GANTTCONTROL 3.0

DELPHI - Developer Reference
16/11/2023
Version 3.2

Weise Software GmbH
Bamberger Straße 4-6
01187 Dresden
Germany
http://www.gantt-komponente.de/
http://www.gantt-component.com

Inhalt

Component Overview

Abstract

The GanttChart is an interactive non database aware front-end VCL component for Delphi that visualises tasks in a gantt chart. The component may be used to schedule a number of resources and tasks and can be used in several project scenarios such as project management, task management, production scheduling or employee scheduling.

The component includes a pert chart view, a print preview component and some minor visual editor components for different data types.

Code-compatibility with the previous version

Due some design limits of the Version 2.0 of the GANTTCONTROL component the Version 3.0 has been completely redesigned and redeveloped from scratch. Your previous Delphi code will not be compatible and needs to be redeveloped in order to be applicable.

The component does not provide an isolated non visual DataSource component (WGSDataSource) anymore – its functionality is included in the internal DataController object. Furthermore, the component provides access to a single row object now and does not implement two different row object types for the tree and the table.

Supported Delphi versions

The GANTTCONTROL component supports following development environments:

- Embarcadero Delphi XE2
- Embarcadero Delphi XE3
- Embarcadero Delphi XE4
- Embarcadero Delphi XE5
- Embarcadero Delphi XE6
- Embarcadero Delphi XE7
- Embarcadero Delphi XE8
- Embarcadero Delphi 10 Seattle
- Embarcadero Delphi 10.1 Berlin
- Embarcadero Delphi 10.2 Tokyo
- Embarcadero Delphi 10.3 Rio
- Embarcadero Delphi 10.4 Sydney
- Embarcadero Delphi 11 Alexandria
- Embarcadero Delphi 12 Athens

The component supports the VCL framework and may be used for the 32bit or 64bit Windows platform.

Installation

Automatic Installation (recommended)

Please use the Installer – GanttInstaller.exe in order to install the package in your Delphi environment. The installer will compile the required packages, integrate it into your Delphi environment and add the search paths.

If you have any previous version installed, the installer will not install the component until you have manually removed the component packages from the Delphi IDE.

Manual Installation

Please make sure that the component is not already installed in your Delphi environment and remove it before you manually install it.

Delphi Version	Run-time package	Design-time package
Embarcadero Delphi XE2	GanttChart16.dpk	GanttChartDesign16.dpk
Embarcadero Delphi XE3	GanttChart17.dpk	GanttChartDesign17.dpk
Embarcadero Delphi XE4	GanttChart18.dpk	GanttChartDesign18.dpk
Embarcadero Delphi XE5	GanttChart19.dpk	GanttChartDesign19.dpk
Embarcadero Delphi XE6	GanttChart20.dpk	GanttChartDesign20.dpk
Embarcadero Delphi XE7	GanttChart21.dpk	GanttChartDesign21.dpk
Embarcadero Delphi XE8	GanttChart22.dpk	GanttChartDesign22.dpk
Embarcadero Delphi 10 Seattle	GanttChart23.dpk	GanttChartDesign23.dpk
Embarcadero Delphi 10.1 Berlin	GanttChart24.dpk	GanttChartDesign24.dpk
Embarcadero Delphi 10.2 Tokyo	GanttChart25.dpk	GanttChartDesign25.dpk
Embarcadero Delphi 10.3 Rio	GanttChart26.dpk	GanttChartDesign26.dpk
Embarcadero Delphi 10.4 Sydney	GanttChart27.dpk	GanttChartDesign27.dpk
Embarcadero Delphi 11 Alexandria	GanttChart28.dpk	GanttChartDesign28.dpk
Embarcadero Delphi 12 Athens	GanttChart29.dpk	GanttChartDesign29.dpk

1. Open the run-time package according to your Delphi Version and compile it.
2. Open the design-time package according to your Delphi Version compile it and install it.
3. Add the /lib and /source subdirectories to your Delphi library search path. In the Tools Menu |Options | Environment Options | Library | Library Path add the /lib and /source paths.

The components should now be installed correctly in your Delphi IDE.

Quick Start

In this chapter we will show the very basic steps to create a small sample application that demonstrate the usage of the component.

In Delphi, please create a new VCL project and place a TGanttChart component in your form. You will find the TGanttChart component in the Gantt Suite 3 section of the Tool-palette.

Select the newly placed TGanttChart component and select alClient for the Align property that you will find in the object inspector (F11).

Compile and run (F9) the application

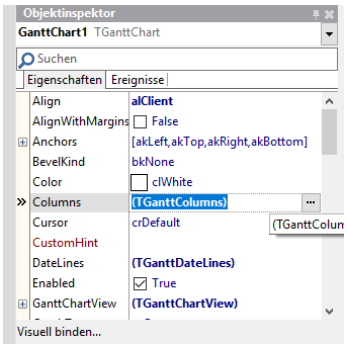
After you have accomplished the steps you should see a basic application that contains the gantt chart component. The component does already have a column named ‘Nr’ and a single time scale that displays the days of the week. However as there are no rows yet, the end user will not be able to do something useful.

In the next step we will add some more rows, some columns and some time scales.

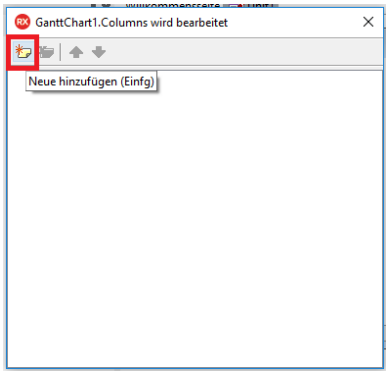
Select your form and go to the events section in the object inspector. Now double-click in the OnCreate event of the form. A new empty event handler will be created. Add the following line of code in the event handler in order to create 10 rows.

```
1.  procedure TForm1.FormCreate(Sender: TObject);
2.  begin
3.    GanttChart1.DataController.RowCount := 10;
4.  end;
```

Now please press F12 in order to switch back from the code view into the designer-view. Select the GanttChart component by left clicking on it. In order to create some more columns, please click the button (…) shown at the columns property.



Add at least two columns by clicking the Button ‘Add new’.



After closing the GanttChart.Columns enumerator window, please compile and run (F9) your application again.

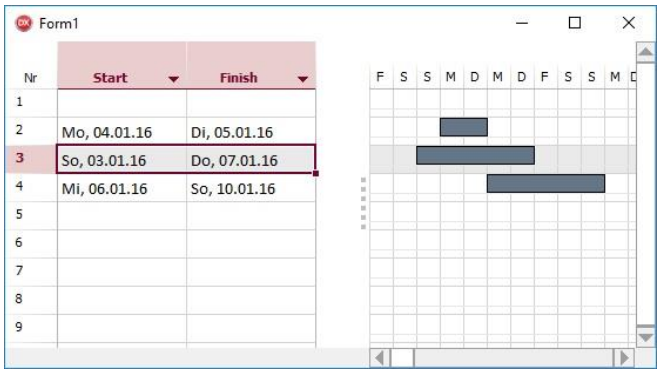
When everything is done correctly in the sample applications, the gantt chart has two more columns and ten new rows. The user is now able to edit cell values and drag in new bars in the right section of the gantt chart. In most cases it would be helpful if there is a column that shows the start and the end of a task. In order to implement that, we simply have to set the columns CellEditor property to AutoStart and AutoEnd.

