a Visual Symbol in the Tree

These steps create a visual symbol to use in the tree. See also: About the Visual Symbol.

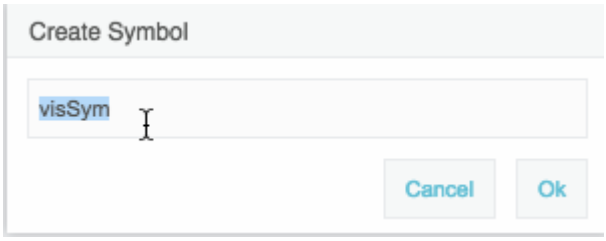
1. Create the tree described in Tutorial: Create a Simple Tree to List Files.
2. In the `assets` directory, add some image files. For example, add images in the following locations:
    1. In `assets/assets_1/PNGs`, add images `pngA.png` and `pngB.png`
    2. In `assets/assets_2`, add image `pngC.png`
    3. In `assets/assets_3`, add image `pngD.png`
    4. In `assets/assets_4`, add image `pngE.png`

    The following image shows the completed file structure.

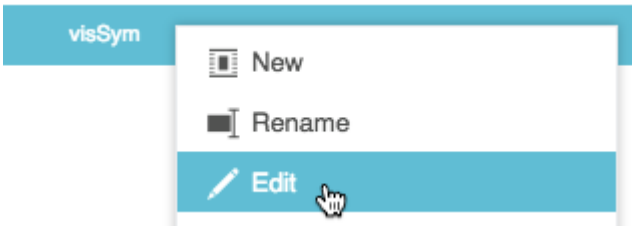
3. Create the visual symbol:
    1. In the Symbols panel, click **New**, as shown in the following image.



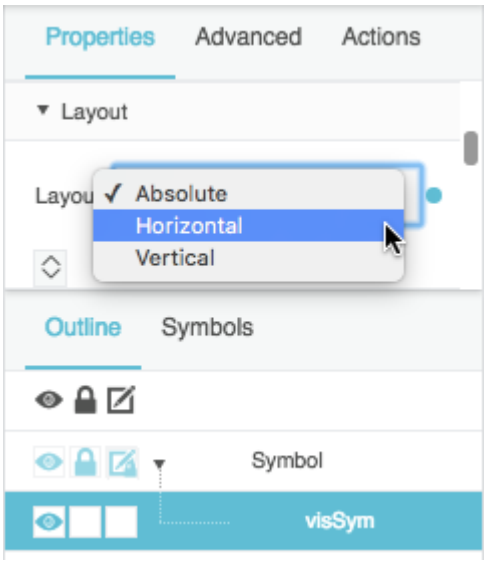
    2. Name the new symbol `visSym`, as shown in the following image.



3. In the Symbols panel, right-click **visSym**, and choose **Edit**, as shown in the following image.



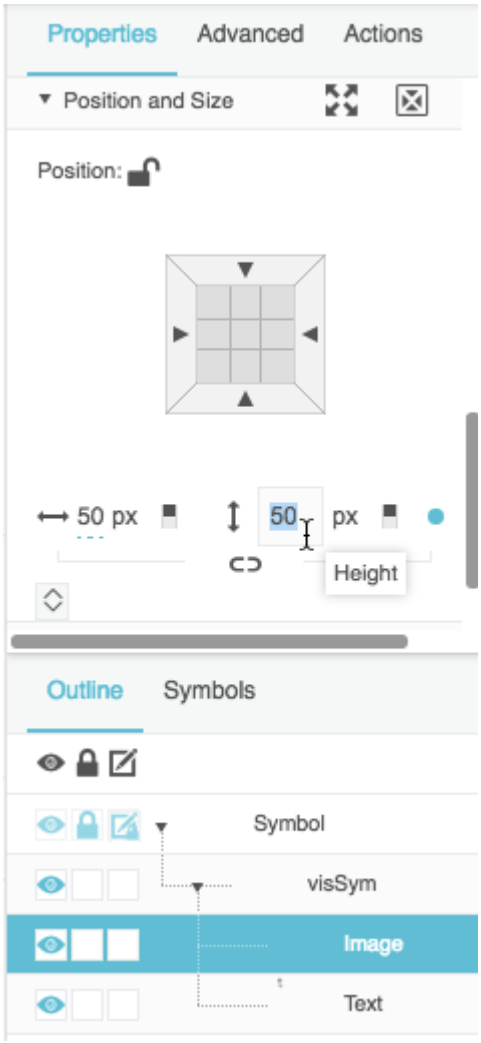
4. Set the Layout property of **visSym** to Horizontal, as shown in the following image.



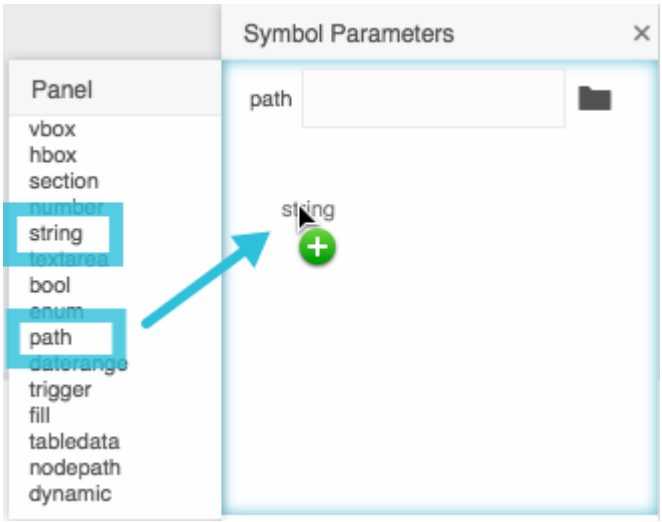
Important

Horizontal layout is strongly recommended for visual symbols used in trees. Symbol content might not be displayed in Vertical or Absolute layout.

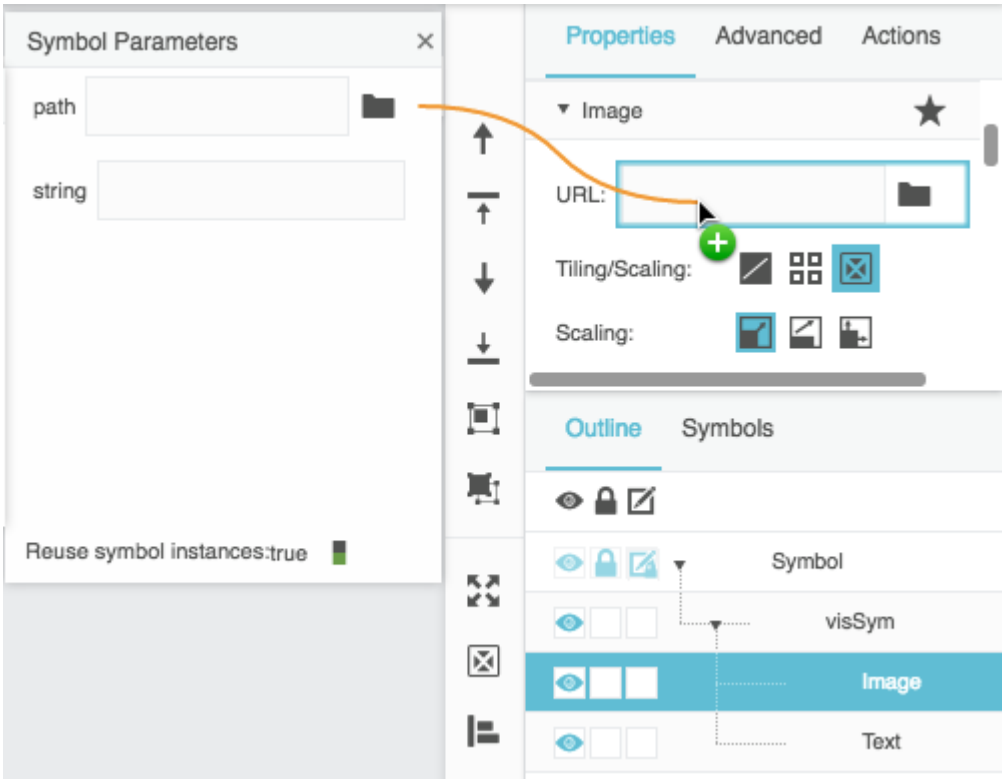
5. Add an image component and a text component to **visSym**.
6. Set the image height and width to 50 pixels, as shown in the following image.



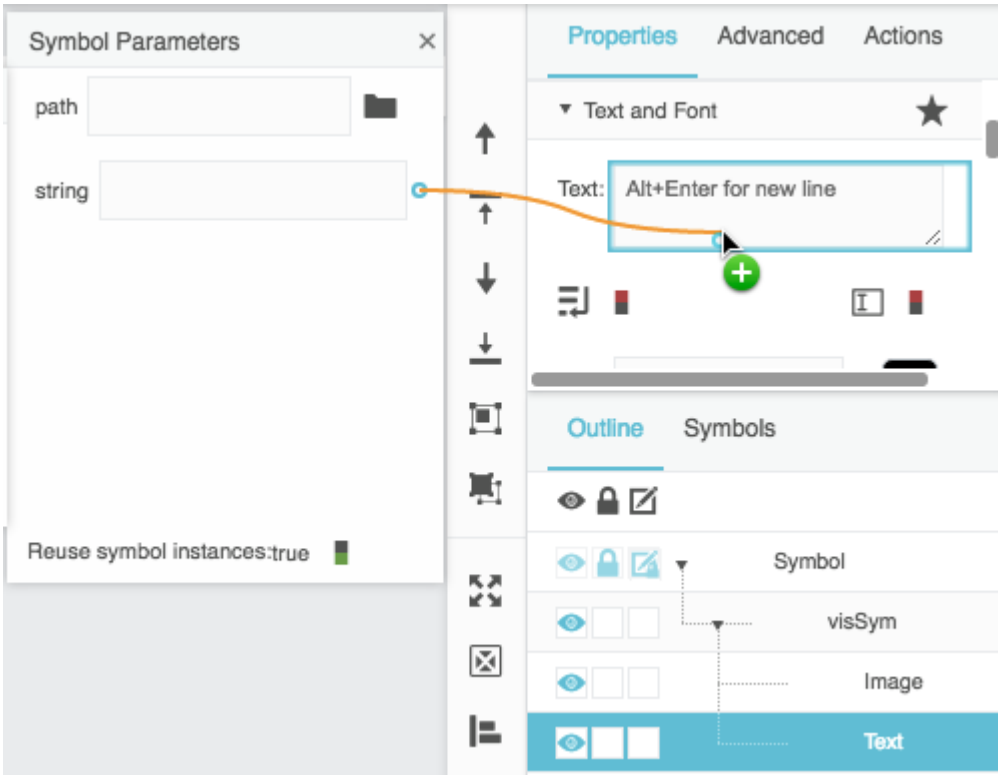
7. Add a path parameter and a string parameter to **visSym**, as shown in the following image.



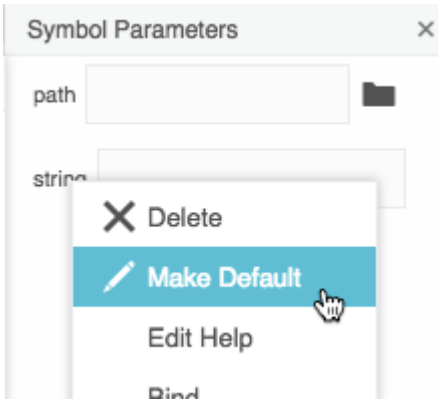
8. Bind the path parameter to the image component's **URL** property, as shown in the following image.



9. Bind the string parameter to the text component's **Text** property, as shown in the following image.

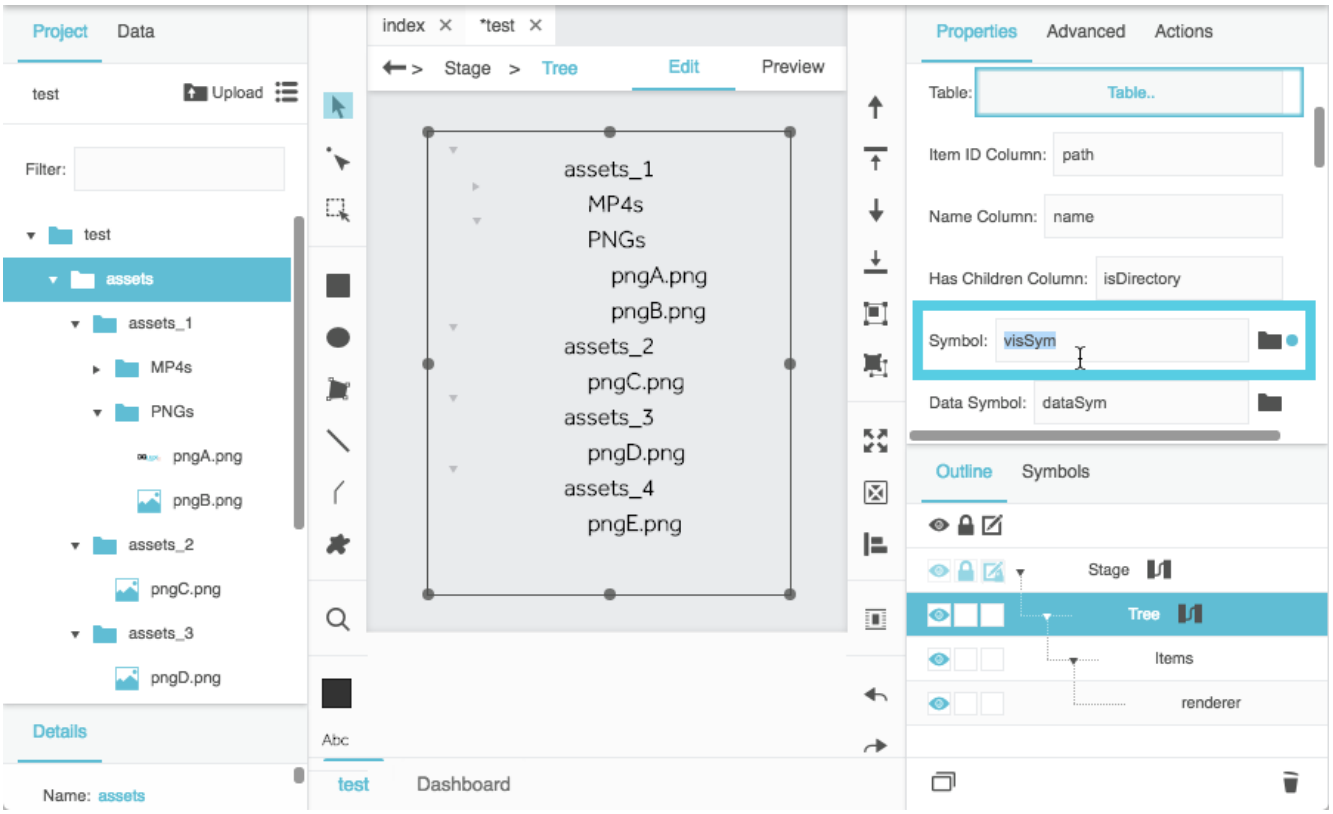


10. Set the string parameter as default, as shown in the following image.



11. Click **OK** to exit symbol editing mode.

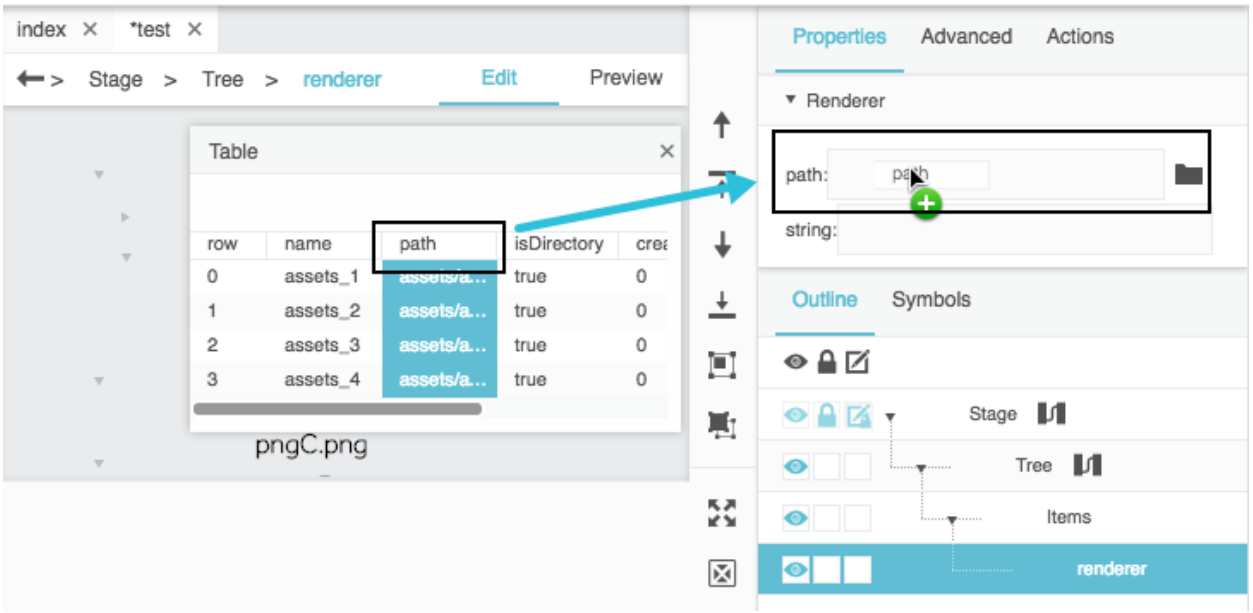
4. In the Tree properties, type `visSym` as the **Symbol** property, as shown in the following image.



