# Dynamic Tree Grid Control API

The Dynamic Tree Grid Control is available in Effectivity Services. This section describes the API. The Dynamic Tree Grid control is inherited from the TreeGrid component and has all its public methods, fields, and events.

## Constructor

Table below describes two parameters that the constructor supports.

Constructor-supported parameters

| Name | Type | Description |
|---|---|---|
| dom | Object | Required parameter. It is a DOM element that will be used as a control container. |
| settings | Object | Optional parameter. It contains the following global settings for grid initialization: multiSelect – gives you the ability to select multiple rows in the grid. The default value is false. editable – determines whether a grid can be edited or not. The default value is true. sortable – determines whether grid columns can be sorted. The default value is true. draggableColumns – determines whether grid columns can be moved. The default value is true. resizable – determines whether columns can be resized. The default value is true. If you do not specify the settings parameter, its default value will be used for grid initialization. |

## Public fields

DynamicTreeGrid does not have its own public fields. It uses the base TreeGrid component public fields.

## Events

You can add and remove Events using the “on” and “off” public methods.

Each event listener callback accepts a single parameter: an event object of the type “CustomEvent” containing a “detail” property with specific event information.

**Important**

Column and row indexes are zero-based. Row indexing starts from the first row displayed in a grid and ends at the bottom row of the entire grid. It means that rows whose parents are collapsed/invisible are not counted. The column indexing starts from

the first column on the left. All invisible rows and columns have an index of -1.

**addRow**

Event fired after each row is added to the grid when adding rows using the “addRows” or “loadData” methods.

Property details for addRow

| Name | Type | Description |
|---|---|---|
| rowId | String | ID of the added row. |
| parentId | String | ID of the parent row. Null if added row is root. |

**addRows**

Event fired after all rows are added using the “addRows” or “loadData” methods.

Property details for addRows

| Name | Type | Description |
|---|---|---|
| rowIds | Array | Added row IDs. |
| parentId | String | ID of the parent row. The ID is Null if the added row is root. |

**removeRow**

Event fired after deleting a row from the grid using the “removeRow” method.

Property details for removeRow

| Name | Type | Description |
|---|---|---|
| rowIds | Array | IDs of the deleted rows. Array contains ID of the deleted row and IDs of all descendant rows. |
| parentId | String | ID of the parent row. The ID is Null if the deleted row is a root row. |

**removeAllRows**

Event fired after deleting all rows from the grid using the “removeAllRows” method.

Property details for removeRows

| Name | Type | Description |
| --- | --- | --- |
| rowIds | Array | IDs of the deleted rows. |

#### addColumn

Event fired after adding a new column using the “addColumn” or “loadData” methods.

Property details for addColumn

| Name | Type | Description |
| --- | --- | --- |
| columnName | String | Column name. |

#### removeColumn

Event fired after deleting a column using the “removeColumn” method.

Property details for removeColumn

| Name | Type | Description |
| --- | --- | --- |
| columnName | String | Column name. |

## Common objects description

This section describes common objects frequently used in the DynamicTreeGrid public methods.

#### Metadata objects

A Metadata object is an object that contains additional information about a cell that is required for the grid editors and formatters to render the cell properly.

A Metadata object can have the following properties:

- formatter – optional property.
  The formatter name used to display cell content. If you do not specify a “formatter” property, the formatter type is determined by the grid based on the cell value. E.g., if the cell value is a string the ‘text’ formatter will be used, and if the cell value is boolean the ‘boolean’ formatter will be used.
- editor – optional property.
  The name of the editor used to display cell content when the cell is in the edit state. If you do not specify an “editor” property, the editor type will be determined by the grid.

- Other properties that can be used in the formatter and editor functions from the “metadata” parameter.

Example:

```
{
formatter: ‘select’,
editor: ‘select’,
options:         [
{label: 'Red', value: 'id1'},
{label: 'Green', value: 'id2'},
{label: 'Blue', value: 'id3'}
]
}
```

**Column settings object**

The Column settings object contains additional settings applicable to the corresponding column. If a grid is not editable/resizable/sortable, then you cannot edit/resize/sort columns. However, if the grid is editable/resizable/sortable you can adjust these columns. The Column setting does not change when the corresponding grid setting is changed. The Column settings object can have the following optional properties:

- **resizable** enables you to decide whether to resize a column. The default value is obtained from grid settings when adding a new column.
- **editable** enables you to decide whether to edit cells in this column. The default value is obtained from grid settings when adding a new column. This setting value will be used as a default value for the cell’s editability setting when adding new rows. You can also use the “setCellEditability” and “getCellEditability” methods to set/get cell editability.
- **sortable** enables you to decide whether to sort by this column. The default value is obtained from the grid settings when adding a new column.
- **visible** enables you to decide if the column should be visible or not. The Default value is true when adding a new column.