Select the first column. Please note in the Delphi designer you simply can left click on a column header to select the column.

Please set AutoStart as value for the CellEditor property of the first column. You may set the columns Caption property to ‘Start’.

Please set AutoEnd as value for the CellEditor property of the second column. You may set the columns Caption property to ‘Finish’.

After you have run the sample application again and dragged in some bars, you should see that the value for the start and finish of the bars has been automatically set to the cell values. Additionally, you are now able to define a start and end of a task by editing the corresponding cell – the bar in the right section will be automatically updated.



Now we will add two more timescales that display week and month information.

Select the GanttChart component and open the TimeScales sub dialog by clicking the [...] button at the TimeScale property.

Add two more time scales. Please note that the lower most time scale has the id 0 and the top most time scale has the id 2.

Please select the time scale #1 and set the TimeMode to tmWeek and the TimeFormat to tmfWeekMonth

Now select the time scale #2 and set the TimeMode to tmMonth and the TimeFormat to tmfMonthMediumYear

Finally, we will add a GanttPrintPreview component and will show the print preview.

In order to do so, please add a new TGanttPrintPreview component onto your form. You will find the component in the GanttSuite3 section of the tool palette.

After you have added the component, please select it and link your gantt chart to the print preview by assigning the GanttChart property.

Now, please add a new button somewhere onto your form. Double click the button to create the button OnClick event handler.

```
1.  procedure TForm1.Button1Click(Sender: TObject);
2.  begin
3.   GanttPrintPreview1.ShowPreview();
4.  end;
```

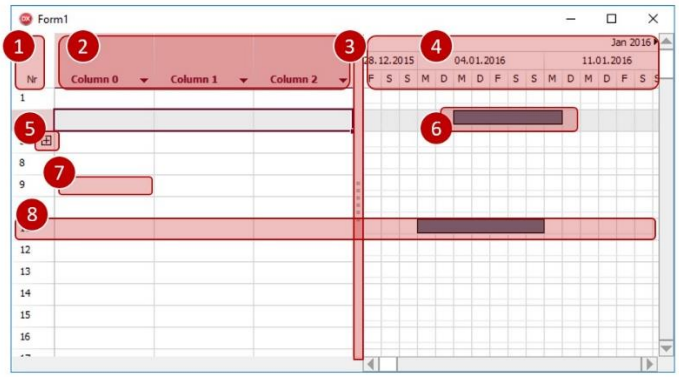
Please add the single line of code GanttPrintPreview1.ShowPreview(); into the event handler. At runtime of the application, the print preview will be shown after the user clicks on the button.

The sample application is finished so far. After having reproduced the steps, you should be able to set up the component, add basic elements such as rows, column, timescales and invoke a print preview.

At this point we suggest to have a look at the sample applications that comes along with the component. They may give you an overview of the features and functionalities of the component and how to implement them or you may continue to read the developer reference.

Gantt Chart

Overview



The GanttChart is an interactive user interface component that connects a tree grid structure with a table that visualizes bars or tasks based upon their scheduled start and finish dates. We will give a brief overview of the most relevant elements of the GanttChart component.

Nr	Item	Description
1	Indicator column	The indicator column. For each row the component will automatically display a row number. There are two options for numbering a row. You either can set a hierarchical row numbering (rnHierarchical) or a continuous row numbering (rnContinuous) for the Numeration property of the Indicator Column. The GanttChart.IndicatorColumn object provides access to the property of the Indicator column. The end user may resize the height of a row, or change to row order by mouse-dragging a cell of the indicator column.
2	Column headers	The column header of the column. You can optionally resize the columns width with the mouse. If you click on the triangle in the right of a column header, the column menu will be shown. The column menu allows sorting and filtering of rows. By default, if you double-click on a column header you can rename the caption of the column.
3	Splitter	There is a splitter object between the tree grid and the chart area of the GanttChart component. If you set the fixed property of the Splitter to true, the splitters position could not be changed by the end user. In order to specify a position for the splitter assign a numeric value (0-100) for the OpenPercent property. Alternatively you can specify the GanttChart.TreeWidth property.
4	Timescales	The Timescale objects of the GanttChart component. For each time scale object a TimeMode (tmDay, tmHour, tmWeek, tmMonth, tmYear, tmQuarter) and its corresponding TimeFormat can be specified. Please note that when you change the global TimeMode of the GanttChart component GanttChart.GanttChartView.TimeMode the time modes of the time scales are changing accordingly.
5	Collapsed Row	Rows can be structured hierarchically. Therefore rows can be collapsed or expanded.
6	Bar Object	Each row – with the exception of rows that has own child rows -can hold an unlimited amount of several bar objects, such as task bars, progress bars, milestones, text bars, image bars or shadow bars. Rows that have child rows itself (so called grouped rows or summary rows) automatically create a summary bar object and cannot hold other bars.
7	Cell	A single cell of the tree grid. Cells may be accessible by the GanttChart.AbsoluteRow[n].Cells[c] or the GanttChart.Rows[n].Cells[c] object. Each cell can hold unique values for the background color or the cells font.
8	Row	A row object of the Gantt Chart. There are three different ways to access rows. We recommend to use the GanttChart.AbsoluteRow[] Array as it provides easy and flat access to all rows ignoring any hierarchical structure. Alternatively, when requiring a hierarchical access, you should use the GanttChart.Rows[] array. The GanttChart.VisibleRows[] array provides access only to the unfiltered visible rows.

Rows (TGanttRow)

The TGanttRow object is a basic element of the GanttChart component as it provides access to its bars and cells.

Accessing Rows

As mentioned above, there are 3 ways on how to access row objects of the GanttChart.

Row Array []	Description
TGanttChart.AbsoluteRow[]	Disregarded of any hierarchical structure, all existing rows can be accessed using the AbsoluteRow[] array. The rows are indexed from top to bottom, where the topmost row has the index 0. Rows that are not visible due some column filtering match criteria are still part of the AbsoluteRow[] array.
TGanttChart.Row[]	To access rows hierarchical use the Row[] array. The Row[] array only contains the rows with the top most level of the hierarchy, any child rows must be accessed using the child[] Array. Rows that are not visible due some column filtering match criteria are still part of the Rows[] array.
TGanttChart.VisibleRow[]	The VisibleRow[] array provides a non-hierarchical access to all visible rows. Rows that are not visible because of some column filtering match criteria are not part of the VisibileRow[] array.

For each array there is a function to retrieve the total number of rows it holds:

Method	Description
TGanttChart. AbsoluteRowCount	Returns the number of rows in the AbsoluteRow[] Array .
TGanttChart. RowCount	Returns the number of rows in the Row[] Array.
TGanttChart. VisibleRowCount	Returns the number of rows in the VisibleRow[] Array.