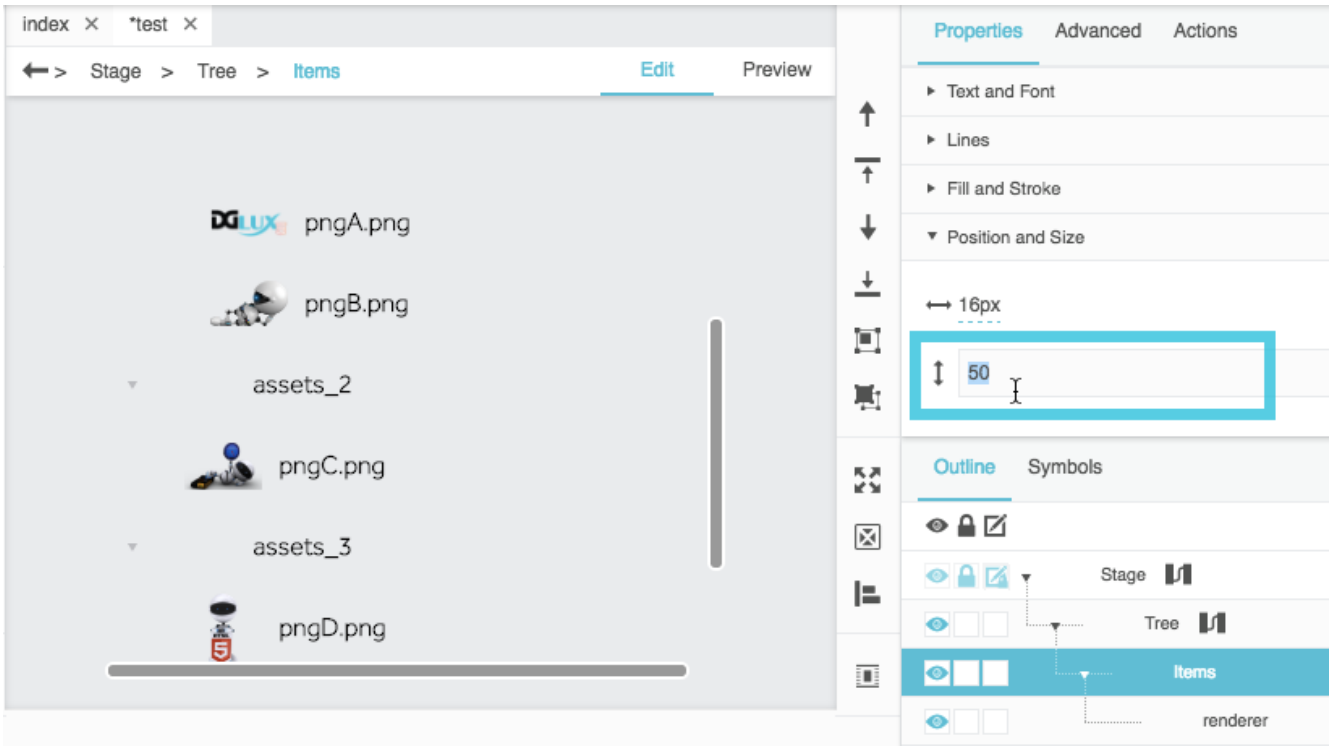
Now a symbol instance is rendered for each tree item.

5. Bind the table column to the renderer:
    1. Open the tree's top-level source table.
    2. In the Outline, expand **Tree** > **Items** > **renderer**.

    3. Drag the header of the **path** column to the **path** property of the renderer, as shown in the following image.



6. In the Outline, select **Tree** > **Items**.
7. Set the tree item height to 50 pixels, as shown in the following image.

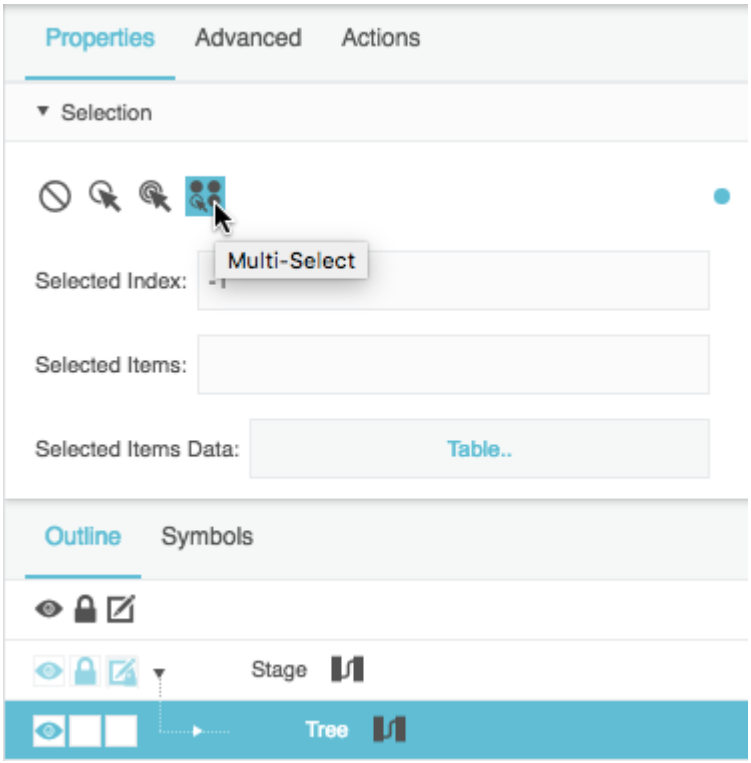


Tutorial: Use Tree Selection

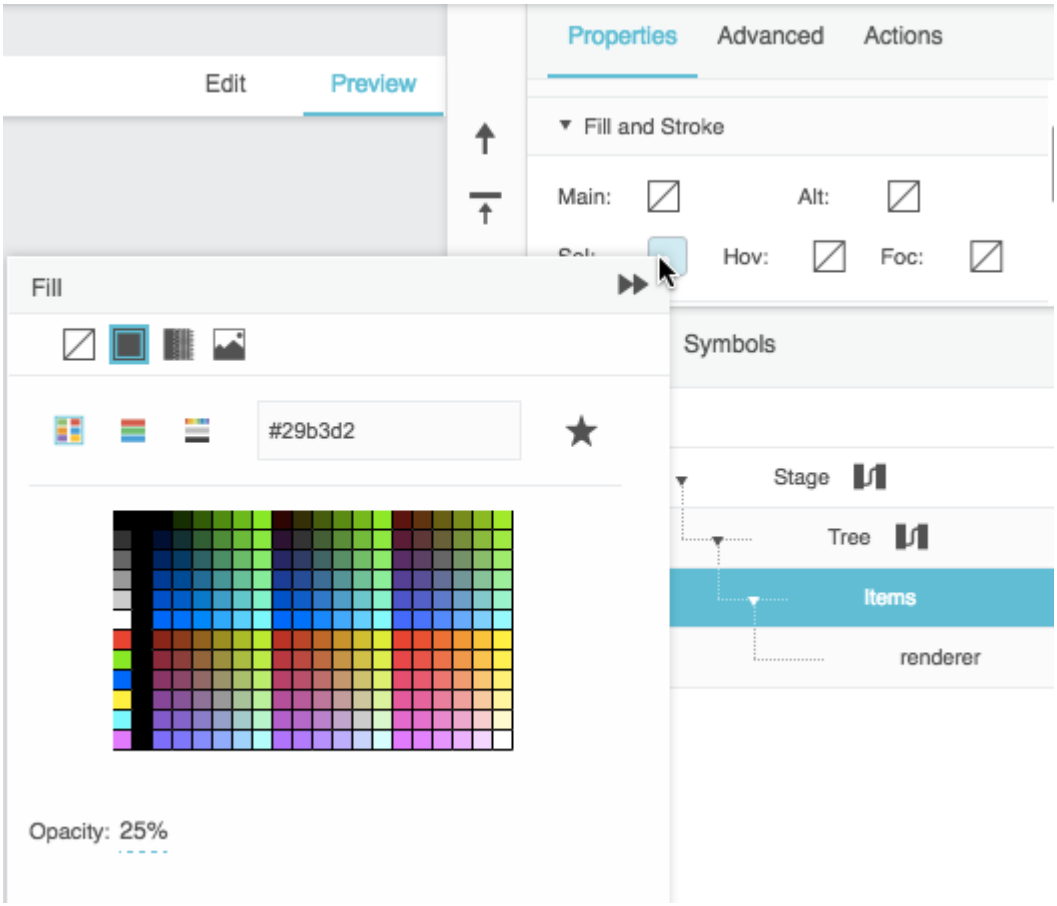
These steps cause the currently selected tree items to drive a repeater.

1. Create the tree, as described in Tutorial: Create a Simple Tree to List Files.
2. Render the visual symbol, as described in Tutorial: Render a Visual Symbol in the Tree.

3. Set the tree's **Selection Behavior** property to **Multi-Select**, as shown in the following image.

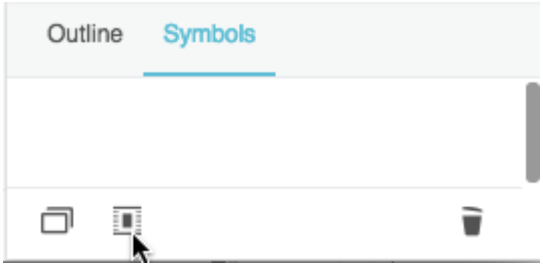


4. Set the tree items' **Selected Items Fill** property to a value, as shown in the following image.

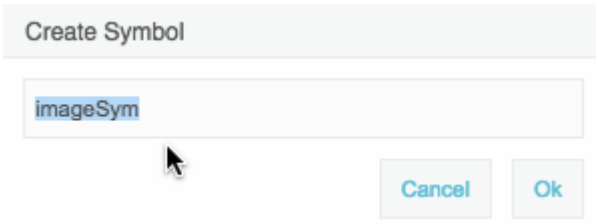


5. Create the symbol for the repeater:

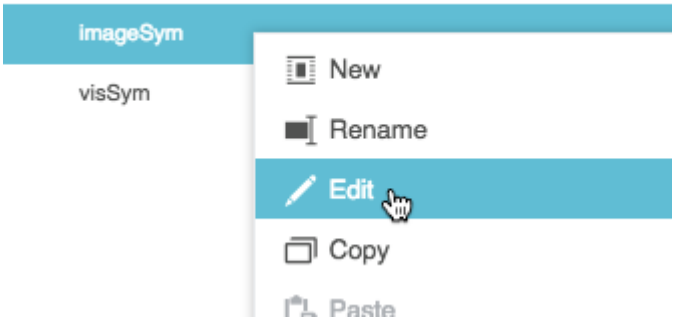
    1. In the Symbols panel, click **New**, as shown in the following image.



    2. Name the new symbol `imageSym`, as shown in the following image.

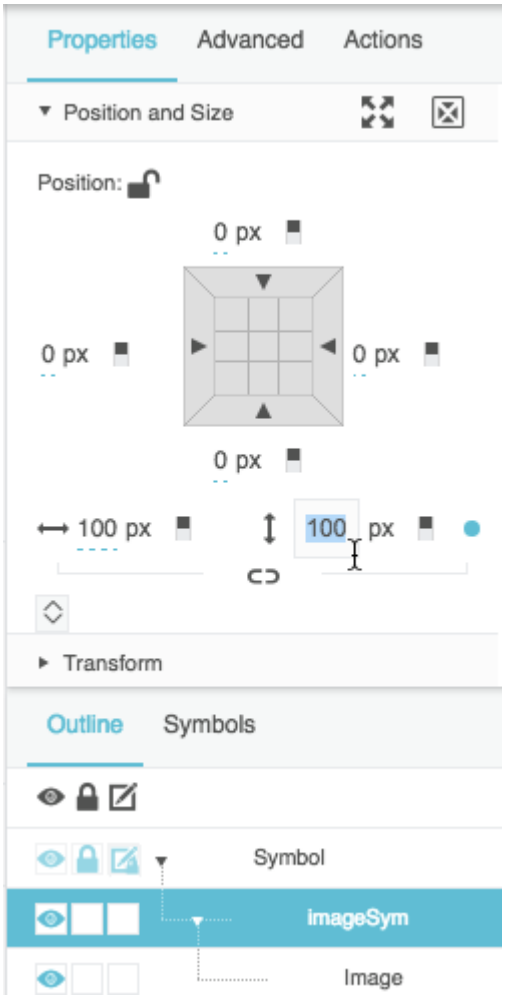


    3. In the Symbols panel, right-click **imageSym**, and choose **Edit**, as shown in the following image.

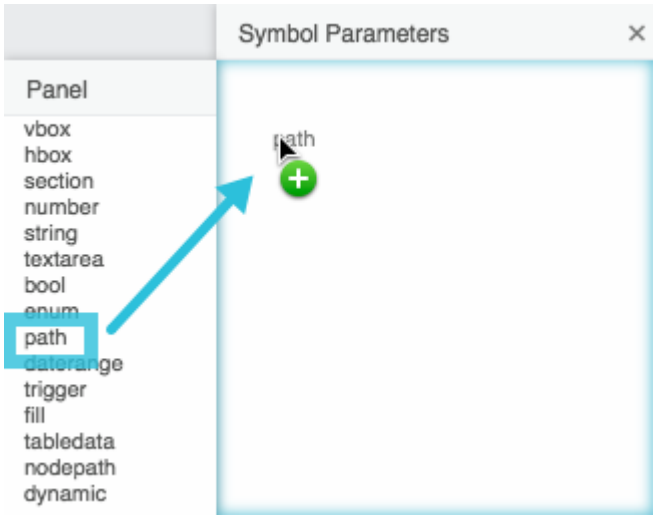


4. Add an image component to the **imageSym** symbol.

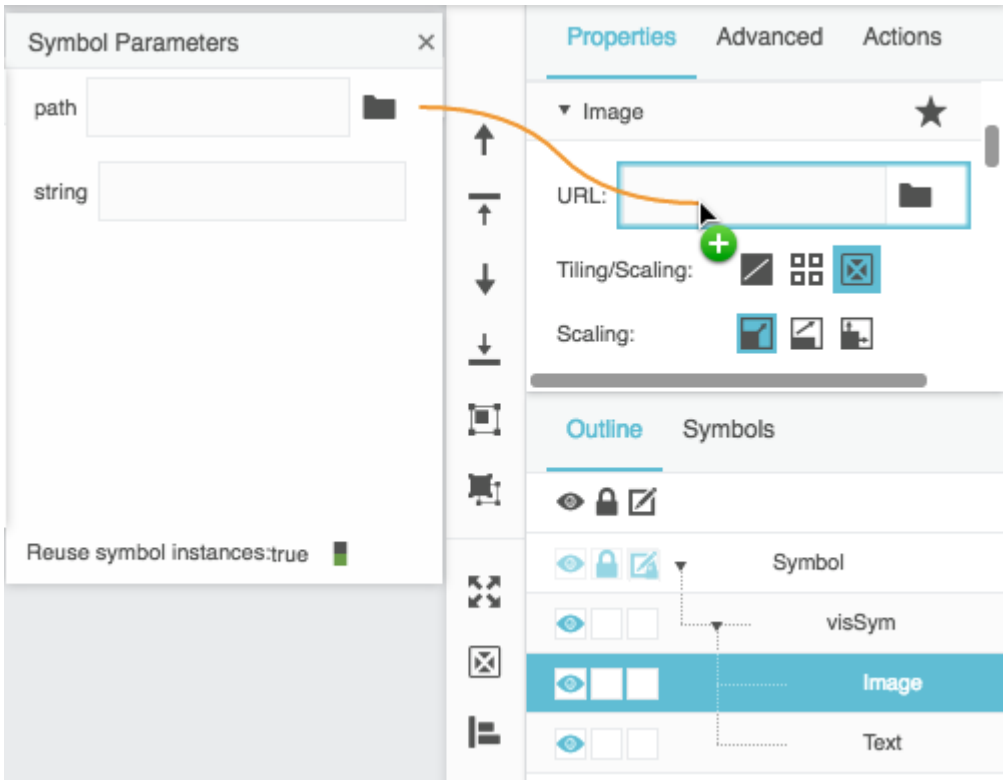
5. Set the width and height of the **imageSym** symbol to 100 pixels, as shown in the following image.



6. Add a **path** parameter to the **imageSym** symbol, as shown in the following image.



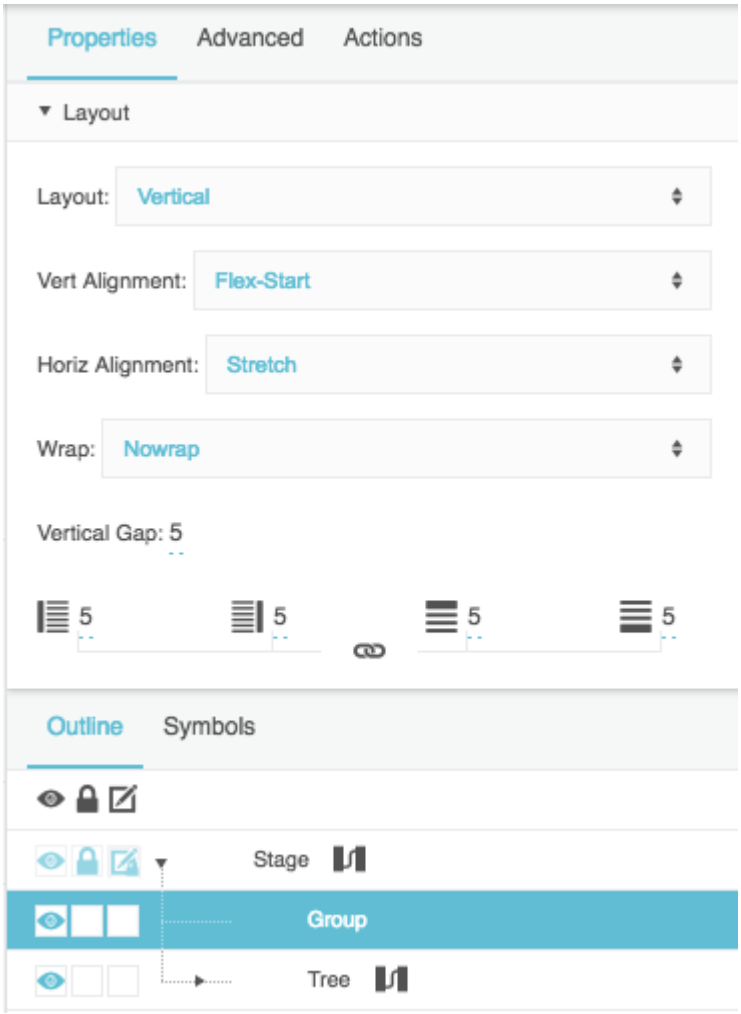
7. Bind the **path** symbol parameter to the image URL inside the symbol, as shown in the following image.