Example:

```
{
resizable: false,
sortable: true,
editable: false,
visible: true
}
```

**Column object**

The Column object is used in the “loadData” and “addColumn” methods. It can contain the following properties:

- name – required property. Column name.
- label – optional property. The column name is used as the label if the “label” property is not set.
- width – optional property. The width is calculated automatically based on the column label if the “width” property is not set.
- metadata – optional property. It contains a metadata object for all cells in this column. Column metadata is used only for cells without their own metadata.
- settings – optional property containing settings that are applied to the corresponding column.

Example:

```
{
label: ‘Column 1’,
width: 100,
name: ‘property1,
metadata: {
formatter: ‘boolean’
},
settings: {
resizable: false,
editable: false
}
}
```

**Row object**

The Row object is used in the “loadData” and “addRows” methods. It can contain different properties. If the row object property name matches the column name, then its value is used as a cell value. The Row object property is updated automatically when the corresponding grid cell value is changed. Example:

```
{
property1: ‘value 1’,
property2: ‘value 2’,
someOtherProperties: ‘additional property’
}
```

## Public methods

### loadData

Initializes the grid with specified rows and columns. If the “columns” parameter is not specified, the columns will be obtained automatically from the “rows” parameter. Each unique property of the row object will represent a column with the same name. If both the “rows” and “columns” parameters are not specified, the method removes all existing rows and columns.

> **Important**
>
> The grid automatically updates the corresponding properties of the provided row object when the cell value is changed.

Optional Input parameters for loadData

| Name | Type | Description |
|---|---|---|
| rows | Array | An array containing the row objects described in section .If the parameter is not an array, the rows are not added. |
| columns | Array | An array containing column objects described in section .The Array’s element position determines the column order. |

***Return value*** Array – an array that contains the IDs of the added rows. Example of the “rows” array parameter:

```
[
{
property1: ‘row 1’,
property2: ‘value2’,
someOtherProperties: ‘additional property’
},
{
property1: ‘row 2’
}
]
```

Example of the “columns” array parameter:

```
 [
{
label: ‘Column 1’,
width: 55,
name: ‘property1,
metadata: {
formatter: ‘select’,
editor: ‘select’,
options:         [
{label: 'Red', value: 'id1'},
{label: 'Green', value: 'id2'},
{label: 'Blue', value: 'id3'}
]
}
},
{
name: ‘property2’,
settings: {
resizable: false,
sortable: true,
editable: false
}
}
]
```

**obtainRowID**

obtainRowID is an overridable handler which should return a unique row id for the given row object when adding new rows using the “loadData” or “addRows” methods. It returns null by default. The Unique row id will be generated automatically if the handler returns a falsy value, e.g., false, empty string, null or undefined. If the handler returns a truthy value which is not a string or grid that already contains a row with the same id, an error will be thrown in the “loadData” or “addRows” methods. Input parameters for obtainRowID.

| Name | Type | Description |
|---|---|---|
| rowInfo | Object | Row object which is used to add a new row. |
| parentId | String | Parent row ID. The ID will be Null if the row is added as root. |

***Return value* String—row ID.** Example:

```
dynamicTreeGrid.obtainRowId = function(rowObj, parentId) {
return rowObj.id_property;
};
```

**addRows**

Adds rows to the specified position in the grid.

Input parameters for addRows.

| Name | Type | Description |
| --- | --- | --- |
| rows | Array | Required parameter. Arrays containing row objects are described in section.<br>If the parameter is not an array, the rows will not be added. |
| parentId | String | Optional parameter. ID of the parent row.<br>If the “parentId” parameter is null or undefined, rows will be added as roots. |
| position | Number | Optional parameter. It is a zero-based index that is used to insert new rows into the child row IDs array of the parent row (or roots array if parentId is not specified). The Index can be greater than or equal to 0 and less than or equal to the children array length.<br>If the “position” parameter is not specified, new child rows will be appended to the end of the children array.<br>The getChildRowIds method can be used to find out the row index in the children array. |
| doRender | Boolean | Optional parameter. Renders grid after adding rows. The default value is true. |

***Return value***

**Array** - array which contains IDs of the added rows.

**removeRow**

Removes the row with the specified ID from the grid and grid rows collection. Input parameters for removeRows

| Name | Type | Description |
| --- | --- | --- |
| rowId | String | Row ID. |

**removeAllRows**

Removes all rows from the grid and the grid rows collection.

**selectRow**

Selects the row with the specified ID. Otherwise, the specified row becomes selected, and all previously selected rows stay selected.

Input parameters for selectRow

| Name | Type | Description |
|---|---|---|
| rowId | String | Row ID. |

**deselectRow**

Cancels the selection of the specified row.