For a given TGanttRow you can access its child using the child[] Array and the ChildCount function.

Method	Description
TGanttRow.Child[]	Provides Access to the child rows of a row.
TGanttRow.ChildCount	Returns the total number of childrows of a row.

Properties

The following list gives an overview of the properties of the TGanttRowObject:

Data:Pointer

Each row holds a Data pointer reference that may be used to point to any application specific user data objects.

Tag:integer; Tag2:integer; Tag3:integer; Tag4:integer

With the help of the Tag properties any developer that uses the component can store user defined integer values to each row for their own purpose. The Tag properties will be ignored by the GanttChart component.

ID:integer

The ID returns the index of the row according to its parent row. The first child row of a row does have the index 1. If the row itself is a top-level row and therefore has no parent, ID returns the number of the row within the chart.

Visible:Boolean

You can hide a row if you set the Visible property to false. Please note, when a row is not visible as it is not in the current view range, or a row is not visible because a column filter is applied. The visible property of the row may still return true.

Height:integer

The total height of a row in pixel. Each row can have a unique height. By default, the end-user is able to resize the row in the corresponding Indicator cell. If you do not want to allow the user to resize rows you can either set TGanttChart.OptionsBehaviour.RowResizing to false or set the AllowResize property of a specific TGanttRow to false.

Level:integer

The Level property returns the branch level for each row of the gantt chart. For the uppermost root level rows, level returns the value 0.

AbsoluteIndex:integer

The AbsoluteIndex property returns the index of the row in the AbsoluteRow[] array.

VisibleIndex:integer

The VisibleIndex property returns the index of the row in the VisibleRow[] array.

HierarchicalNumber:string

Returns the unique row identifier for a row in a hierarchical format, e.g. (“3.1” or “2.1.1”).

ContinousNumber:string;

Returns the unique row identifier for a row in a continuous format, e.g. (“1” or “2”).

Number:string

Based on the Numeration property of the IndicatorColumn, Number returns the unique row identifier in either a hierarchical or a continuous format, e.g (“3.1” or “4”).

Collapsed:Boolean

Returns true if the row is collapsed otherwise false.

Filtered:Boolean

Returns true if the row is filtered out or not.

Color:TColor

Specifies the color of the row. Please note, that each cell of the tree does have its own color. If you want to colorize the entire row you have to use the code shown below:

```
1.  ...
2.   GanttChart1.BeginUpdate;
3.   GanttChart1.AbsoluteRow[2].Color := clLime;
4.   for c:=0 to GanttChart1.AbsoluteRow[2].CellCount-1 do
5.    GanttChart1.AbsoluteRow[2].Cells[c].Color := clLime;
6.   GanttChart1.EndUpdate;
7.  ...
```

RowCalendar:TGanttRowCalendar

By default, the GanttChart component uses a single calendar (BaseCalendar) entity that defines all working and non-working date exceptions, such as holidays, weekends or vacations. The GanttControl 3.0 component provides the possibility to assign a different calendar to a single or a set of rows. If you assign a TGanttRowCalendar object to the RowCalendar property, the RowCalendar will be used for this row instead of the BaseCalendar.

StartDate:TDateTime

Returns the start date of the first (earliest) bar object in this row.

EndDate:TDateTime

Returns the end date of the last (latest) bar object in this row.

IsGroupRow:Boolean

Returns true, if the row does have children otherwise it will return false.

SummaryBar:TGanttSummaryBar

If the row has at least one children that holds a bar object, the GanttChart component will automatically create a TGanttSummaryBar object. The summary bar starts at the earliest start of a child bar object and ends at the latest

end of a child bar object. If the start- or enddate of the child bars will be changed the summary bar will update itself immediately.

DurationDisplayMode:TSpinDurationMode

Specifies whether the duration of a task, that will be automatically shown in the cells of AutoDuration columns, will be shown in hours (dmHours), days (dmDays) or weeks (dmWeeks).

AutomaticScheduling:Boolean

If a bar requires to be rescheduled due some restriction defined by a connection, the GanttChart component will reschedule the bar automatically if AutomaticScheduling is set to true, otherwise it will not reschedule the bar.

Note:string

Optionally the user can assign a text note to each row. To edit the row note you can call the TGanttChart.EditRowNote function.

PertBar:TPertBar

Returns the TPertBar object that is linked with the row.

AllowResize:Boolean

Specifies whether the end user can modify the row height of this row.

Methods / Functions

The following list gives an overview of the methods and functions of the TGanttRowObject.

function GetCalendar: TGanttCalendar;function CellCount:integer

Returns the calendar that is actually being used for this row. If the RowCalendar is nil for this row, it will return the BaseCalendar otherwise it will return the RowCalendar.

function Parent : TGanttRow

Returns the parent row object for the given row.

function GetRecursivChildRows(var AChildRows :TList<TGanttRow>):integer

Returns a list of all child rows and their recursive children of a given row.

function GetRecursivChildBars(var AChildBars :TList<TGanttBaseBar>; AOpBars:Boolean=false):integer

Returns a list of all child bars and their recursive child bars of a given row.

function ChildCount : integer

Returns the number of children a row has.

function GetFirstChild:TGanttRow

Returns the first child of a row.

function GetLastChild:TGanttRow

Returns the last child of a row.

function AddChild:TGanttRow

Adds a new child row to a specific row.

function InsertChild(APos:integer):TGanttRow

Insert a child at the specified absolute index.

procedure DeleteChilds

Deletes all children.

function DeleteChildRow(AIndex:integer):Boolean

Deletes a child row according to its Index.

function IsHotTracked:Boolean

Returns true if the current row is being hot tracked.

procedure Collapse

Collapses the row.

procedure Expand

Expands the row.

procedure MakeVisible

Will force the control, to set a proper scroll position, so that the row will be part of the view range.

procedure DeselectBars

Deselect all bars of the given row.

function BarCount:integer

Returns the number of bars the row holds.

function BarCount(ABarSet:TGanttBarTypeSet):integer

Returns the number of bars for a given bar type or a set of bar types.

function AddTaskBar(AStartDate, AEndDate : TDateTime):TGanttTaskBar

Adds a new task bar to the row and returns a reference on it.

function AddProgressBar(AStartDate, AEndDate : TDateTime):TGanttProgressBar

Adds a new progress bar to the row and returns a reference on it.

function AddMileStoneBar(AStartDate, AEndDate : TDateTime):TGanttMileStoneBar

Adds a new milestone bar to the row and returns a reference on it.

function AddTextBar(AStartDate, AEndDate : TDateTime):TGanttTextBar

Adds a new text bar to the row and returns a reference on it.

function AddImageBar(AStartDate, AEndDate : TDateTIme):TGanttImageBar

Adds a new image bar object to the row and returns a reference to it.

function CreatePertBar(AFromPertBar:TPertBar):TPertBar

Creates a new PertBar object for the given row. If AFromPertBar is not null it will copy the suitable properties from AFromPert.

procedure Assign(Source: TGanttRow)

Copies the properties from the Source row object.

constructor Create(AOwner: TComponent)

Constructor. Creates a new TGanttRowObject

destructor Destroy

Destructor. Destroys the TGanttRowObject

procedure ApplyVisualFrom(AGanttRow:TGanttRow)

Assigns the visual properties for the given row (such as color, cell values) from the AGanttRow parameter.

procedure RefreshAutoValues

Updates values for AutoStart, AutoEnd and AutoDuration columns.