8. Click **OK** to exit symbol editing mode.

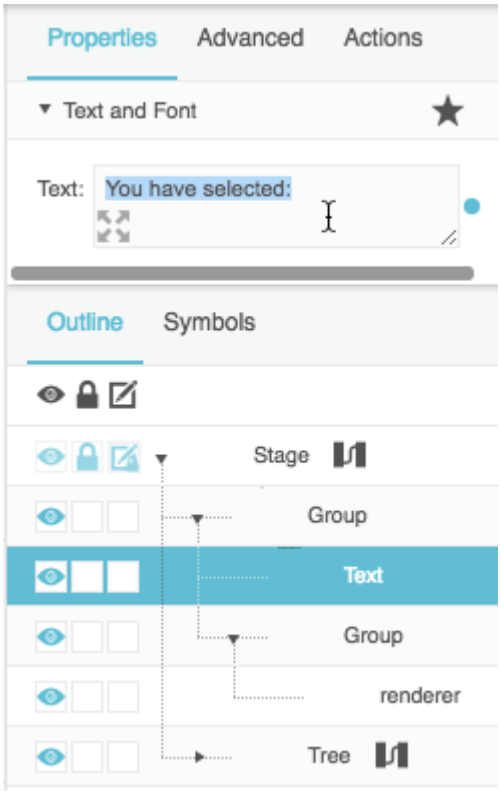
6. Add a group to the stage.
7. Set the group's Layout properties as follows:
    1. Set **Layout** to Vertical.
    2. Set **Horiz Alignment** to Stretch.
    3. (Optional) Add padding and a vertical gap.

    The following image shows the configured layout properties.

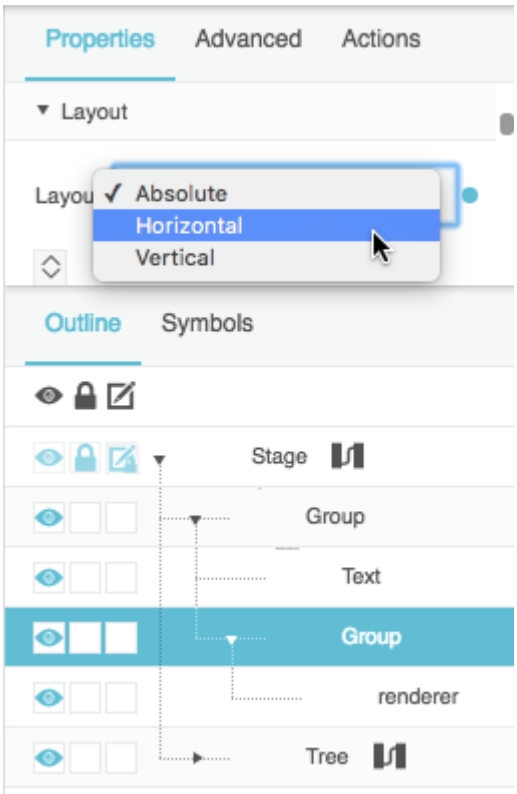


8. Inside the group, add a text component and a group.
9. Set the text component and inner group properties as follows:

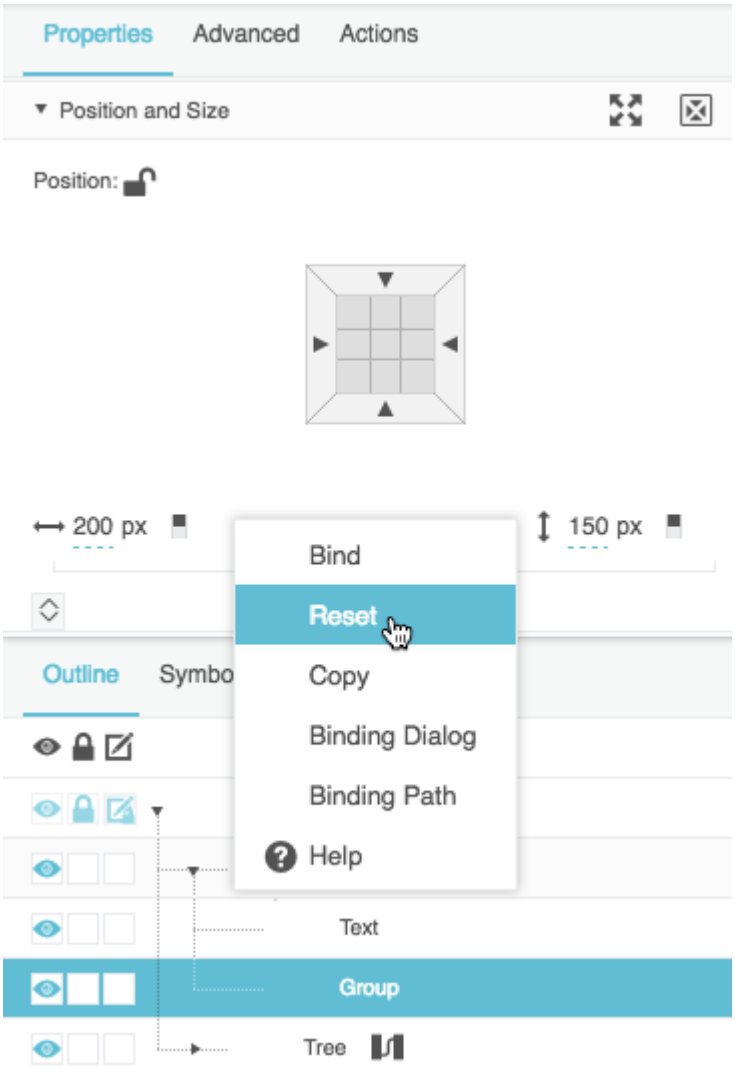
    1. Enter `You have selected:` as the value of the text component's Text property, as shown in the following image.



2. Set the layout of the inner group to Horizontal, as shown in the following image.

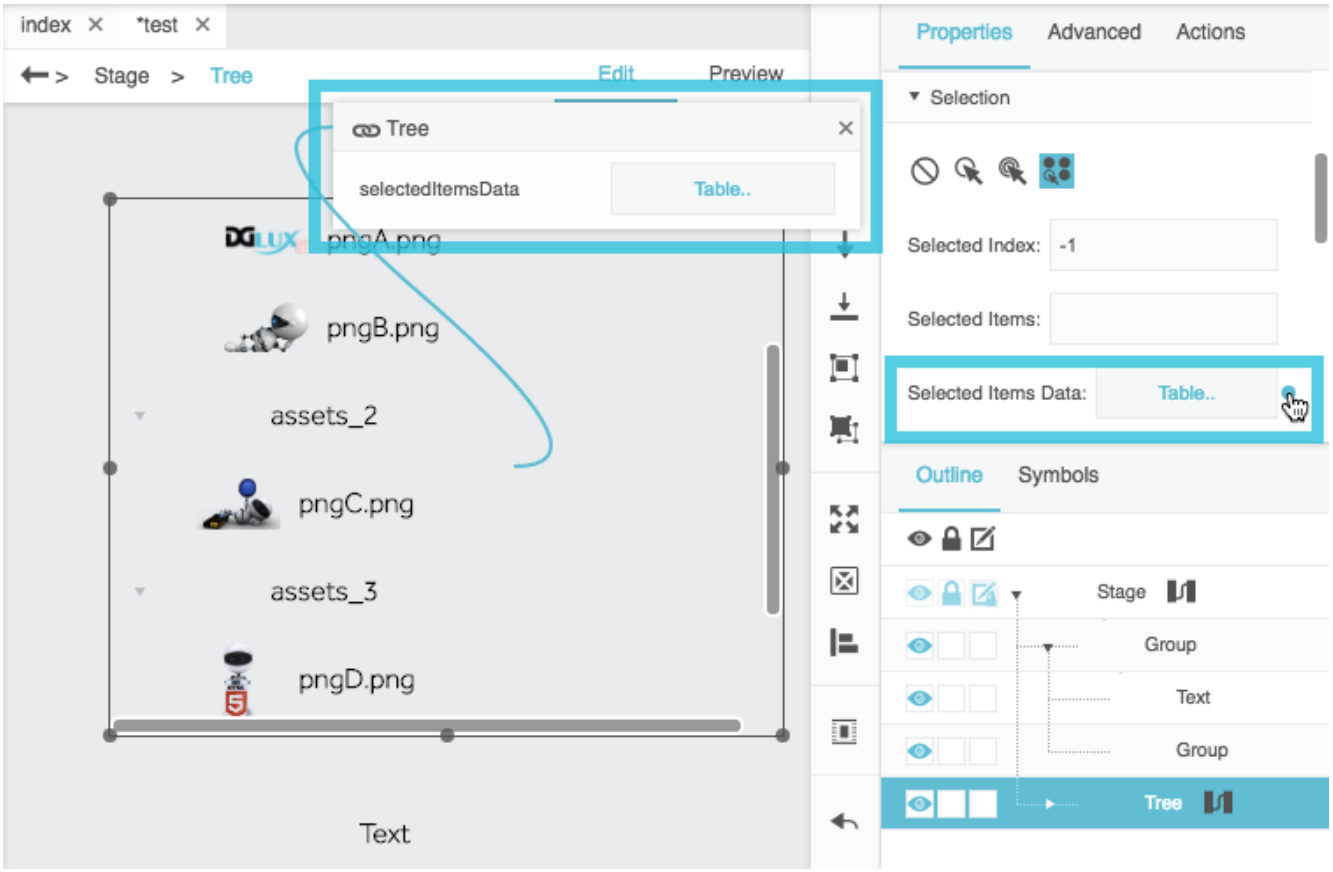


3. Set the width of the inner group to Auto, as shown in the following image.



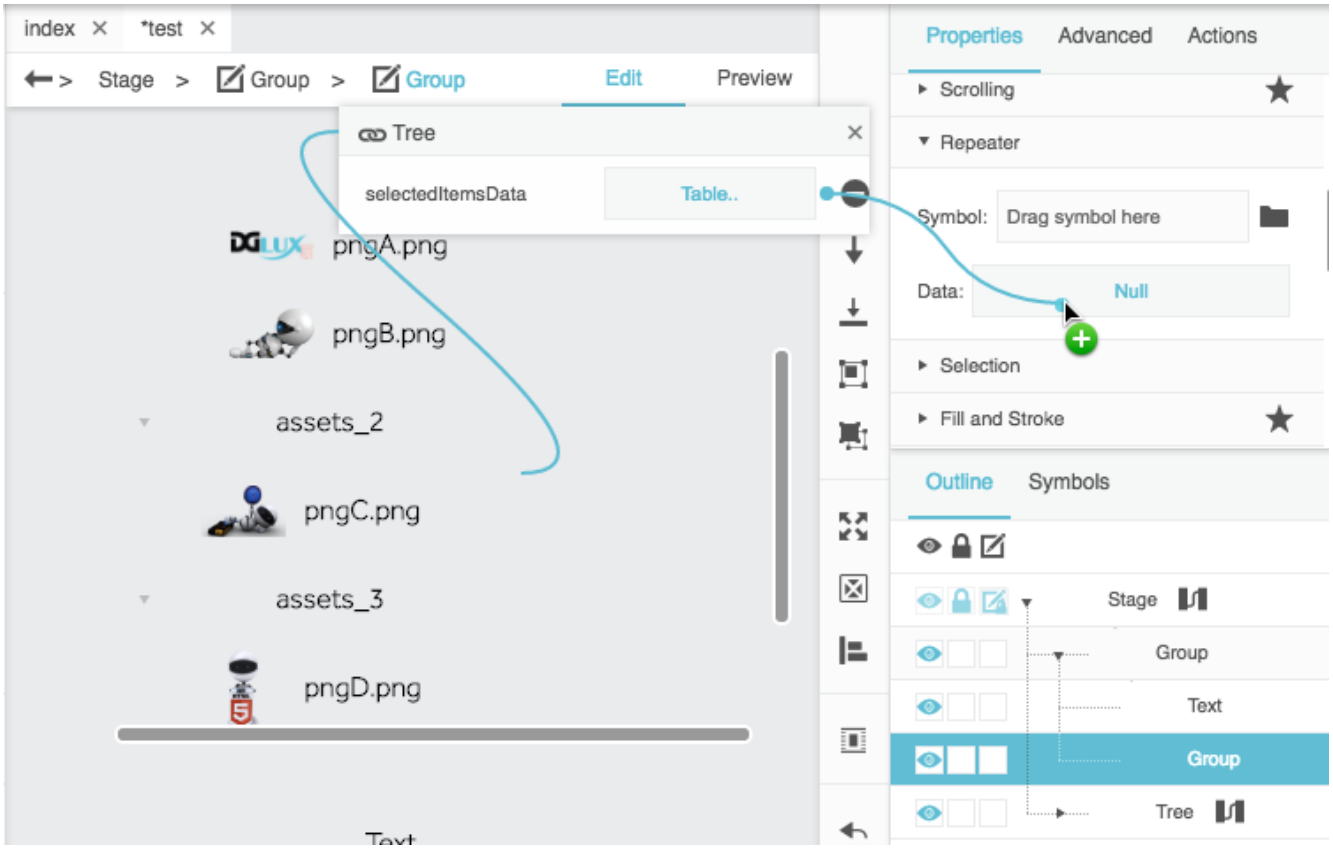
10. Select the tree in the Outline, and expand the tree's Selection properties.
11. Hover over **Selected Items Data** until a blue dot appears, and then double-click the blue dot.

    This opens a small binding pop-up, as shown in the following image.

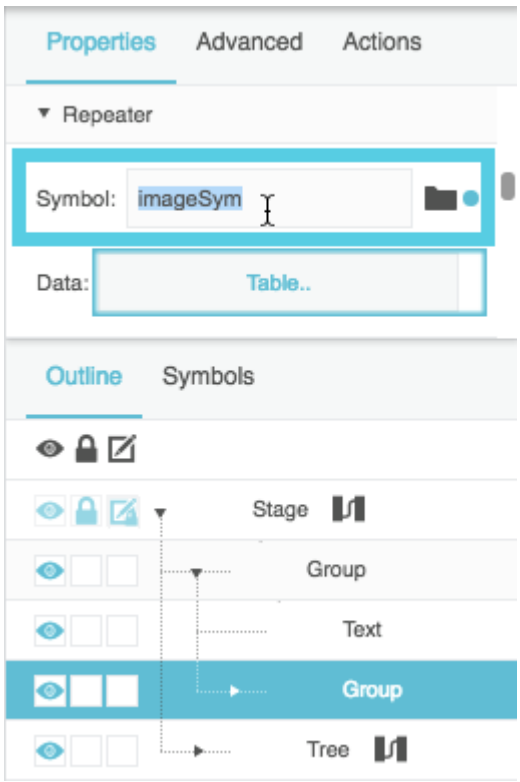


12. Select the inner group in the Outline, and expand the Repeater properties.

13. Bind the **Selected Items Data** property in the small binding pop-up to the **Data** property, as shown in the following image.