Input parameters for deselectRow

| Name | Type | Description |
|---|---|---|
| rowId | String | Row ID. |

**addColumn**

Adds a column to the grid.

Input parameters for addColumn

| Name | Type | Description |
|---|---|---|
| column | Object | Required parameter. The Column object is described in the section . |
| position | Number | Optional parameter. Zero based column index. If its position is not specified, the column will be appended after existing columns. |

**removeColumn**

Removes the specified column from the grid and the grid columns collection.

Input parameters for removeColumn

| Name | Type | Description |
|---|---|---|
| name | String | Column name. |

**getChildRowIds**

Returns an Array containing the child row IDs for the specified parent row. This method returns the IDs of the root rows if the “parentId” parameter is null or undefined.

Input parameters for getChildRowIds

| Name | Type | Description |
|---|---|---|
| parentId | String | Optional parameter. ID of the parent row. |

***Return value*** **Array**—array with child row IDs or an empty array if there are no children.

**getRowUserData**

Gets user data stored by the specified row and key.

Input parameters for getRowUserData

| Name | Type | Description |
|---|---|---|
| rowId | String | Row ID. |
| key | String | Key for storing user data. |

***Return value***

**Anything** - user data stored by the specified row and key.

Example:

```
let itemtype = grid.getRowUserData(rowId, ‘itemtype’);
```

**setRowUserData**

Sets user data for the specified row and key.

Input parameters for setRowUserData

| Name | Type | Description |
|---|---|---|
| rowId | String | Row ID |
| key | String | Key for storing user data |
| value | Anything | Value |

Example:

```
grid.setRowUserData(rowId, ‘itemtype’, ‘Part’);
```

**setCellEditability**

Sets cell edit availability.

Input parameters for setCellEditability

| Name | Type | Description |
|---|---|---|
| rowId | String | Required parameter. Row ID. |
| columnName | String | Required parameter. Column name. |
| editable | Boolean | Optional parameter. Can a cell be edited or not? Default value is true. |

**getCellEditability**

Returns `true` if the specified cell is editable, otherwise it returns `false` .

Input parameters for getCellEditability

| Name | Type | Description |
|---|---|---|
| rowId | String | Row ID. |
| columnName | String | Column name. |

***Return value*** **Boolean** - is cell editable or not.

**setCellValue**

Sets cell value and updates the corresponding property in the user row object.

Input parameters for setCellValue

| Name | Type | Description |
|---|---|---|
| rowId | String | Row ID |
| columnName | String | Column name |
| value | Anything | Cell value |

**getCellValue**

Gets the cell value.

Input parameters for getCellValue

| Name | Type | Description |
| --- | --- | --- |
| rowId | String | Row ID |
| columnName | String | Column name |

***Return value*** **Anything** - cell value.

#### getColumnCount

Gets a count of all columns (visible and invisible).

***Return value*** **Number**

#### getVisibleColumnCount

Gets a count of visible columns.

***Return value*** **Number**

#### getColumnIndex

Gets a column index by the column name. Returns -1 if the column is not found or is invisible. Input parameters for getColumnIndex

| Name | Type | Description |
| --- | --- | --- |
| columnName | String | Column name. |

***Return value*** **Number** - column index

#### getColumnName

Gets the column name using the column index.

Input parameters for getColumnName

| Name | Type | Description |
| --- | --- | --- |
| columnIndex | Number | Column index |

***Return value* String** - column name

**getColumnOrder**

Gets Array with the names of visible columns in the same order as they are displayed in the grid.

***Return value* Array**

**getAllColumnNames**

Gets Array with the names of all columns in the order they are added to the grid.

***Return value* Array**

**getParentId**

Gets the ID of the parent row for the specified row. Returns null if the specified row is root and has no parent.

Input parameters for getParentId

| Name | Type | Description |
|---|---|---|
| rowId | String | Row ID. |

***Return value* String**—parent row ID.

**containsRow**

Returns true if the grid has a row with the specified ID in the rows collection, otherwise it returns false.

Input parameters for containsRow

| Name | Type | Description |
|---|---|---|
| rowId | String | Row ID. |

***Return value***
**Boolean**

**containsColumn**

Returns `true` if the grid has a column with the specified name in the columns collection, otherwise it returns `false` .

Input parameters for containsColumn

| Name | Type | Description |
| --- | --- | --- |
| columnName | String | Column name. |

***Return value***

**Boolean**

#### expandAll

Expands all rows with their descendants.

#### collapseAll

Collapses all rows with their descendants.

#### isRowExpanded

Returns `true` if the specified row is expanded, otherwise it returns `false` .