Adding / removing rows

GanttChart.DataController.RowCount:integer

Use the RowCount property to obtain and set the total number of rows for the GanttChart component.

GanttChart.DataController.AddRow:TGanttRow

The AddRow() method will add a new row behind the last currently existing row and will return a reference to the newly created row.

GanttChart.DataController.ClearRow(ARow: TGanttRow)

The ClearRow(ARow:TGanttRow) method will remove all existing bars of a given row and clears all cells of a row. The visual properties of the cells will not be modified.

GanttChart.DataController.DeleteRow(ARowIndex:integer):Boolean

Removes the row, that is specified with the row index of the Row[] Array.

GanttChart.DataController.DeleteRow(ARow:TGanttRow):Boolean

Removes the row.

GanttChart.DeleteAbsoluteRow(AAbsoluteIndex:integer):Boolean

Removes the row, that is specified with the absolute Row Index.

GanttChart.DataController.DeleteSelectedRows (AllowPartialSelectedRows:Boolean): Boolean

Deletes the selected rows. If AllowPartialSelectedRows is true, then all rows will be deleted, that contains at least a single selected cell. If AllowPartialSelectedRows is set to false, then only rows are deleted that are fully selected (all cells are selected).

GanttChart.DataController. AddChildRow(AParentRow: TGanttRow): TGanttRow

Adds a new child row to the AParentRow row object.

GanttChart.DataController.InsertRow (AParentRow:TGanttRow; AIndex:integer):TGanttRow

This will add a new child row to the row that is set by the AParentRow parameter. AIndex will define the insert position of the row.

GanttChart.DataController. InsertAbsoluteRow(AAbsoluteIndex:integer):TGanttRow

InsertAbsoluteRow will insert a new row at the position that is defined by the AAbsoluteIndex parameter.

GanttChart.DataController.InsertChildRow()

Inserts a new child row for the given row and position.

GanttChart.DataController.DeleteChildRow()

Deletes a child row.

GanttChart.DataController.DeleteChilds()

Delete all children of a row.

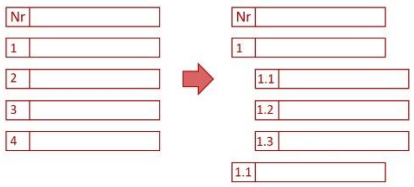
Hierarchical structuring of rows

The tree of the gantt graph allows it to create a hierarchical structure. There are several methods enabeling the developer to apply a hierarchical tree structure.

GanttChart.DataController.RowsGroup(AbsoluteStartIndex, AbsoluteEndIndex: integer): TGanttRow

You can group a range of rows and increase their nesting level by using the RowsGroup(AbsoluteStartIndex, AbsoluteEndIndex:integer) method. The range is determined by the AbsoluteStartIndex and the AbsoluteEndIndex.

```
1.  GanttChart1.DataController.RowsGroup(0,2);
```



Grouping the first three rows by using the RowsGroup(0,2) statement – the structure of the tree will alter as it is visualized in the pictogram on the left side.

GanttChart.DataController.AddChildRow(AParentRow: TGanttRow): TGanttRow

This will add a new child to the specified row.

GanttChart.DataController.RowsChangeParent(AbsoluteStartIndex, AbsoluteEndIndex : integer; ANewParent:TGanttRow):TGanttRow

The parent row of a set of child rows can be reassigned by using the RowsChangeParent(AbsoluteStartIndex, AbsoluteEndIndex:integer; ANewParent:TGanttRow):TGanttRow method. The range of rows, that's parentIndex will be reassigned is defined by the AbsoluteStartIndex and the AbsoluteEndIndex.

GanttChart.DataController.RowsUpgrade

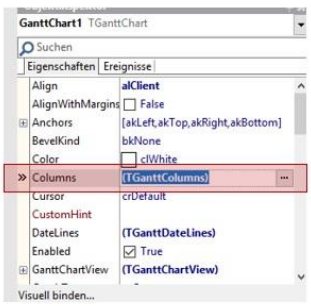
The RowsUpgrade method will decrease the hierarchical level of all selected rows.

GanttChart.DataController.RowsDowngrade

The RowsDowngrade method will increase the hierarchical level of all selected rows.

Columns (TGanttColumn)

By default, the tree of the gantt chart does only hold an Indicator column. The value of the cells of the indicator column is automatically generated by the component itself and displays the logical row identifier. When adapting the gantt control to your specific requirements its necessary to add columns to the tree.



Columns can be added and deleted at design time, if you edit the Columns property of the TGanttChart in the Object Inspector.

The most important property of a column is the CellEditor property. The columns editor type is specified by its CellEditor property.

Basically there are two types of columns – predefined columns and user defined columns. Predefined columns are columns that are handled completely by the logic of the gantt control component itself.

The following table shows all predefined column types and a brief description.

Predefined column types

CellEditor	Description
AutoStart	Displays the earliest start date of all bars for each row. When the user assigns a value the component will either create a new bar or reschedule an existing bar.
AutoEnd	Displays the latest end date of all bars for each row. When the user assigns a value, the component will either create a new bar or reschedule an existing bar.
AutoDuration	Displays the duration of all bars.

User defined column types

User defined columns are not handled by the component itself and hold several different CellEditor types, shown below:

CellEditor	Description
SingleLineText	Is used to edit a single line of text strings.
MultiLineText	Is used to edit or display multi line text.
ComboBox	Provides an editable combo box.
CurrencyEdit	Is used to edit and display currency values.
ButtonEdit	Is a composition of a single line of text and an additional button.
DateTime	Is used to edit and display datetime values.
SpinEdit	Is used to edit and display integer values, contains a spin edit control.
ImageComboBox	Contains an image combo box control.
Indicator	Columns with the Indicator-CellEditor can hold multiple IndicatorItems, that can be used to visualize special states of a row in form of an icon. The component provides hints for IndicatorItems. Additional there is the possibility to specify an automatic in-built logic for IndicatorItems, such as to set the viewport to the first bar, or to display and edit the row note or to toggle the automatic scheduling property of a row.