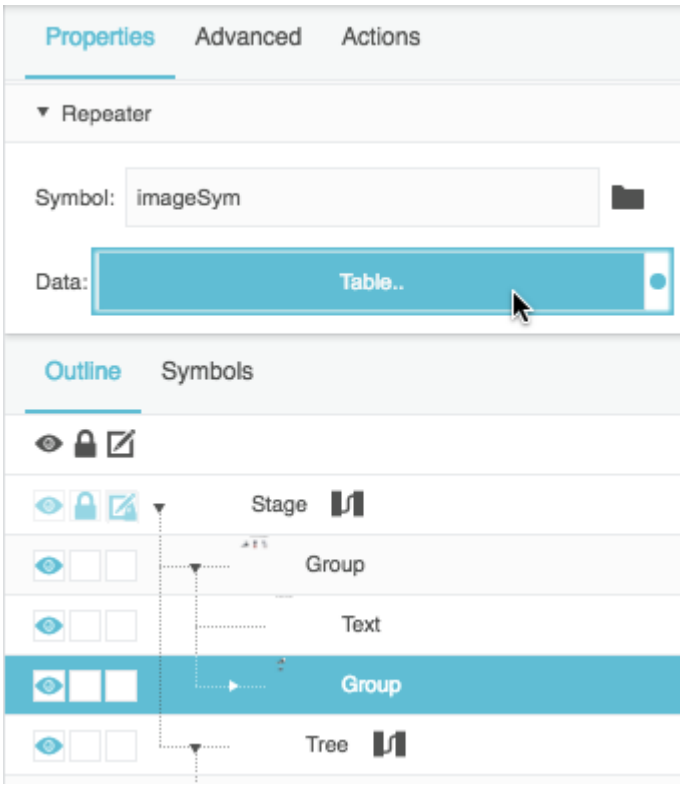


14. In the inner group, type `imageSym` as the **Symbol** property, as shown in the following image.



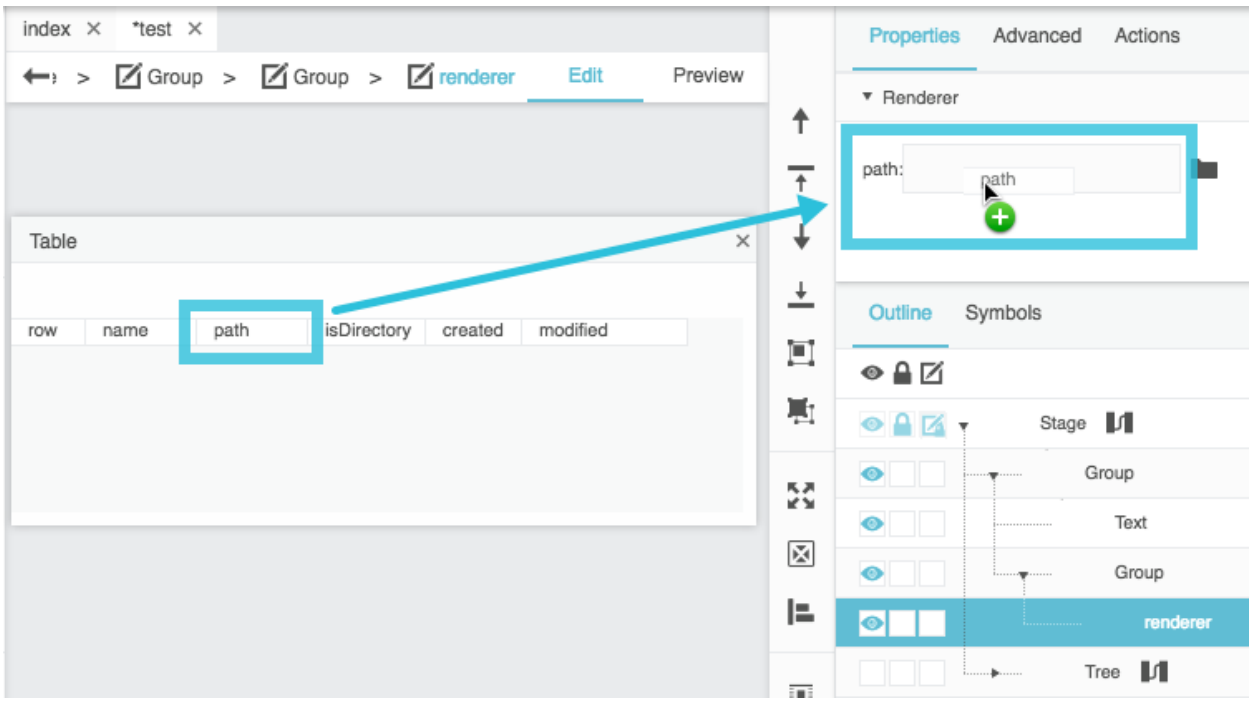
15. Bind the table column to the renderer:

1. Open the table that drives the inner group repeater, as shown in the following image.



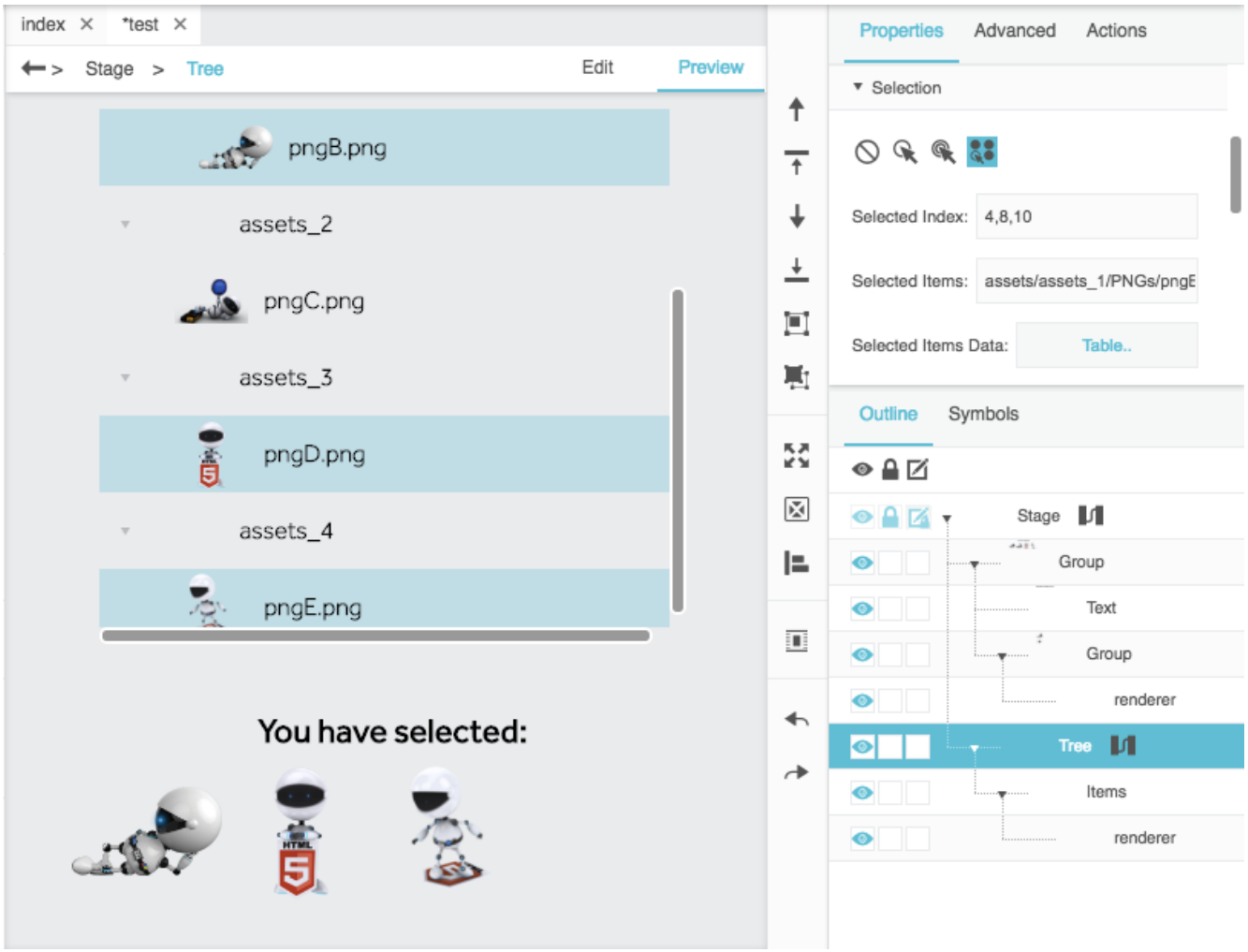
2. In the Outline, select **Group** > **Group** > **renderer**.

3. Bind the **path** table column header to the **path** renderer property, as shown in the following image.



16. In Preview mode, click in the tree to select items. Selected items are displayed in the repeater, as

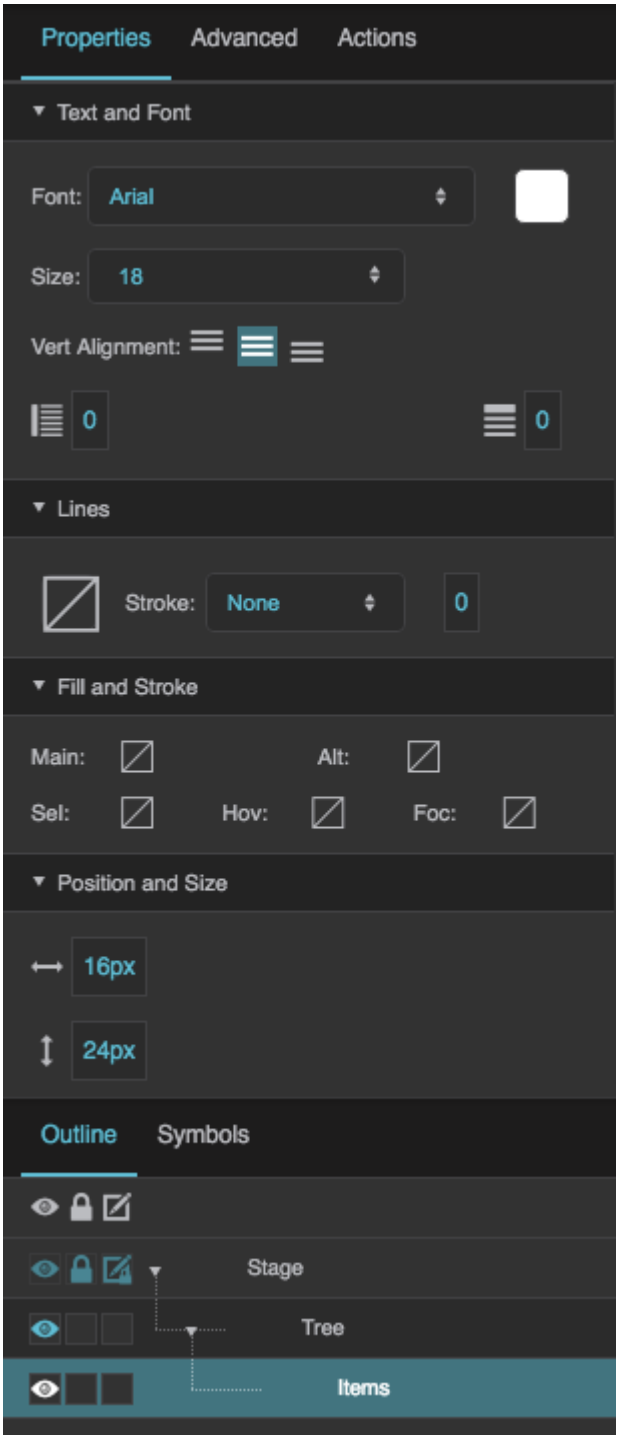
shown in the following image.



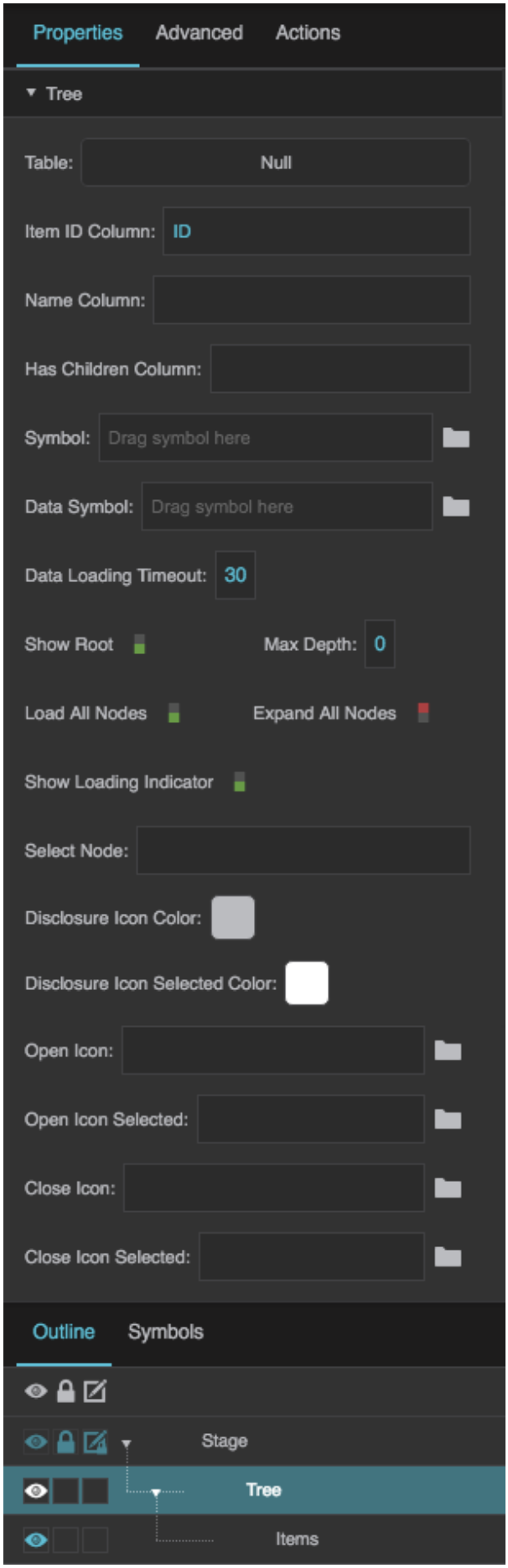
Tree Properties

These properties affect the tree component. For a guide to using the tree component, see Tree.

Trees are also affected by Common Properties.



The Tree Items properties in the Property Inspector


Properties
Advanced
Actions
Tree
Table:
Null
Item ID Column:
ID
Name Column:
Has Children Column:
Symbol:
Drag symbol here
Data Symbol:
Drag symbol here
Data Loading Timeout:
30
Show Root
Max Depth:
0
Load All Nodes
Expand All Nodes
Show Loading Indicator
Select Node:
Disclosure Icon Color:
Disclosure Icon Selected Color:
Open Icon:
Open Icon Selected:
Close Icon:
Close Icon Selected:
Outline
Symbols
Stage
Tree
Items

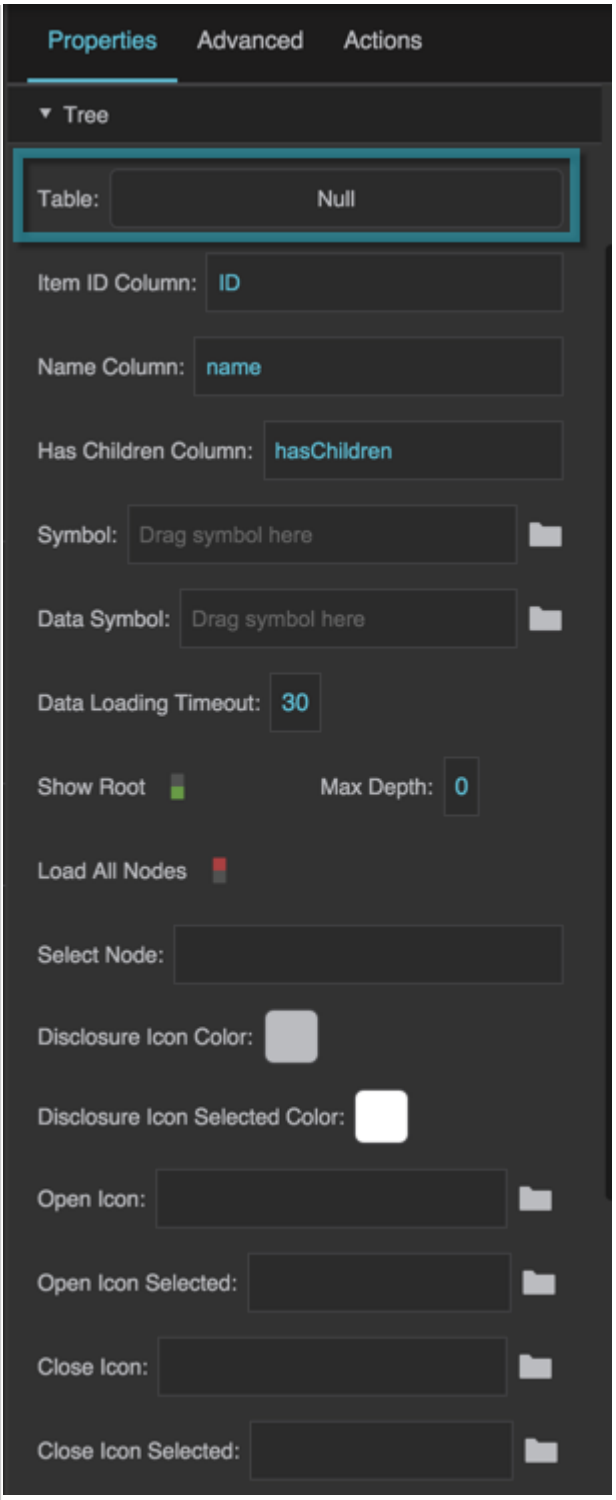
The Tree properties in the Property Inspector

Click to display/hide all elements

Tree Properties

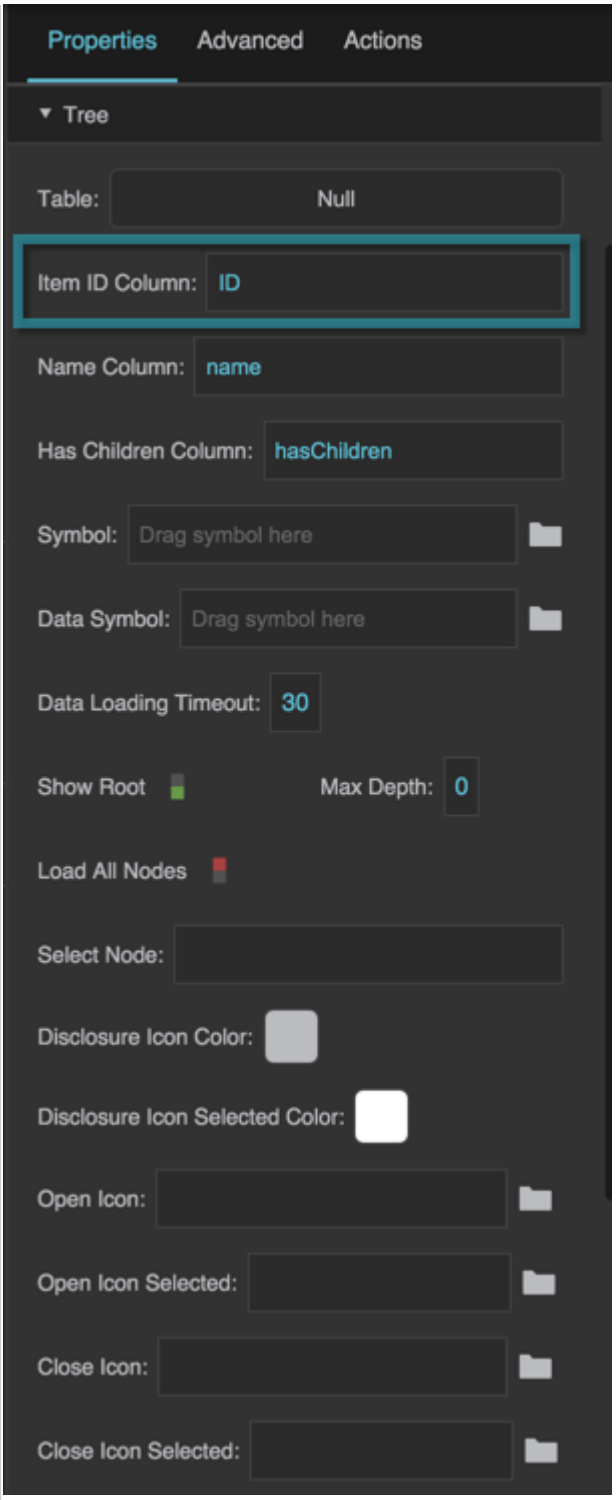
These properties affect the entire tree.