Input parameters for isRowExpanded

| Name | Type | Description |
| --- | --- | --- |
| rowId | String | Row ID. |

***Return value***

**Boolean**

#### isRowVisible

Returns `true` if the specified row is visible, otherwise it returns `false` .

Input parameters for isRowVisible

| Name | Type | Description |
| --- | --- | --- |
| rowId | String | Row ID. |

***Return value***

**Boolean**

**getVisibleRowCount**

Gets the number of visible rows that can be displayed in the grid.

> **Important**
>
> The number of visible rows is not the total rows count. For example, if the grid has only one root row with 10 children and the root row is collapsed, the getVisibleRowCount() method will return 1.

***Return value***
**Number**

**getAllRowCount**

Gets the number of all rows (visible and invisible) contained in the rows collection.

***Return value*** Number

**getVisibleRowIds**

Gets the Array with the IDs of visible rows displayed in the grid.

***Return value*** Array

**getAllRowIds**

Gets the Array with the IDs of all rows contained in the rows collection.

***Return value*** Array

**getRowId**

Gets the row ID by the row index.

Input parameters for getRowId

| Name | Type | Description |
| --- | --- | --- |
| rowIndex | Number | Row index (zero based, from “top” to “bottom”). |

***Return value*** **String** - row ID.

#### getRowIndex

Gets the row index using the row ID. The method returns -1 if the row is not found or not visible.

Input parameters for getRowIndex

| Name | Type | Description |
| --- | --- | --- |
| rowId | String | Row ID |

***Return value*** **Number** - row index.

#### moveColumn

Moves a column to the specified position.

Input parameters for moveColumn

| Name | Type | Description |
| --- | --- | --- |
| columnName | String | Column name |
| columnIndex | Number | New column position – zero based index. |

#### setColumnSettings

Sets settings for the specified column.

Input parameters for setColumnSettings

| Name | Type | Description |
| --- | --- | --- |
| columnName | String | Column name |
| settings | Object | The Column settings object described in the section If a column setting is not specified in the column object, its value will not be changed. |

#### getColumnSettings

Gets settings for the specified column.

Input parameters for getColumnSettings

| Name | Type | Description |
| --- | --- | --- |
| columnName | String | Column name. |

***Return value*** **Object -** column settings object described in section Column settings object .

**setColumnMetadata**

Sets metadata for the specified column that will be used by the column cells without their own metadata.

Input parameters for setColumnMetadata

| Name | Type | Description |
| --- | --- | --- |
| columnName | String | Column name |
| metadata | Object | Metadata object described in the section . |

**getColumnMetadata**

Gets metadata for the specified column. Returns null if the column has no metadata.

Input parameters for getColumnMetadata

| Name | Type | Description |
| --- | --- | --- |
| columnName | String | Column name |

***Return value*** **Object** - metadata object described in section 10.4.1.

**setColumnVisibility**

Sets column visibility.

Input parameters for setColumnVisibility

| Name | Type | Description |
| --- | --- | --- |
| columnName | String | Required parameter. Column name. |
| visible | Boolean | Optional parameter. Column visibility. The default value is true. |

**getColumnVisibility**

Returns `true` if the specified column is visible, otherwise it returns `false` .

Input parameters for getColumnVisibility

| Name | Type | Description |
| --- | --- | --- |
| columnName | String | Column name |

***Return value***

**Boolean**

#### setColumnLabel

Sets the label for the specified column.

Input parameters for setColumnLabel

| Name | Type | Description |
| --- | --- | --- |
| columnName | String | Column name |
| label | String | Column label. |

#### getColumnLabel

Gets the label for the specified column.

Input parameters for getColumnLabel

| Name | Type | Description |
| --- | --- | --- |
| columnName | String | Column name |

***Return value*** **String** - column label.

#### setColumnWidth

Sets the width for the specified column.

Input parameters for setColumnWidth

| Name | Type | Description |
| --- | --- | --- |
| columnName | String | Column name |

| | | |
|---|---|---|
| width | Number | Column width |

**getColumnWidth**

Gets the width for the specified column.

Input parameters for getColumnWidth

| Name | Type | Description |
|---|---|---|
| columnName | String | Column name |

***Return value*** **Number** - column width.

**setCellMetadata**

Sets metadata for the specified cell.

Input parameters for setCellMetadata

| Name | Type | Description |
|---|---|---|
| rowId | String | Row ID |
| columnName | String | Column name |
| metadata | Object | Metadata object described in the section |

**getCellMetadataOnly**

Gets metadata for the specified cell. Returns null if the cell has no metadata (even if the column has metadata).

Input parameters for getCellMetadataOnly

| Name | Type | Description |
|---|---|---|
| rowId | String | Row ID |
| columnName | String | Column name |

***Return value*** **Object** - metadata object described in section 10.4.1.