Adding, deleting columns at runtime

If you want to add columns at runtime by code, please use the code below. The parameter AColumnType defines the columns editor type and can take one of the following values (*etSingleLineText, etMultiLineText, etDateTime, etComboBox, etSpinEdit, etButtonEdit, etIndicator, etAutoStart, etAutoEnd, etAutoDuration, etCurrencyEdit, etImageComboBox*).

```
1.  with GanttChart1.DataController.AddColumn(etDateTime) do
2.  begin
3.    Caption := 'A Column';
4.    // set columns property here
5.  end;
```

Alternatively, you can use the following line of code. Here the overloaded AddColumn method does not take any parameters and creates a single line text editor column. You can change the columns editor type if you assign a new value to the PropertySelector property:

```
1.  with GanttChart1.DataController.AddColumn do
2.  begin
3.    Caption := 'A Column';
4.    PropertySelector := etDateTime;
5.  end;
```

In order to delete an existing column at runtime, you can use the DeleteColumn method of the DataController as shown in the code snippet below:

```
1.  GanttChart.DataController.DeleteColumn(1);
```

If it is required to change the order of two existing columns, use the MoveColumn method of the DataController. The following code will take the first column and will insert it after the second column:

```
2.  GanttChart.DataController.MoveColumn(0,2);
```

Below is a list of all functions and methods the data controller holds, in order to manipulate TGanttColumn column objects:

GanttChart.DataController.ColCount:integer

Returns the total number of columns.

GanttChart.DataController.SortColumn(AColumnID: integer; ASortOrder:ganttconsts.TColumnSortOrder)

SortColumn will perform a sorting on the Column defined by the parameter AColumnID. For the parameter ASortOrder, use csAsc for an ascending and csDesc for a descending sorting. In order to cancel the sorting please assign csNone. Please note, that after you have called the SortColumn method, any changes made in the cell values of the column, are not automatically sorted in.

GanttChart.DataController.CancelSort

CancelSort will cancel the sorting of any column.

GanttChart.DataController.ApplyFilter(AColumnID: integer; var AFilteredItems: TStrings)

If you want to set a filter to a specific column, you can use the ApplyFilter method. The parameter AFilteredItems is a TStrings object, such as a TStringList, that contains the strings that defines the filter.

GanttChart.DataController.CancelAllFilter

CancelAllFilters will cancel filtering of any column.

GanttChart.DataController.ColumnOfCell(ACell:TGanttCell):TGanttColumn

The method ColumnOfACell will return the corresponding column object of a cell.

GanttChart.DataController.AddColumn: TGanttColumn

AddColumn will add a new column and will return a reference to the column.

GanttChart.DataController.InsertColumn(Index: integer): TGanttColumn

InsertColumn will insert a new column at the position, defined by the parameter Index, and will return a reference to the column.

GanttChart.DataController.DeleteColumn(Index:integer)

Deletes the column, specified by the Index parameter.

GanttChart.DataController.MoveColumn(IndexFrom, IndexTo: integer)

Changes the position of the column. The column defined with the index IndexFrom will be repositioned to the new position, defined by the parameter IndexTo.

GanttChart.DataController.ClearColumns

Removes all columns.

GanttChart.DataController.ClearColumnFixation

Columns do have the property “FixedTillHere”. All columns left of the column that has the property FixedTillHere set to true, will remain permanent visible when the user scrolls the content of the tree horizontally. Only one column can have the boolean “FixedTillHere” property set to true at a given time. The method ClearColumnFixation will remove any column fixation of the tree.

GanttChart.DataController.GetFixedColumn

The method GetFixedColumn will return the index of the column that has its property “FixedTillHere” set to true.

GanttChart.DataController.EditCaptionTitle(AColumn: TGanttColumn)

The method EditCaptionTitle(AColumn:TGanttColumn) will set the columns header in an edit mode, so that the end user can edit the columns caption at runtime. If you do not want to make it possible for the end-user to edit the caption of the column at runtime by double-clicking the columns header, please set the boolean poperty AllowCaptionEditing of the column to false.

Properties

The following list gives an overview of the properties of the TGanttColumn object.

AllowCaptionEditing:Boolean

If set to true, the end-user is able to edit the columns caption by double-clicking the columns header.

AllowResize:Boolean

Allow resize determines whether it is possible for the end-user to resize the width of a column header.

AutoWidth:Boolean

If set to true, the width of a column will be adjusted according to the content of a column.

BarTextKey:string

If a value for the property BarTextKey has been assigned and matches with a value for a bar text (Bar.TextSettings.TextLeft, Bar.TextSettings.TextTop, Bar.TextSettings.TextRight, Bar.TextSettings.TextBottom or Bar.TextSettings.TextCenter) the component will automatically display the content of a cell at its corresponding bar.

In the sample code below you can see that both - the captions BarTextKey property and the TextLeft property of the bar - are mapped to the same value. As soon as you have created a bar object and have edited a cell value of the column, the cells value should be drawn left of the bar.

```
1.  procedure TForm1.FormCreate(Sender: TObject);
2.  begin
3.   with GanttChart1.DataController.AddColumn do
4.    begin
5.      Caption := 'Column';
6.      BarTextKey := '[mybartextkey]';
```

```
7.    end;
8.    GanttChart1.DataController.RowCount:=5;
9.  end;
10.
11. procedure TForm1.GanttChart1AfterBarAdd(Bar: TGanttBaseBar);
12. begin
13.   Bar.TextSettings.TextLeft := '[mybartextkey]';
14. end;
```

Caption:string

The caption of the column.

CellEditor: TEditorProperties

The CellEditor property provides access to specific column editor properties. Assign a new value to the PropertySelector property of the column if you want to modify the columns editor type at runtime. For accessing and setting sub properties of the cell editor it is required to typecast the CellEditor. Please refer to the next chapter on how to access and set properties of a cell editor.

Color:TColor

Specifies the background color of the cells header.

ColumnHintDescription.HintDescription:string

Use the HintDescription property to set up a multiline hint text for the column.

As you can see on the left side you can control the font style in the column hint.

The following code snippet shows how to set up the formatted text for the column hint.

```
1.  with GanttChart1.DataController.AddColumn do
2.   begin
3.     Caption := 'Column';
4.     BarTextKey := '[mybartextkey]';
5.     ColumnHintOptions.ShowHint := true;
6.     ColumnHintOptions.HintTitle := 'HintTitle';
7.     ColumnHintOptions.HintDescription := '@@@bLorem @dipsum dolor'+#13#10+
8.                                          '@@sit @iamet@d, consetetur sadipscing @u@i@belitr@d,'+#13#10+
9.                                          'dsed diam nonumy ...';
10.  end;
```

As you can see there are some control character that triggers special formatting of the text:

Character	Desscription
@@	Triggers the text formatting mode. Must be used, in order to use any of the control characters below.
@b	Sets the bold font style mode.
@i	Sets the italic font style mode.
@u	Sets the underline font style mode.
@d	Sets the default font style. The bold, italic and underline flags will be unset.

ColumnHintDescription.HintTitle:string

Sets a title for the columns hint.

ColumnHintDescription.ShowHint:Boolean

Determines whether the column hint will be shown or not.

ColumnMenuOptions.Filtering:Boolean

When set to true, the column menu provides a section where the end-user is able to apply a filter to the gantt chart.

ColumnMenuOptions.Sorting:Boolean

When set to true, the column menu provides the possibility to apply an ascending, descending sorting of the gantt chart.