Table (Required)
Defines the table that the tree uses as a data source. This table contains the top level of nodes and references to the next level of nodes, if they exist. This table must include a column that contains item IDs.



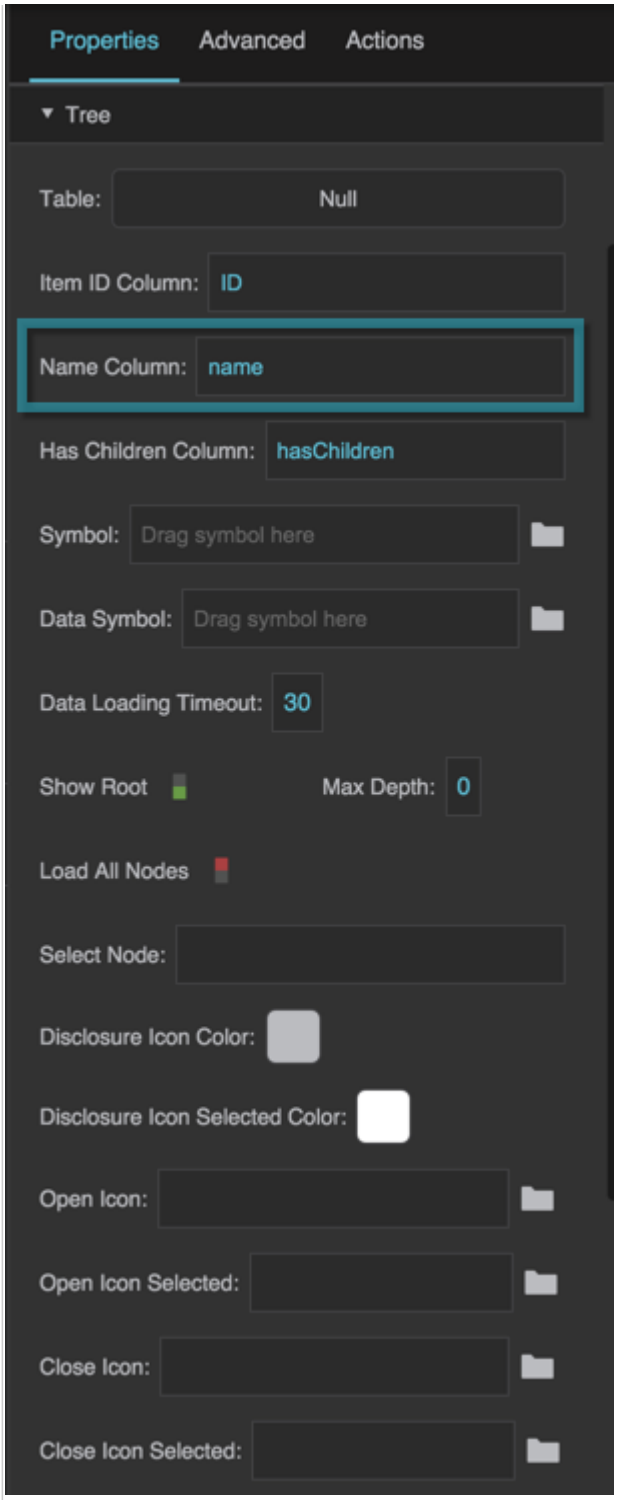
The Table property

Item ID Column (Required)
Contains the name of a column in this tree's source table. This column contains IDs for tree nodes. IDs must be unique within a given table.



The Item ID Column property

Name Column (Recommended)
Contains the name of a column in this tree's source table. This column contains names for tree nodes. If default tree styling is used, the items in this column are used as display text. If a symbol is used for the tree, and the symbol has a default property defined, the items in this column are used for that default property.



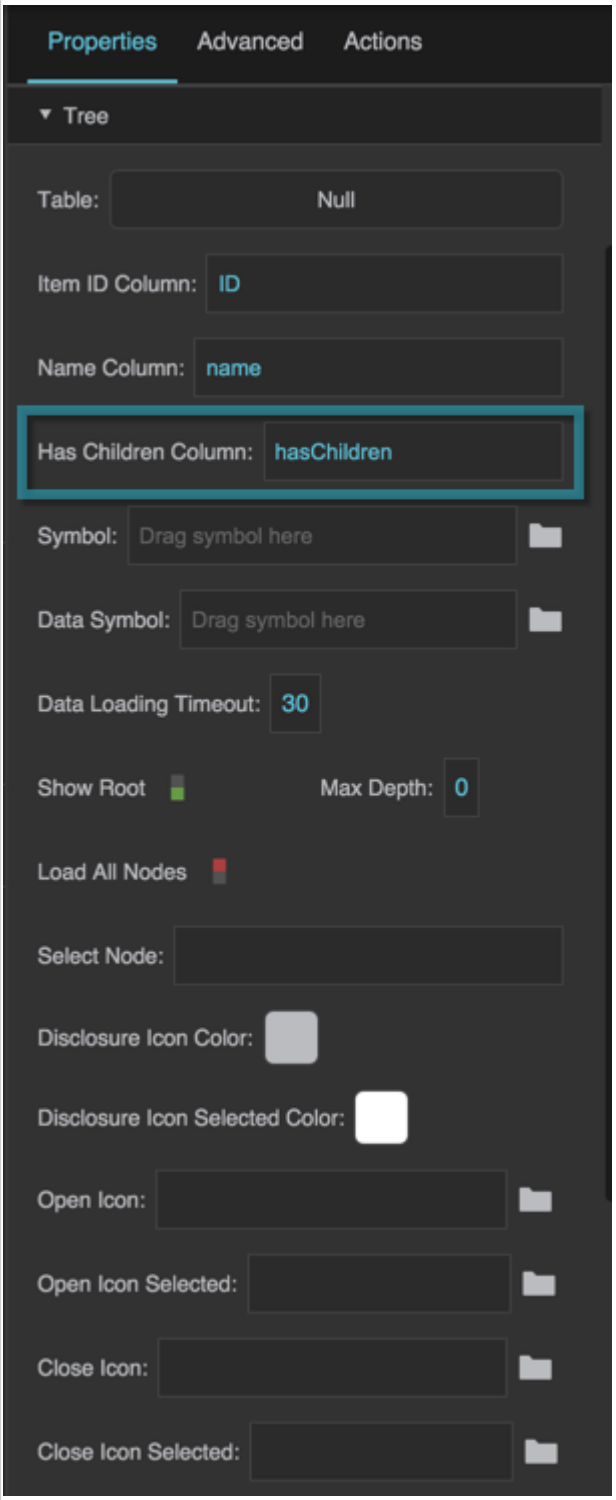
The Name Column property

Has Children Column (Recommended)
Contains the name of a column in this tree's source table. This column indicates whether items in the tree have children and whether the disclosure icon is displayed. All values are considered TRUE except:

- `false`
- `False`

- a null value
- any string that is parsed as `0`

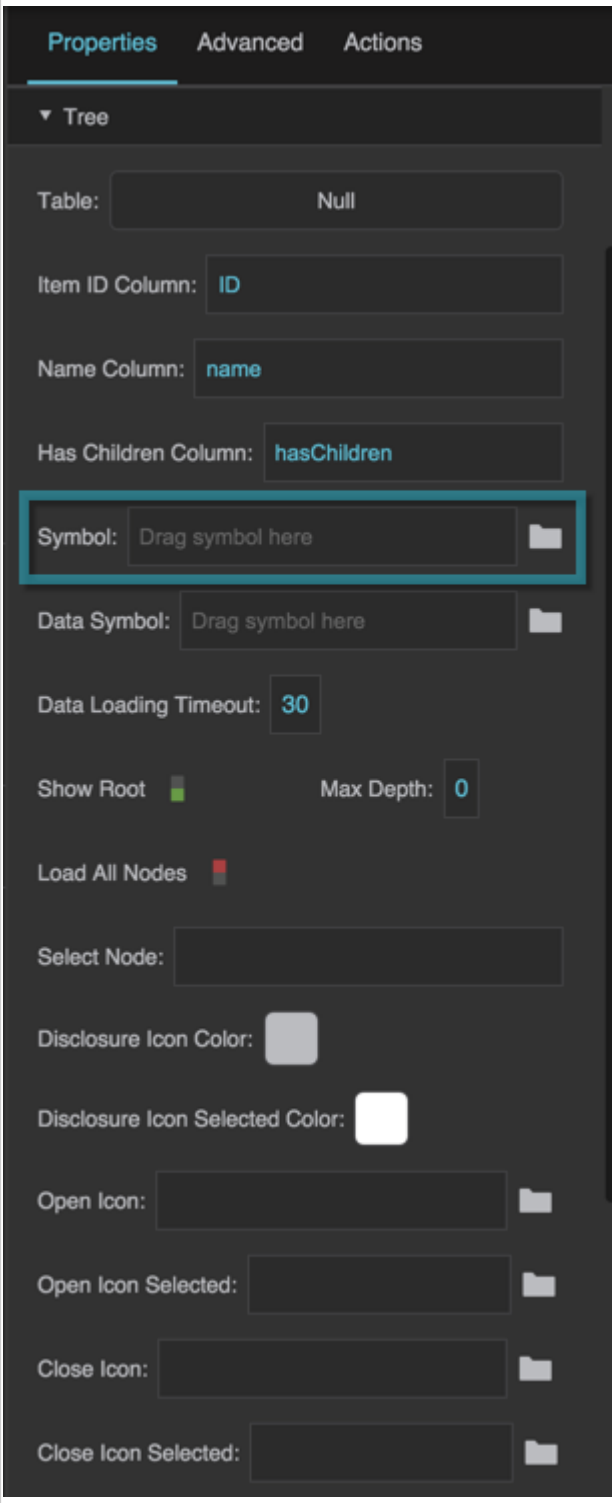
If no column is set, the disclosure icon is always displayed, and an attempt can always be made to load children.



The Has Children Column property

Symbol

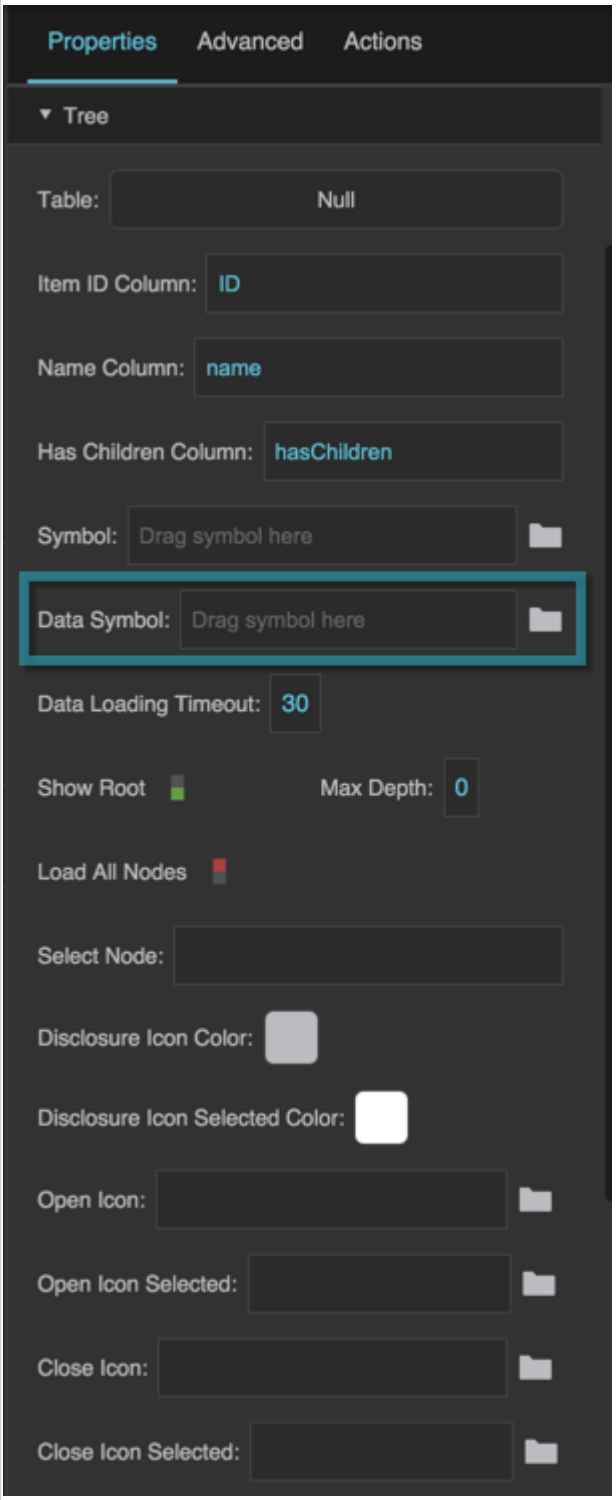
Optionally, this property can be used to specify a symbol to represent the tree nodes, instead of the default tree styling.



The Symbol property

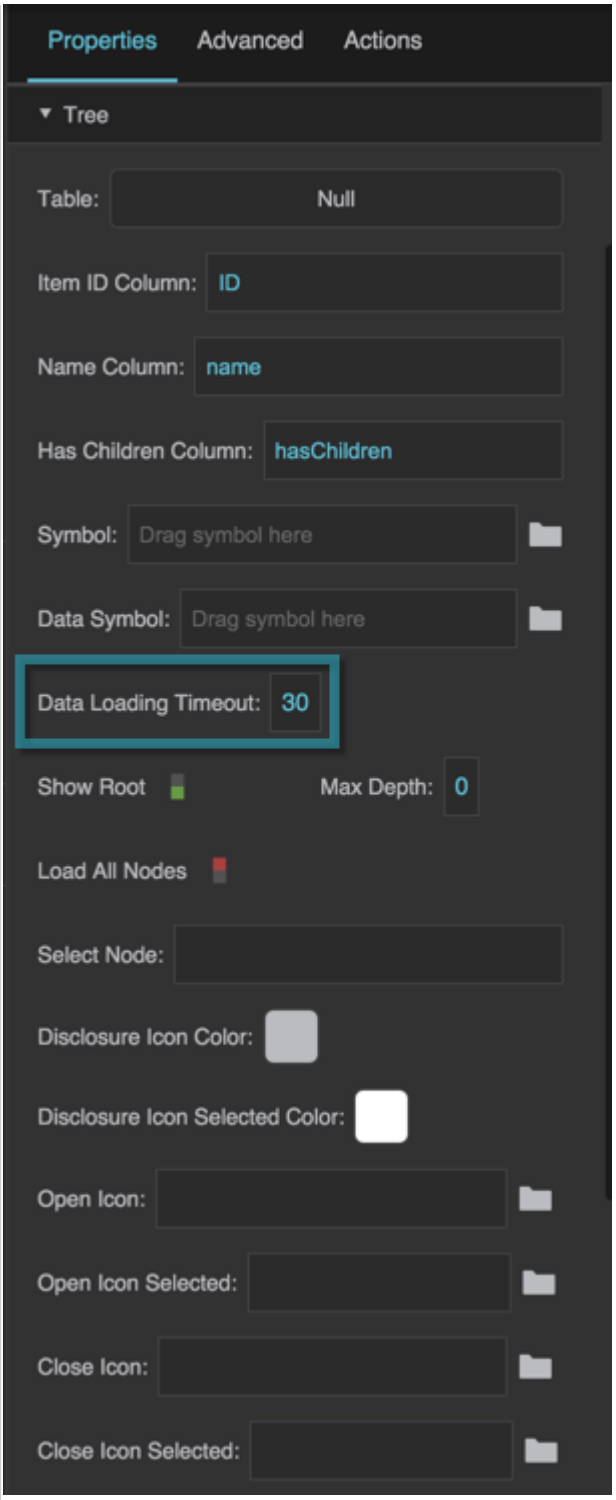
Data Symbol (Required)
Specifies the name of the dataflow symbol to be used by this tree. As described here, the dataflow symbol must include `input` and `output` tabledata parameters. This symbol takes a one-row table and

returns the child table that it refers to, if any.



The Data Symbol property

Data Loading Timeout
Specifies how long, in seconds, DGLux5 will attempt to load a child table before stopping the attempt.