DefaultCellHorizontalAlignment: TCellHorzAlign

Specifies the default horizontal alignment for cells of the column. However, each cell of the column may have its own independent alignment settings.

DefaultCellVerticalAlignment: TCellVerticalAlign

Specifies the default vertical alignment for cells of the column. However, each cell of the column may have its own independent alignment settings.

FilteredItems:TStrings

Defines the values for the column filter. Only rows of the gantt chart will be displayed that cells contain values which are part of the FilteredItems.

FixedTillHere:Boolean

All columns left of the column that has the property FixedTillHere set to true, will remain permanent visible when the user scrolls the content of the tree horizontally. Only one column can have the boolean “FixedTillHere” property set to true at a given time.

Font:TFont

The font object of the columns header.

HeaderGlyph:TPicture

It is possible to display a glyph for each column header.

HeaderGlyphHorzAlignment: TCellHorzAlign = (chaLeft, chaRight, chaCenter)

Defines the horizontal alignment for the header glyph.

HeaderGlyphVertAlignment:TCellVerticalAlign= (cvaTop, cvaBottom, cvaCenter)

Defines the vertical alignment for the header glyph.

HorzAlignment: TCellHorzAlign = (chaLeft, chaRight, chaCenter)

Horizontal alignment for headers caption.

MaxWidth:Integer

Maximal column with in pixel.

MinWidth:Integer

Minimal column with in pixel.

Printable:Boolean

If the printable flag has been set to false, the column will not be printed.

ReadOnly:Boolean

When set to true the cells of the column are not editable.

SortOrder: TColumnSortOrder=(csNone, csAsc, csDesc)

The rows of the gantt chart component will be sorted based upon the cell values of the column. If you have applied a sort order for the column, changes made to a cell values will not force the component to immediately readapt the rows sort order.

Tag:integer

Can be used by the developer to store any user defined unspecific integer value to a colum.

VertAlignment:TCellVerticalAlign= (cvaTop, cvaBottom, cvaCenter)

Vertical alignment for the headers caption.

Visible:Boolean

Defines the visibility of the column.

Width:integer

Specifies the width of the column in pixel.

Column type editors

The following chapter will introduce all column types and their specific properties. Each editor specify property can be accessed and set at run time, by typecasting the CellEditor of the column. The most important property of a column is the CellEditor property. The columns editor type is specified by its CellEditor property.

The code below typecasts the CellEditor property to a TSingleLineTextEditorProperties class and sets the AutoComplete property to false.

```
1.  TSingleLineTextEditorProperties(GanttChart1.Column[1].CellEditor).AutoComplete := false;
```

For each cell editor the associated EditorProperties class will be shown in round brackets.

SingleLineText (TSingleLineTextEditorProperties)

Columns of the single line text type are used to edit a single line of text. This column editor supports an auto complete functionality, which can be activated or deactivated by the AutoComplete property.

AutoComplete:boolean	When the AutoComplete property has been set to true and the end user is editing cell values, the gantt chart component tries to auto complete the current value based on all existing cell values of the column.

MultiLineText (TMultiLineTextEditorProperties)

Columns of the multi-line text type are used to edit a multi-line text including line breaks and word wrapping.

ScrollBars: TScrollStyle (ssBoth, ssHorizontal, ssNone, ssVertical)	Specifies whether and what kind of scroll bars are shown when editing a multi line text cell.

ComboBox

Columns of the combo box type provides an editable drop down combo box.

AutoCloseUp:Boolean	Determines whether the drop-down list closes automatically after the user has selected a list item.
AutoComplete:Boolean	AutoComplete positions entries on matching list values and saves key strokes.
AutoCompleteDelay:Integer	Sets the delay between pressing a key and trying to complete the field automatically.
AutoDropDown:Boolean	Determines whether the drop-down list is automatically opened when the user presses a key.
DropDownCount:Integer	Sets how many entries the drop-down list can display.
ItemHeight:Integer	Specifies the height of the entries in the drop-down list in pixels.
Items:TStrings	Allows access to the elements (strings) in the combo box. Use items to access the items in the combo box. Items can be added, inserted, deleted, or moved using the methods of items. By assigning items, you can copy the elements of another string list into the list of the combination field.
Style:TComboBoxStyle	Determines how the combo box is displayed.

CurrencyEdit

Columns of the currency edit type are used to edit and display currency values.

Currency:String	Defines the Currency symbol as string, such as ($, €, EUR, …).

ButtonEdit

Columns of the button edit type provides editable cells that are a composition of a single line of text and an additional button.

AutoComplete:Boolean	When the AutoComplete property has been set to true and the end user is editing cell values, the gantt chart component tries to auto complete the current value based on all existing cell values of the column.
ButtonCaption:String	The caption of the button.

Furthermore, columns of this type provides the OnButtonClicked event, that is raised when the user clicks on the button.

```
1.  procedure TForm1.GanttChart1Columns0ButtonClicked(Sender: TObject);
2.  var
3.   ACell   : TGanttCell;
4.   AColumn : TGanttColumn;
5.   ARow    : TGanttRow;
6.  begin
7.   if GanttChart1.Selection.GetFirstSelectedCell <> nil then
8.      begin
9.        ACell   := GanttChart1.Selection.GetFirstSelectedCell;
10.       AColumn := GanttChart1.Column[ACell.GetCellProperties.ColumnIndex];
11.       ARow    := GanttChart1.AbsoluteRow[ACell.GetCellProperties.AbsoluteRowIndex];
12.
13.       GanttChart1.TreeCellEditor.CancelEditor;
14.       ACell.Text := IntToStr(ARow.AbsoluteIndex)+','+IntToStr(AColumn.Index);
15.     end;
16. end;
```

The example shows how to get the cell and the corresponding column and row, whose button has been clicked. If you want to assign a value within the ButtonClicked event of the column, you first have to call the GanttChart.TreeCellEditor.CancelEditor statement(13) as shown above.

DateTime

Columns of the Date Time edit type are used to display date time values.

DateFormat:TDTDateFormat = (dfShort, dfLong);	Specifies whether the date value will be editable in short or long date time format.
Format:String	Format sets a custom format for displaying date / time values. Please refer to the next table that lists all format characters and their functions.
Kind: TDateTimeKind = (dtkDate, dtkTime);	Specifies whether date or time values are edited.
ShowCalWeeks:Boolean	When set to true, and the columns kind is set to dtkDate the drop down calendar, that appears when the user edits a date, displays additional calendar weeks.

The following table list all format character and their functions.

Item	Description
d	The one- or two-digit day.
dd	The two-digit day. Single-digit day values are preceded by a zero.
ddd	The three-character weekday abbreviation.
dddd	The full weekday name.
h	The one- or two-digit hour in 12-hour format.
hh	The two-digit hour in 12-hour format. Single-digit values are preceded by a zero.
H	The one- or two-digit hour in 24-hour format.
HH	The two-digit hour in 24-hour format. Single-digit values are preceded by a zero.
m	The one- or two-digit minute.
mm	The two-digit minute. Single-digit values are preceded by a zero.
M	The one- or two-digit month number.
MM	The two-digit month number. Single-digit values are preceded by a zero.
MMM	The three-character month abbreviation.
MMMM	The full month name.
t	The one-letter AM/PM abbreviation (that is, AM is displayed as "A").
tt	The two-letter AM/PM abbreviation (that is, AM is displayed as "AM").
yy	The last two digits of the year (that is, 2017 would be displayed as "17").
yyyy	The full year (that is, 2017 would be displayed as "2017").

SpinEdit

Columns of the spin edit type are used to edit and display integer values in a spin edit control.

MaxValue:integer	MaxValue specifies the maximum value for the Value property the spin edit control can take.
MaxValueAssigned:Boolean	MaxValueAssigned determines whether a range check for the maximum value should be performed.
MinValue:integer	MinValue specifies the minimum value for the Value property the spin edit control can take.
MinValueAssigned:Boolean	MinValueAssigned determines whether a range check for the minimum value should be performed.

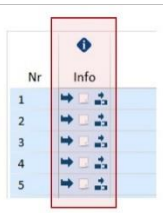
ImageComboBox

Columns of the image combo box type provides an editable combo box including an additional image.

AutoCloseUp:Boolean	Determines whether the drop-down list closes automatically after the user has selected a list item.
AutoComplete:Boolean	AutoComplete positions entries on matching list values and saves key strokes.
AutoCompleteDelay:Integer	Sets the delay between pressing a key and trying to complete the field automatically.
AutoDropDown:Boolean	Determines whether the drop-down list is automatically opened when the user presses a key.
DropDownCount:Integer	Sets how many entries the drop-down list can display.
ItemHeight:Integer	Specifies the height of the entries in the drop-down list in pixels.
Items:TStrings	Allows to access the elements (strings) in the combo box. Use items to access the items in the combo box. Items can be added, inserted, deleted, or moved using the methods of items. By assigning items, you can copy the elements of another string list into the list of the combination field.
Style:TComboBoxStyle	Determines how the combo box is displayed.
Images:TImageList	A reference to an image list that is used for displaying the images
ShowCaption:Boolean	Determines whether the caption are shown when in edit mode.

Indicator

Columns with the indicator cell editor can provide multiple indicator items, that can be used to visualize special states of a row in form of an icon. The component provides hints for indicator items. Additional there is the possibility to specify an automatic in build logic for Indicator items, such as to set the viewport to the first bar or to display and edit the row note or toggle the automatic scheduling property for a row.



The image on the left, shows an example of an indicator column that contains 3 indicator items.

The visibility of each indicator item can be set within the OnShowIndicatorItem of the gantt chart component.

DefaultVisible:Boolean	Specifies, whether the indicator items are by default visible (active).
DrawStyle:TIndicatorDrawStyle	When set to idsColumn all indicator glyphs hold their horizontal position independent from their visibility. When set to idsLeftAligned all visible indicator items will be shown left aligned.
DrawVerticalLine:Boolean	If set to true a small vertical line will be displayed between the glyphs of the indicator items.
Items:TIndicatorItems	The items array grants access to the indicator items. Use the Items array to access, add or remove single indicator items.
ShowInactiveGlpyhs:Boolean	When set to true, the component will draw the GlyphInactive otherwise no glpyh will be drawn for inactive indicators.

The table below shows the properties of a TIndicatorItem object:

AutoBehaviour: TGanttIndicatorAutoItem	AutoBehaviour can hold on of the following values (giaAutoSchedule, giaGoToBar, giaNote, giaNothing). The AutoBehaviour links indicator items to a prepared in build mechanism of the gantt chart component.	
	giaAutoSchedule	When the user clicks the AutoSchedule indicator item, the component toggles the AutoSchedule flag of the row. That determines whether bars are automatically scheduled based on their connections.
	giaGoToBar	When the end-user clicks the GoToBar indicator item, the gantt chart component scrolls to the bars of the row consecutively.
	giaNote	When the end-user clicks the Note indicator the gantt chart component shows an edit window that enables the user the editing of the row note.
	giaNothing	The indicator item is not linked to any automatic build in functionalities.
Caption:String	The caption of the indicator item is shown in the indicators hint.	
Glyph: TPicture	The glyph of the indicator, that is shown when the indicator is visible (active).	
GlyphInactive: TPicture	The glyph of the indicator, that is shown when the indicator is not visible (inactive).	
HintDescription:String	The description that is shown in the indicators hint window.	

The code sample below uses the OnShowIndicatorItem event to set all indicators visible for uneven rows. Within this event set the Boolean var parameter AShow to true, when the indicator items should be visible (active) or false if the indicator item should be invisible (inactive).

```
1.  procedure TForm1.GanttChart1ShowIndicatorItem(AIndicatorItem: TIndicatorItem;
2.    ARow: TGanttRow; AColumn: TGanttColumn; var AShow: Boolean);
3.  begin
4.   AShow:=  Odd(ARow.AbsoluteIndex);
5.  end;
```

If it is required to set a unique (row-dependant) hint for an indicator item, you can use the OnIndicatorGetHint event.

If it is required to react on a mouse down event on the indicator items, you can use the OnIndicatorMouseDown event.

AutoStart

Columns with the auto start editor type, display the earliest start of all bars for each row. When the user assigns a value the component will either create a new bar or reschedule an existing bar.

HandleTaskBars:Boolean	When set to false, the AutoStart column will be in read only mode and will only display the earliest start date of the bars. When set to true, changes made in cells of the columns, will automatically force the component to update the bars start date.
IncludeBars: TGanttBarTypeSet	Defines what kind of bar types should be considered for calculating the start date of the row.

AutoEnd

Columns with the auto end editor type, display the latest end date of all bars for each row. When the user assigns a value, the component will either create a new bar or reschedule an existing bar.

HandleTaskBars:Boolean	When set to false the AutoEnd column will be in read only mode and will only display the latest end date of the bars. When set to true, changes made in cells of the columns, will automatically force the component to update the bars end date.
IncludeBars: TGanttBarTypeSet	Defines what kind of bar types should be considered for calculating the end date of the row.

AutoDuration

Columns with the auto duration editor type, display the duration of all bars.

AsWorkingTime:Boolean	Determines whether the duration of the corresponding bars should be displayed excluding non-working times.
HandleTaskBars:Boolean	When set to false the AutoDuration column will be in read only mode and will only display the duratuion of the bars. When set to true, changes made in cells of the columns, will automatically force the component to update the duration of bars.
IncludeBars	Defines what kind of bar types should be considered for calculating the duration of bars.

IndicatorColumn (TGanttIndicatorColumn)

The indicator column contains cells that displays the logical row number. The rows can either be numbered hierarchically or continuously. Additionally, there are some end-user mouse interactions, such as resizing the height of a row or changing the row order, that may be triggered at cells of the indicator column.

Properties

The following list gives an overview of the properties of the TGanttIndicatorColumn object.

AllowResize:Boolean

AllowResize determines whether it is possible for the end-user to resize the width of the indicator column.