The Data Loading Timeout property

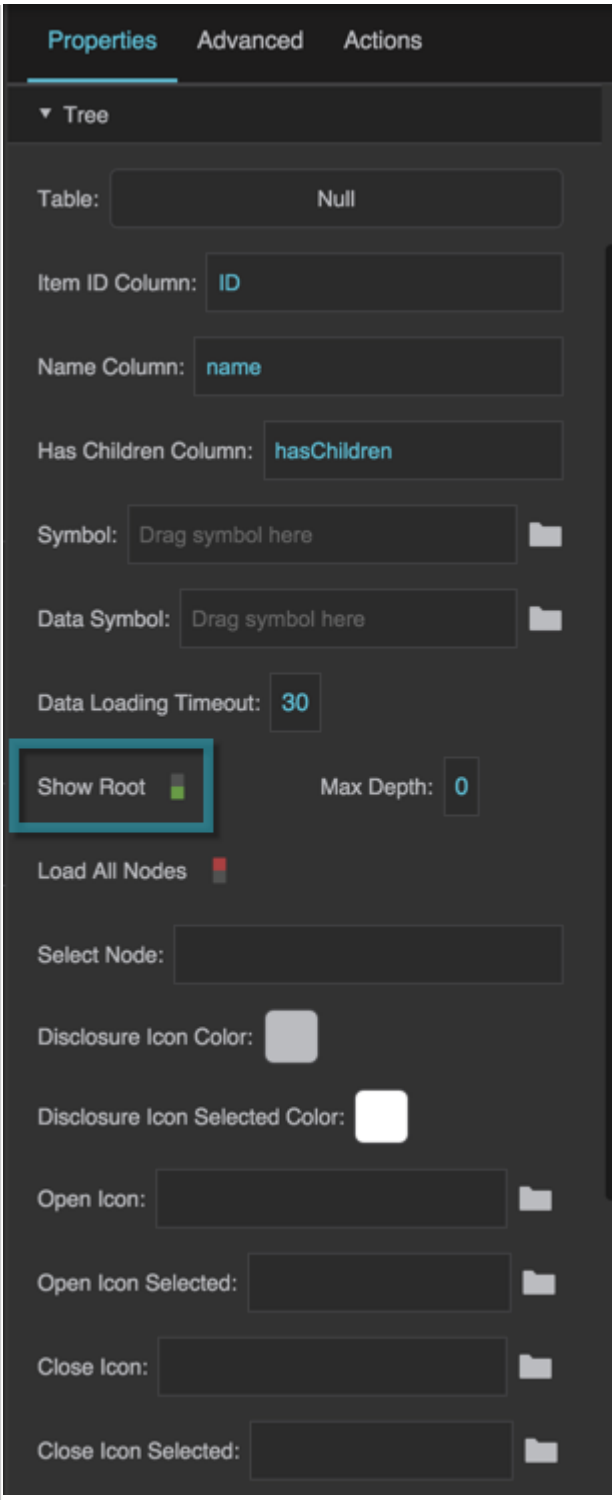
Show Root
Specifies whether the data in the top-level table is displayed in the tree component. This property is TRUE by default.

TRUE
The tree component displays as its top level the nodes named in the table that is used as the **Table**

property.

FALSE
The tree component displays as its top level the *children of the first row of* the table that is used as the **Table** property. The other rows from the top-level table are not displayed.

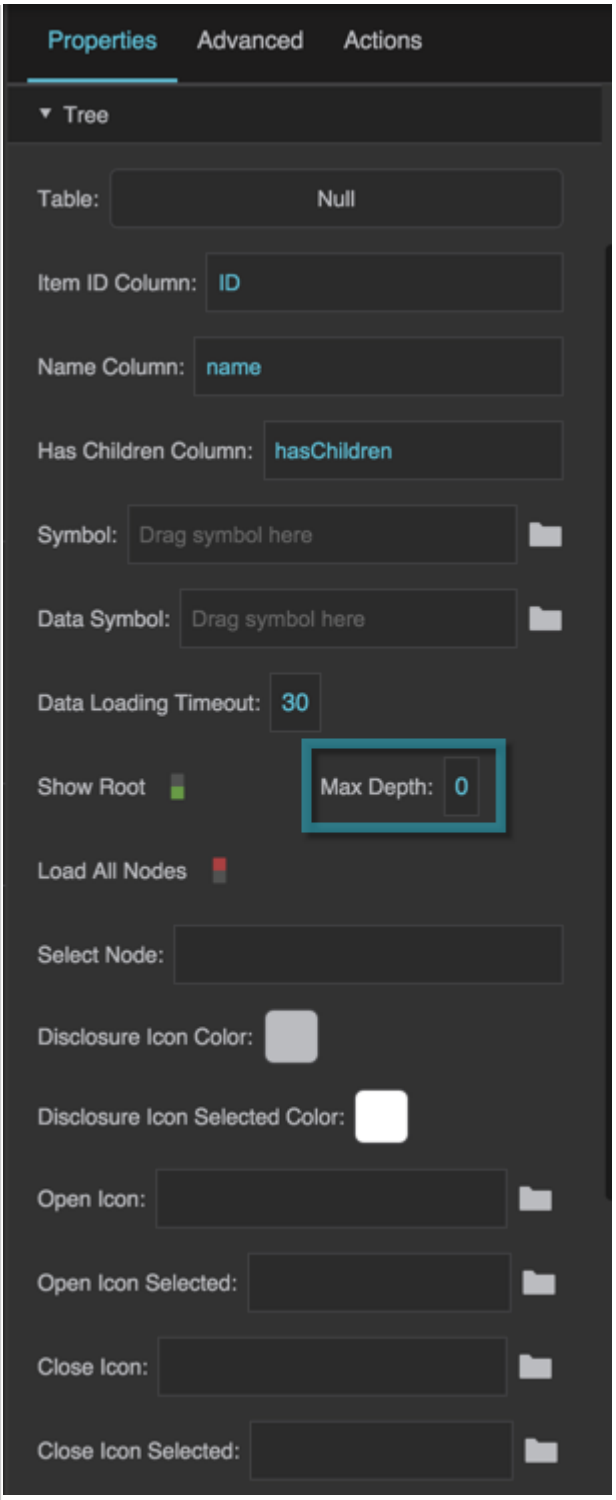
A tree with the Show Root property set to TRUE (left) and FALSE (right)



The Show Root property

Max Depth

An integer that specifies the maximum level of tree descendants that are allowed. A value of 0 allows an unlimited number of levels. A value of 1 allows only one level of nodes and does not allow any descendants to be loaded.

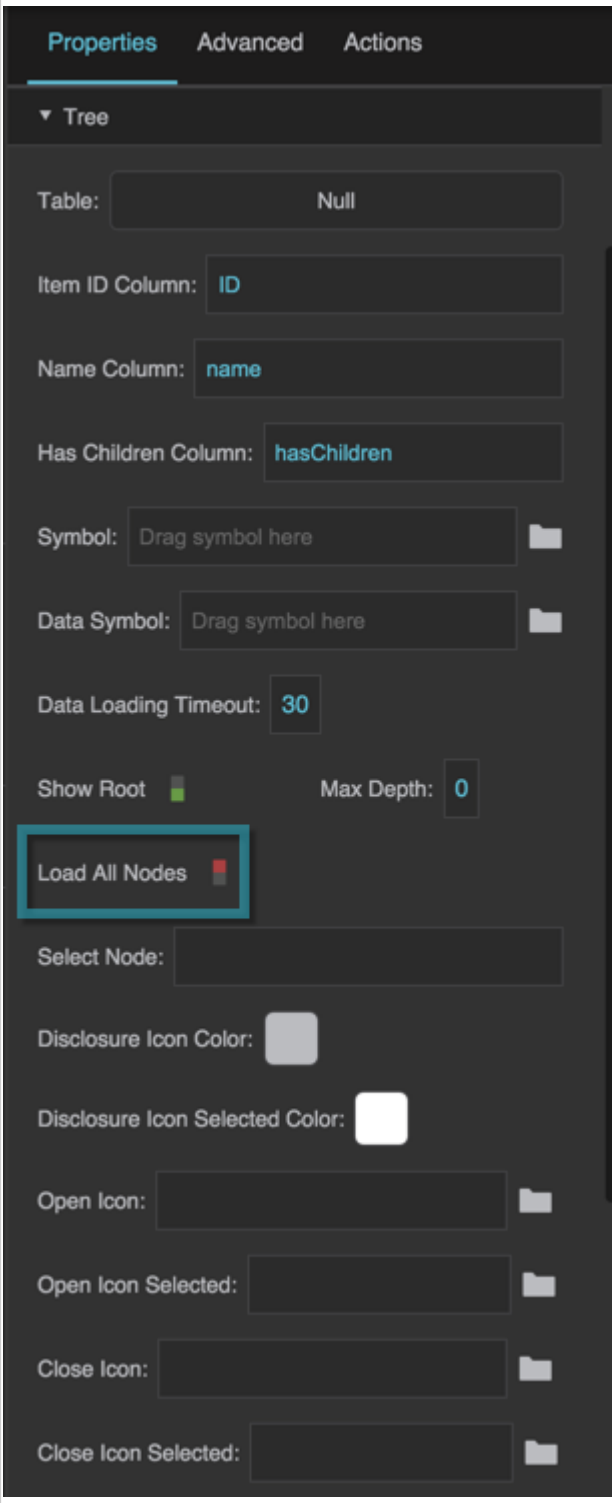


The Max Depth property

Load All Nodes
A boolean value that indicates whether to pre-load children for the tree. This value is FALSE by default.

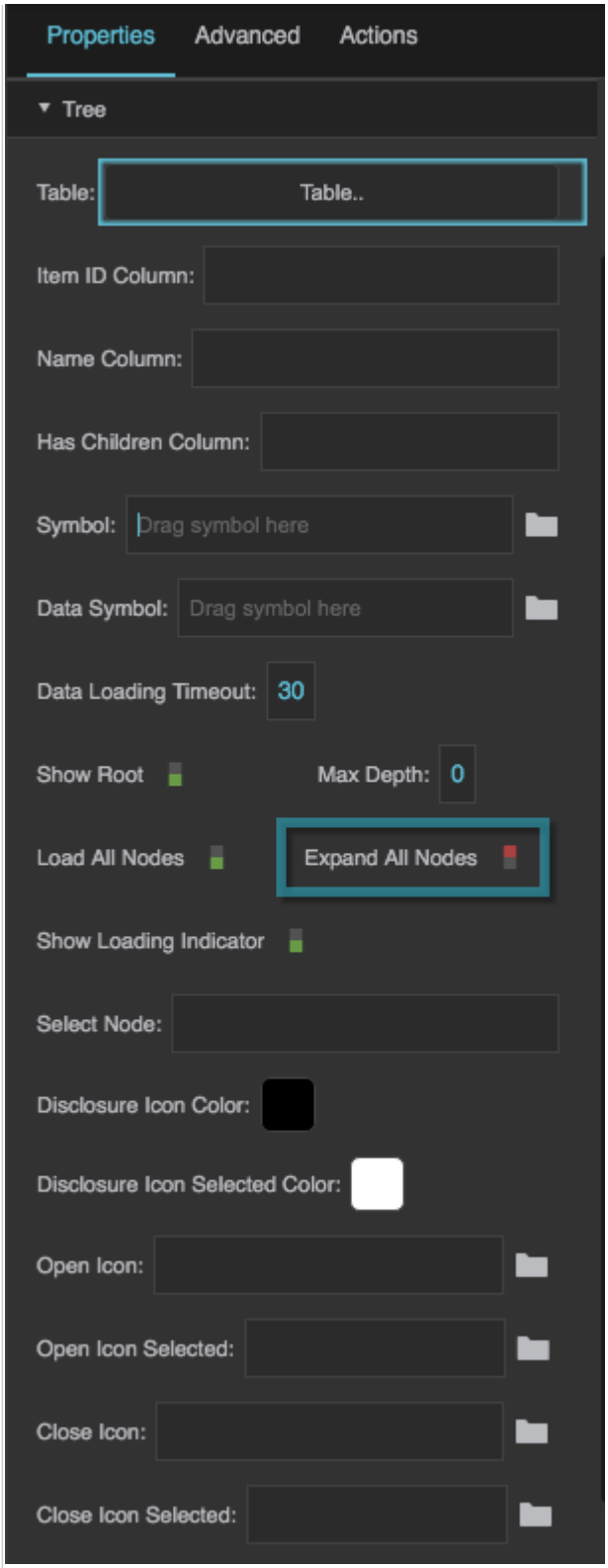
TRUE
All children are loaded when the page is first loaded, when the data table is first set, or when this property becomes TRUE.

FALSE
Children of a node are loaded only when requested, for example when a user clicks the disclosure icon.



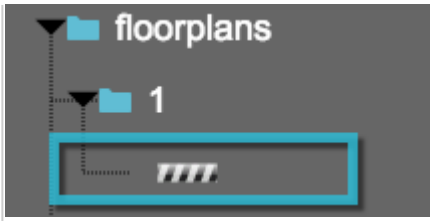
The Load All Nodes property

Expand All Nodes
This property is visible only when **Load All Nodes** is TRUE. It determines whether to expand all nodes after they are loaded.

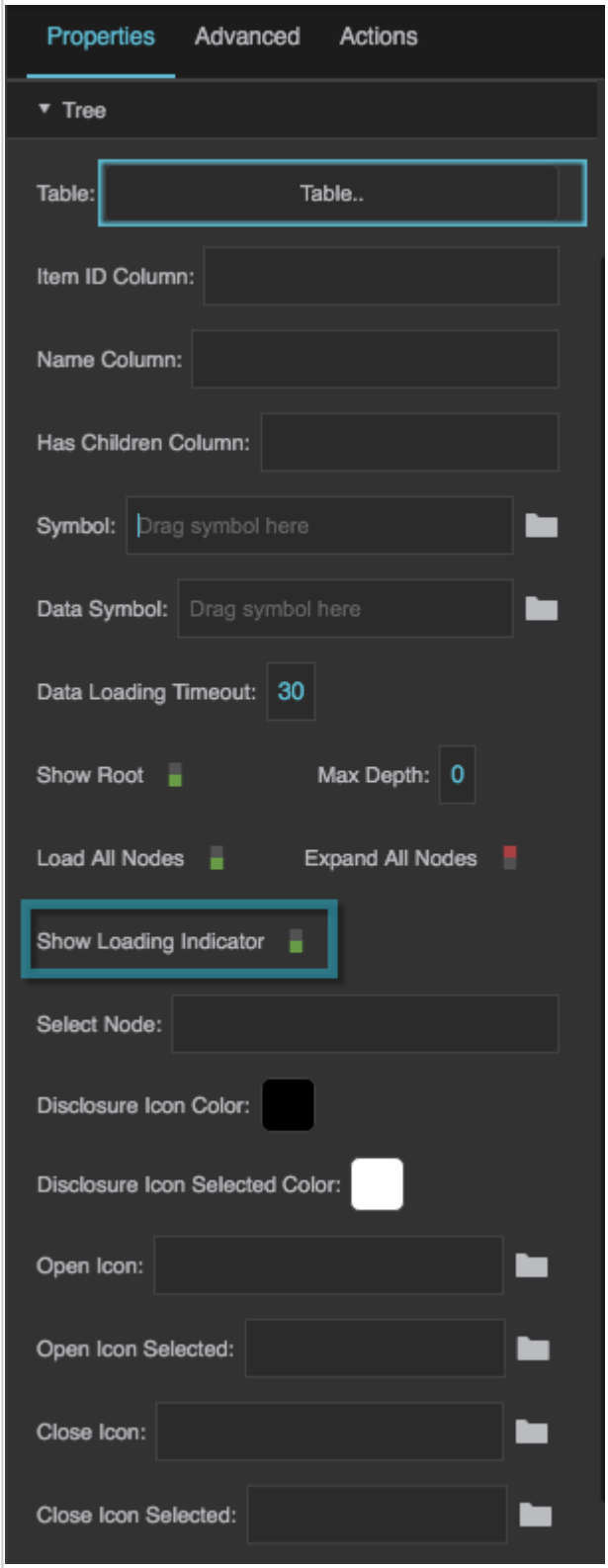


The Expand All Nodes property

Show Loading Indicator
Determines whether to disaply a visual signal when an attempt is being made to load tree nodes.



An example of the loading indicator



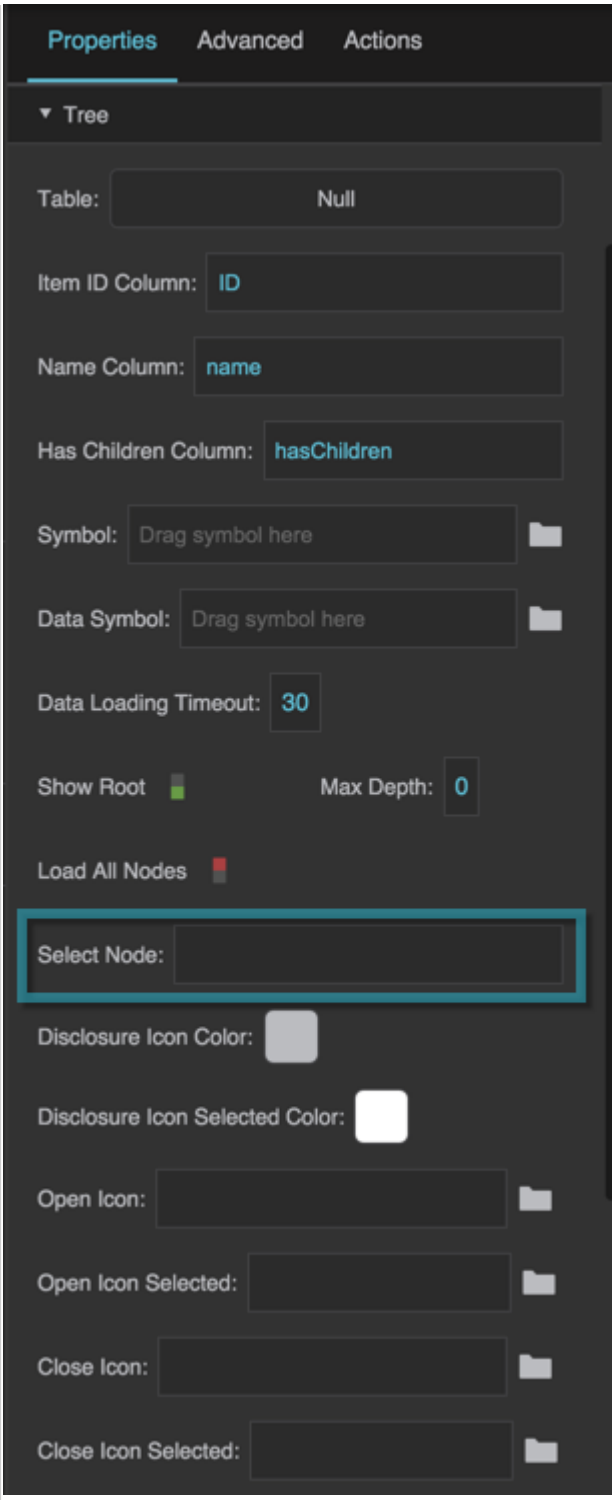
The Show Loading Indicator property

Select Node
A node ID that specifies a loaded node to select, expand, and make visible. Only works if the tree's Selection Behavior is set to a value other than **None**.

If the specified node is not loaded, this property does nothing. You can use this property together with the **Load All Nodes** property and the **onAllNodesLoaded** event property in the Advanced properties, to ensure that all nodes are loaded before this property's value is set.

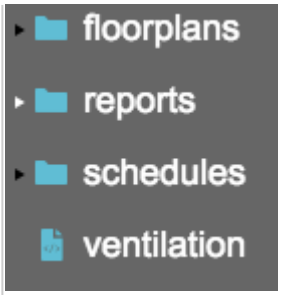
If a node with the specified ID exists in more than one table, then the first node with this ID that is encountered is used.

For selection of more than one node, use the **Selected Items** property in the Selection properties. However, **Selected Items** controls selection only, not which nodes are expanded.

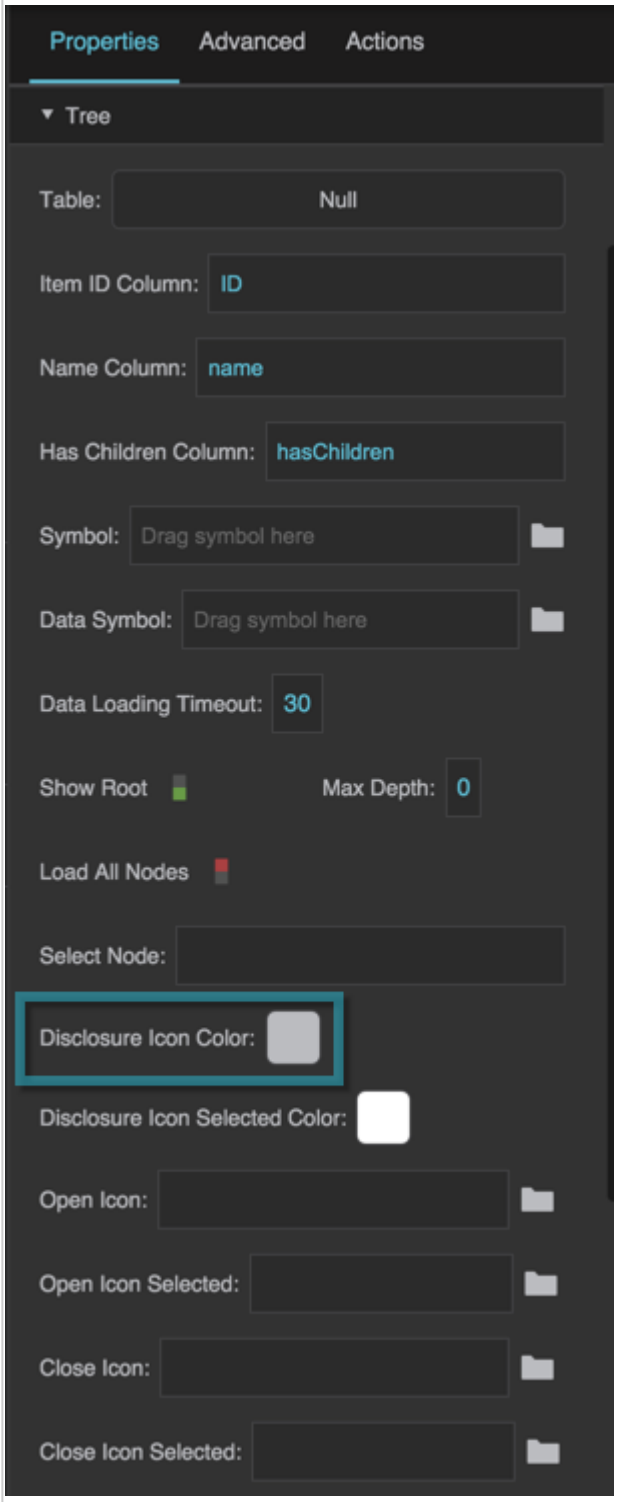


The Select Node property

Disclosure Icon Color
A color to be used for the default "expand" and "collapse," or "open" and "close," icons for a node that has children and is not currently selected.

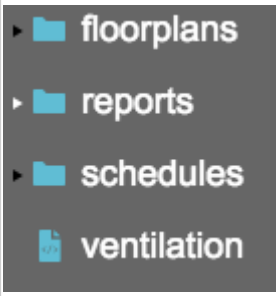


A tree with a black Disclosure Icon Color and a white Disclosure Icon Selected Color

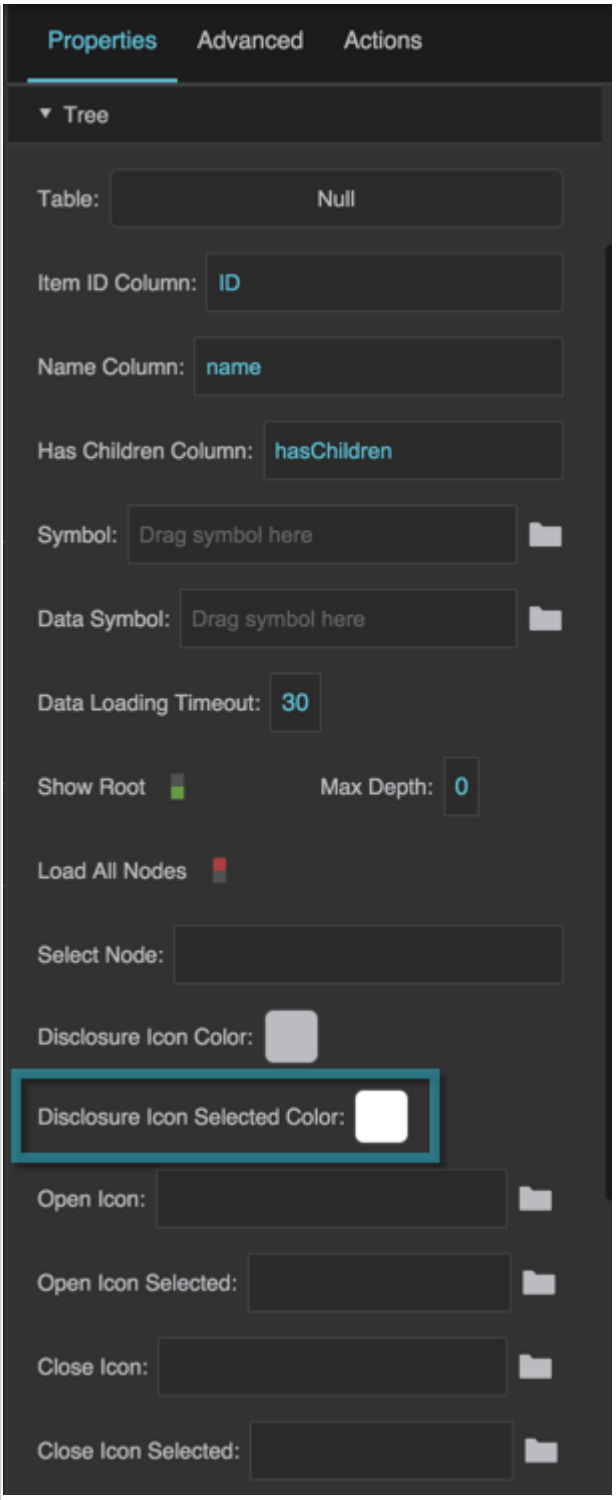


The Disclosure Icon Color property

Disclosure Icon Selected
A color to be used for the default "expand" and "collapse," or "open" and "close," icons for a node that has children and is currently selected. Only works if selection is enabled for this tree.

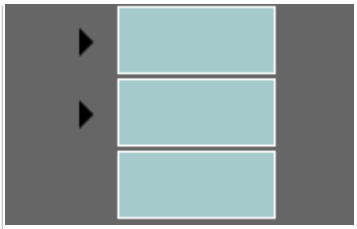


A tree with a black Disclosure Icon Color and a white Disclosure Icon Selected Color

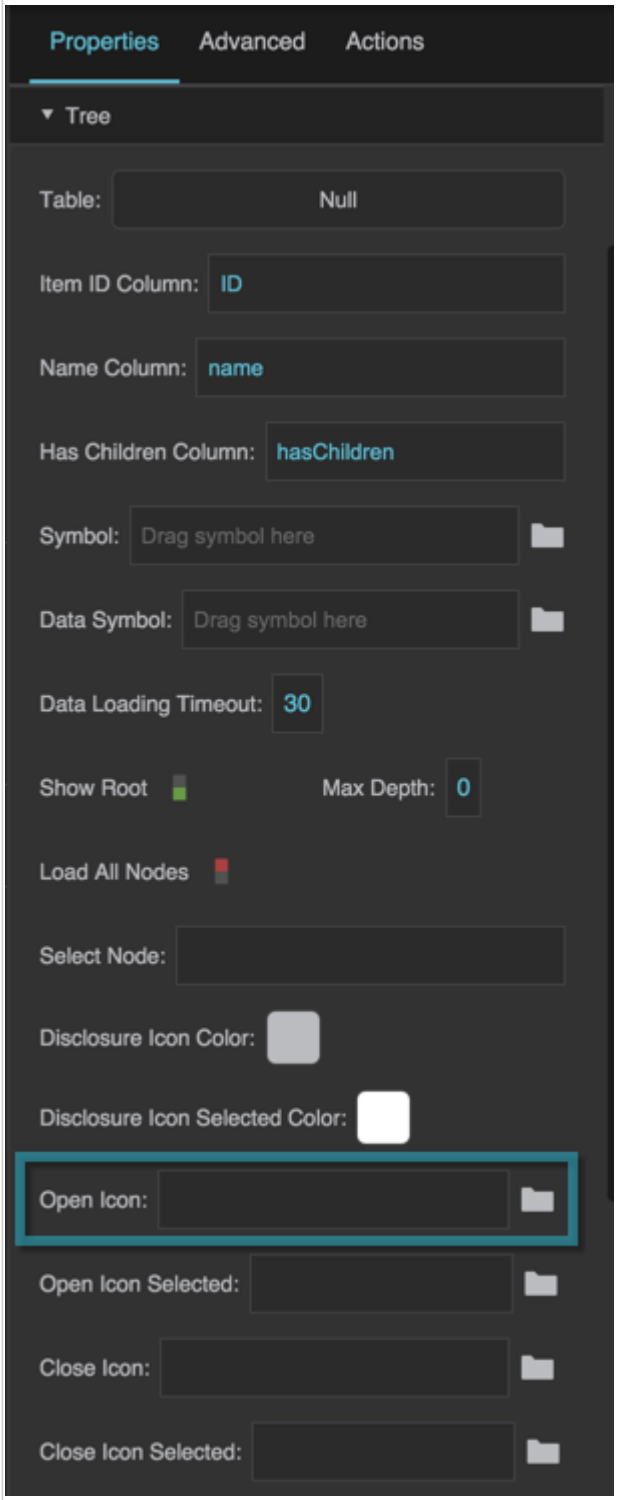


The Disclosure Icon Selected Color property

Open Icon
An image to be used as the "expand," or "open," icon for a node that has children, is currently closed, and is not currently selected.

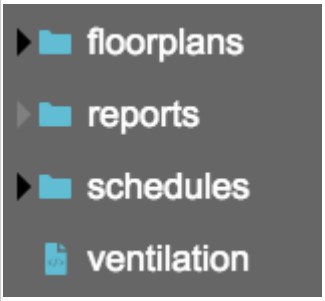


A tree with a triangle SVG used as the Open Icon property

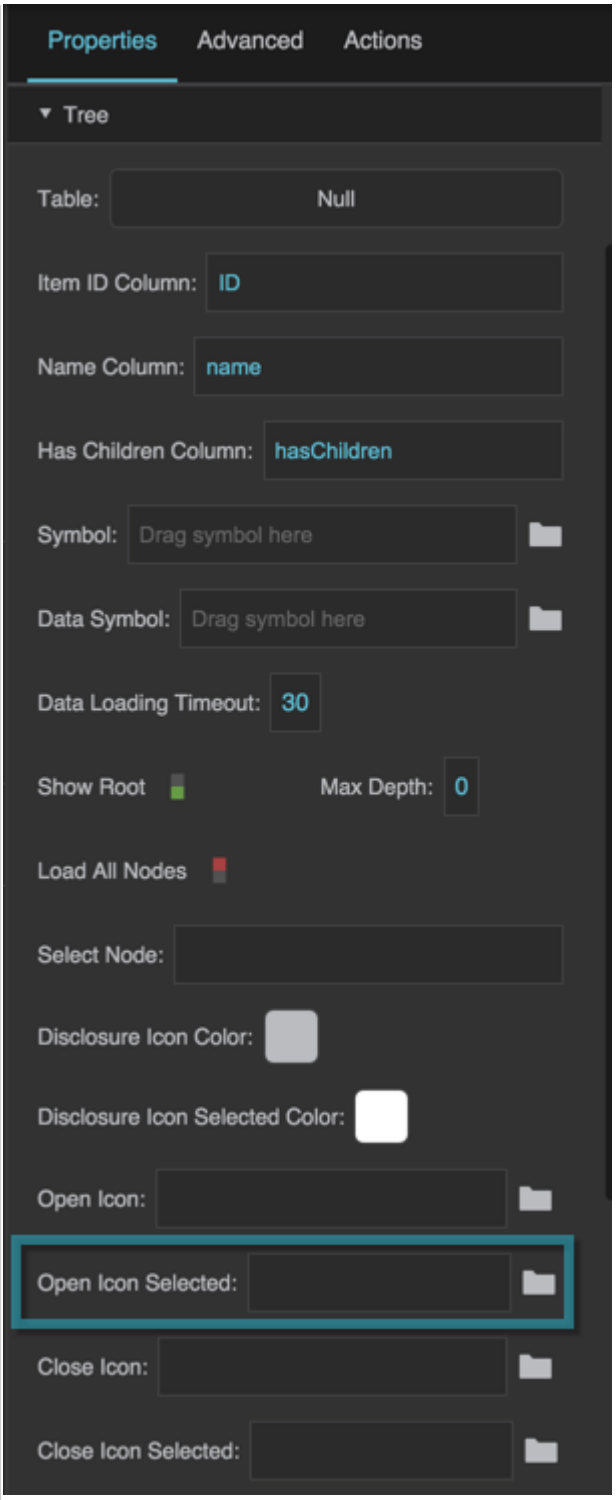


The Open Icon property

Open Icon Selected
An image to be used as the "expand," or "open," icon for a node that has children, is currently closed, and is currently selected. Only works if selection is enabled for this tree.



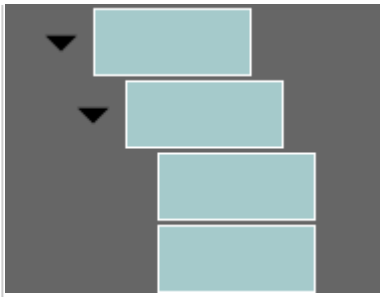
A tree with a gray triangle SVG used as the Open Icon Selected property



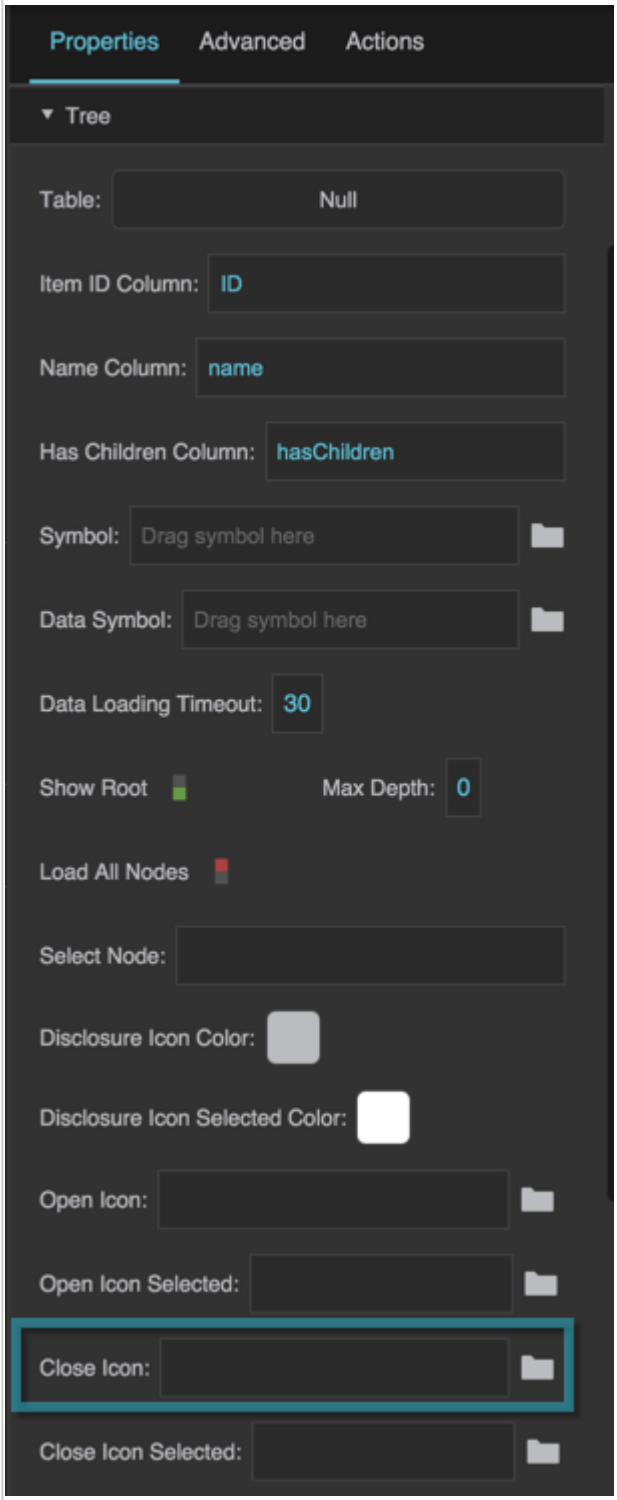
The Open Icon Selected property

Close Icon

An image to be used as the "collapse," or "close," icon for a node that has children, is currently open, and is not currently selected.

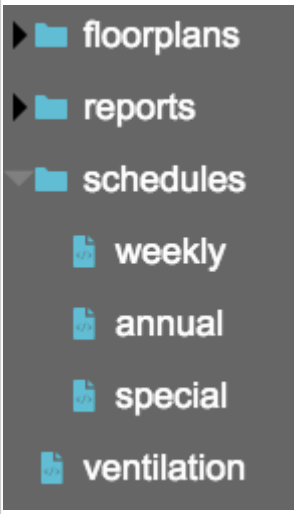


A tree with a black triangle SVG used as the Close Icon property

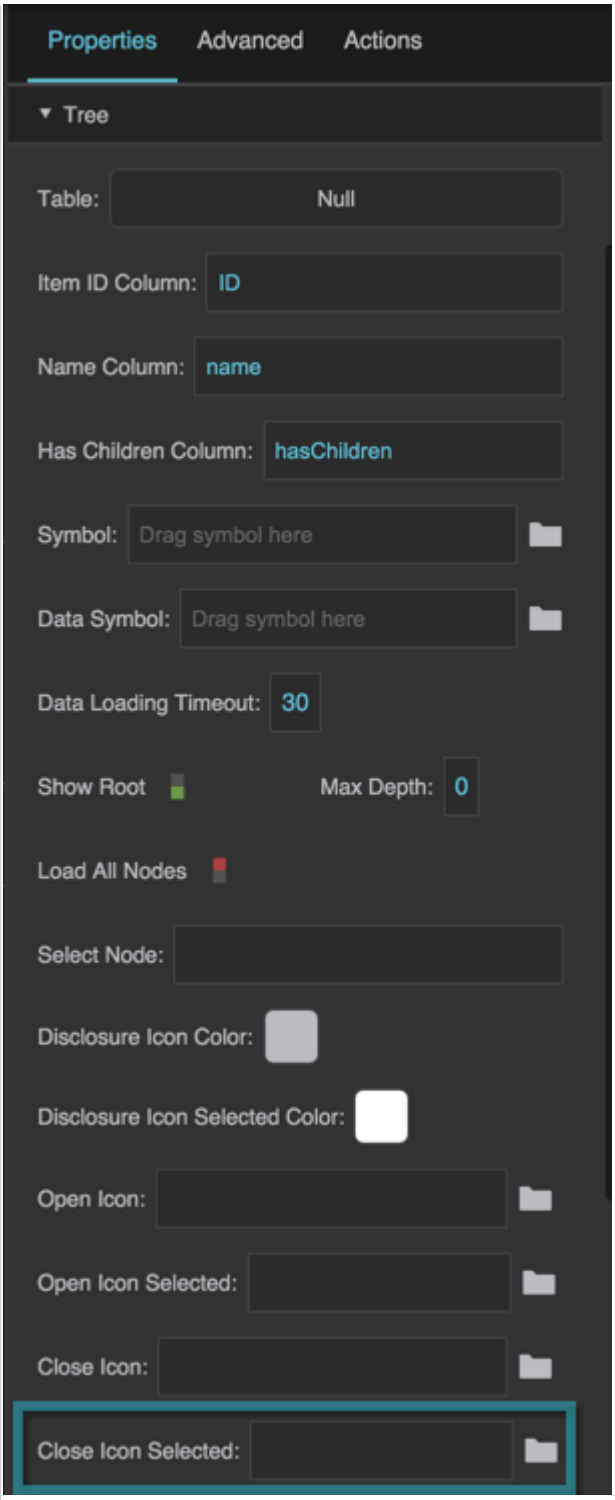


The Close Icon property

Close Icon Selected
An image to be used as the "collapse," or "close," icon for a node that has children, is currently open, and is currently selected. Only works if selection is enabled for this tree.



A tree with a gray triangle SVG used as the Close Icon Selected property



The Close Icon Selected property

Tree Items Properties

These properties affect the visual formatting of the tree.

Tree Items Text and Font Properties

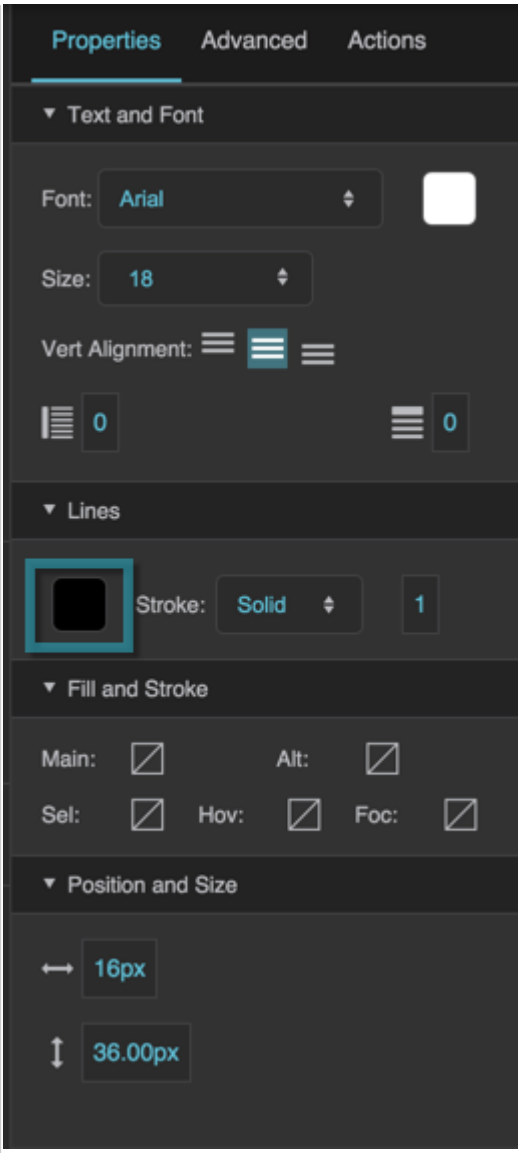
The Text and Font properties for tree items are similar to those for Text components.

Tree Items Lines Properties

The Lines properties for tree items affect the lines that connect tree items.

A tree with the Lines properties set as a black dotted line.

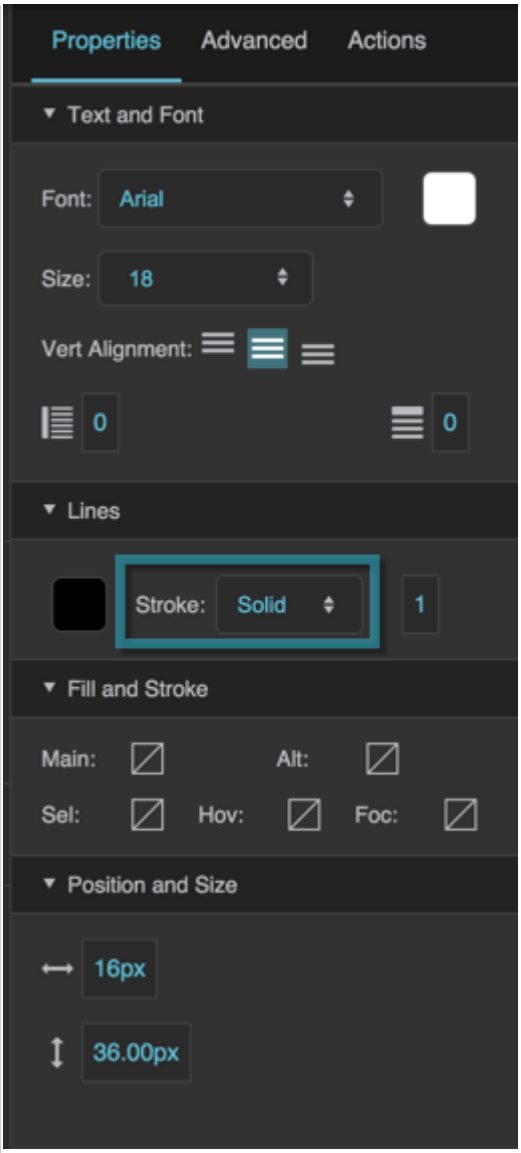
Lines Stroke Color
Defines the color used for lines in the tree.



The Lines Stroke Color property

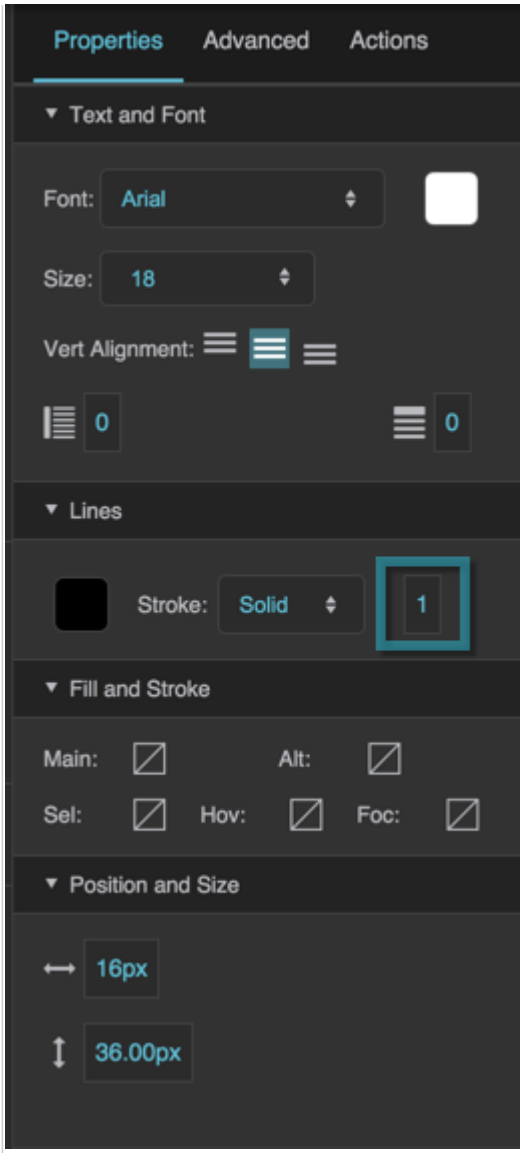
Lines Stroke Style

Defines whether lines in the tree are solid or dotted lines.



The Lines Stroke Style property

Lines Stroke Weight
Defines the stroke weight used for lines in the tree.

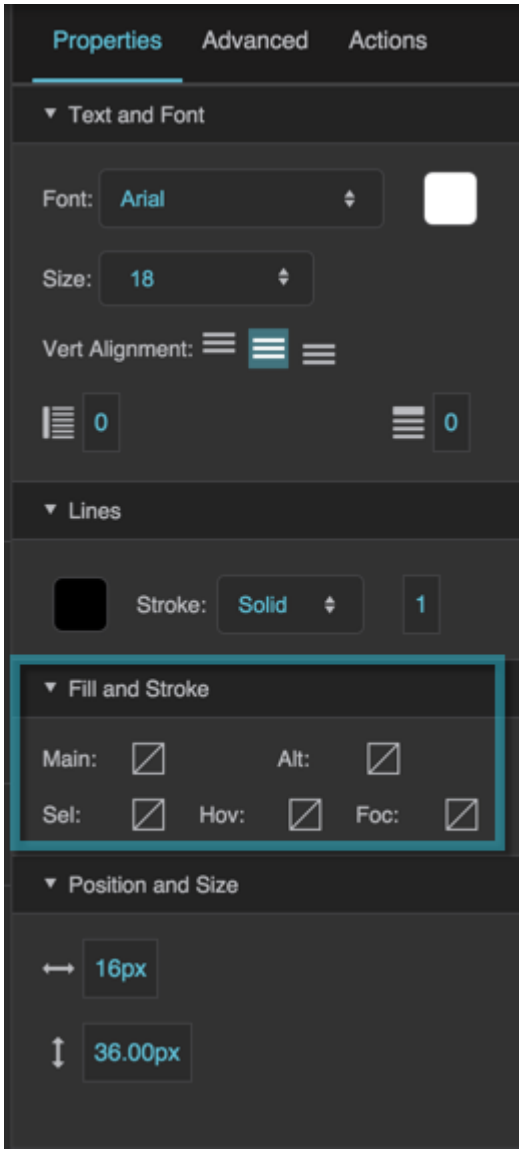


The Lines Stroke Weight property

Tree Items Fill Properties

Tree items fill properties affect the types of rows listed in this table.

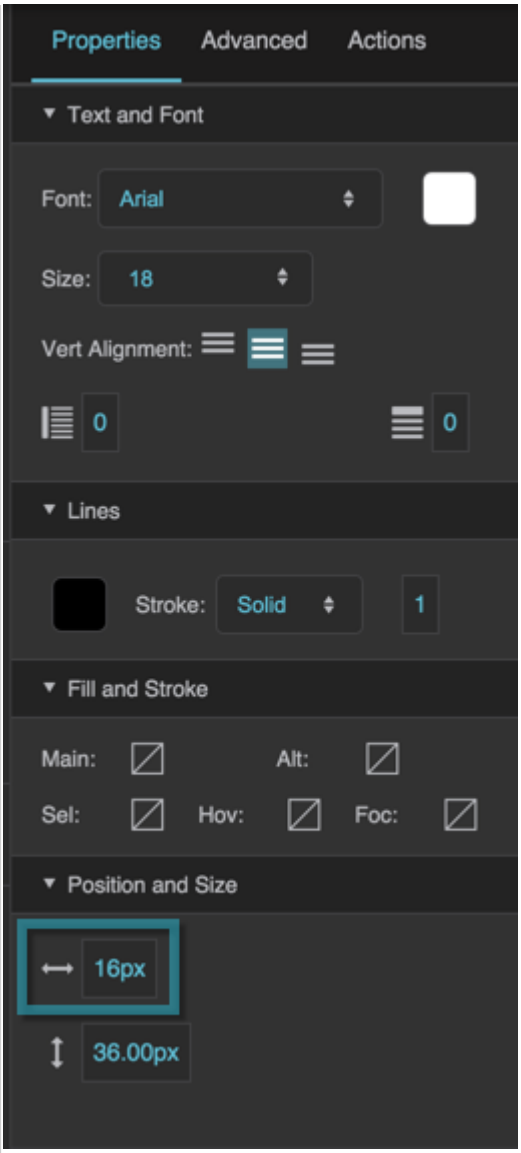
Property	Description
Main Rows	The first group of alternating rows.
Alternate Rows	The second group of alternating rows.
Hovered Row	The row that the user mouses over.
Focused Row	The row that is in focus.
Selected Rows	The selected rows. Only works if selection is enabled for this tree.



The Tree Items Fill properties

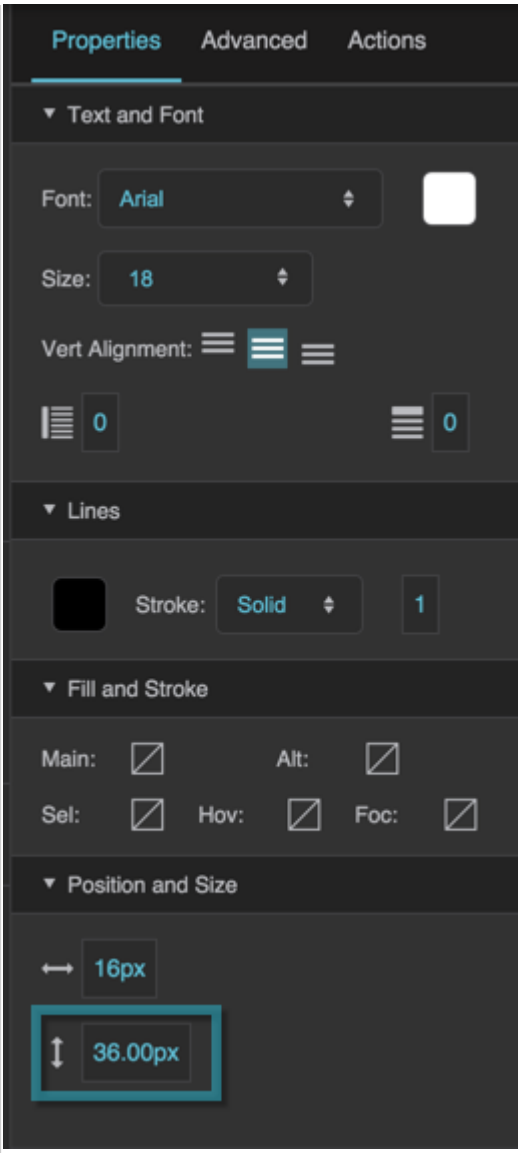
Tree Items Position and Size Properties

Indent
Defines the number of pixels by which each level in the tree is indented. If an image is specified for the **Open Item Icon** and **Close Item Icon** properties, then **Indent** also controls the width of this image.



The Indent property

Item Height
Defines the height of each item in the tree, in pixels.



The Item Height property

2019/07/17 19:17

Previous: Date Range Picker

Next: Tree Grid