AutoWidth:Boolean

If set to true, the width of the indicator column will be adjusted according to the maximum width of the content of the indicator columns cells.

Caption:String

The caption of the indicator column.

Color:TColor

The background colour of the indicator column cells.

Font:TFont

The font object of the indicator columns header.

FontRow:TFont

The font object of the cells of the indicator columns.

HeaderColor:TColor

The background colour of the indicator columns header.

HorzAlignment:TCellHorzAlign

Horizontal alignment for the headers caption of the indicator column.

MaxWidth:integer

Maximal with of the indicator column in pixel.

MinWidth:integer

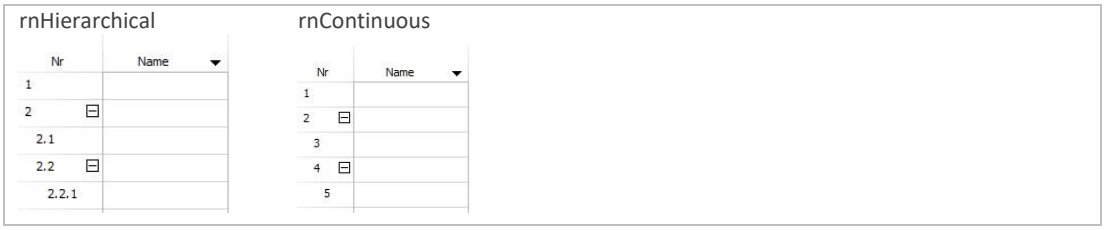
Minimal width of the indicator column in pixel.

NodeCollapserAlign: TNodeCollapserAlign= (ncLeft, ncRight)

Specifies whether the row collapse symbol is located on the left or on the right side within a cell of the indicator column.

Numeration: TRowNumeration = (rnHierarchical, rnContinuous, rnNone)

You can either set a hierarchical row numbering (rnHierarchical) or a continuous row numbering (rnContinuous) for the Numeration property of the Indicator Column



The table above shows the different row numbering options.

Printable:Boolean

If the printable flag has been set to false, the indicator column will not be printed.

TreeNodeIndentWidth:integer

TreeNodeIndentWidth determines how many pixels the horizontal position of the number text will be increased for each new hierarchical level.

UseRowColorAsColor:Boolean

When UseRowColorAsColor is set to true, the gantt chart component uses the color of the row object for the rows indicator cell. The color of the row can be specified by the Color property of a row.

VertAlignment:TCellVerticalAlign = (cvaTop, cvaBottom, cvaCenter)

Vertical alignment for the caption of the indicator columns header.

Visible:Boolean

Defines whether the indicator column is visible or not.

Width:integer

Specifies the width of the indicator column in pixel.

Time scales (TGanttTimeScales)

Time scales are an essential part of the gantt chart component and enable the end-user to identify the start and end dates and the chronological order of elements.

You can add as many time scales as you want – each time scale does have its own time mode. The time mode describes the time spans the time scale will be segmented in. For example, a time scale can visualize days or months if the time mode is set to tmDay or tmMonth.

But however keep in mind, that the gantt chart component itself does have the GanttChart.GanttChartView.TimeMode property. This is the global time mode of the component – whenever you change the time mode of the gantt chart component – the time mode of the time scales will be readapted.

The following table shows the possible values for the time mode (TGanttChartTimeMode):

TimeMode	Time unit
tmHour	Hours
tmDay	Days
tmWeek	Weeks
tmMonth	Months
tmQuarter	Quarters
tmYear	Years

Accessing time scales

All time scales can be accessed in the TGanttChart.TimeScales[] array. The draw order of the time scales may be misleading as the time scales will be drawn from bottom to top – therefore if you access time scales, keep in mind that the time scale with the index 0 will be the bottom most.

Properties

The following table gives a list of all properties of a time scale object.

Color:TColor

The background colour of the time scale.

DisplayFormat:string

DisplayFormat sets a custom format for displaying date / time values. Please refer to the next table that list all format character and their effects.

Character	Description
d	The one- or two-digit day.
dd	The two-digit day. Single-digit day values are preceded by a zero.
ddd	The three-character weekday abbreviation.
dddd	The full weekday name.
h	The one- or two-digit hour in 12-hour format.
hh	The two-digit hour in 12-hour format. Single-digit values are preceded by a zero.
H	The one- or two-digit hour in 24-hour format.
HH	The two-digit hour in 24-hour format. Single-digit values are preceded by a zero.
m	The one- or two-digit minute.
mm	The two-digit minute. Single-digit values are preceded by a zero.

M	The one- or two-digit month number.
MM	The two-digit month number. Single-digit values are preceded by a zero.
MMM	The three-character month abbreviation.
MMMM	The full month name.
t	The one-letter AM/PM abbreviation (that is, AM is displayed as "A").
tt	The two-letter AM/PM abbreviation (that is, AM is displayed as "AM").
yy	The last two digits of the year (that is, 2017 would be displayed as "17").
yyyy	The full year (that is, 2017 would be displayed as "2017").

DrawHorzLine:Boolean

Specifies whether a horizontal line should be drawn at the bottom of the time scale.

DrawVertLine:Boolean

Specifies whether a small vertical line should be drawn between at time segments of a time scale.

Font:TFont

The font object of the time scale.

FontAutoSize:Boolean

If set to true, the component will automatically apply the best font size, so that the date label still fits into the given time segment. Especially if you implement some kind of zooming for the width of a time segment, or changing the time mode at runtime, it is recommended to set the FontAutoSize to true.

Height:integer

The height of a time scale in pixel. Please note, that the total height of all visible time scales, determines the total height of the column headers.

HotTracked:Boolean

Determines whether single time segments can be highlighted when moving the mouse cursor over them.

ShowHint:Boolean

ShowHint specifies, whether a hint should be displayed for a time segment.

TimeFormat: TGanttTimeFormat

The TimeFormat defines, in which way the displayed date label is formatted.

The following table shows all time formats, the corresponding time mode and a brief example.

TimeFormat	TimeMode	Example
tmfNone	(all)	
tmfHourStandard	tmHour	0, 1, 2, 3, .. , 23
tmfHourAMPM	tmHour	1 AM, 2 AM , 3 AM , ..., 11 PM
tmfHourTime	tmHour	00:00, 01:00, 02:00, ..., 23:00
tmfDayFull	tmDay	Monday, Tuesday, Wednesday, ...
tmfDayShort	tmDay	M, T, W, T, F, S, S
tmfDayOfYear	tmDay	1, 2, 3, ..., 364, 365
tmfDayOfMonth	tmDay	1, 2, 3, .., 30, 31
tmfDayOfWeek	tmDay	1, 2, 3, 4, 5, 6, 7
tmfDayShortDayNumber	tmDay	Mo 5, Tue 6, We 7, Thu 8, Fr 9, ...
tmfWeekYear	tmWeek	1.CW, 2. CW, 3. CW, ...52. CW
tmfWeekYearNr	tmWeek	1, 2, 3, 4, 5, ..., 52
tmfWeekMonth	tmWeek	1.W, 2.W, 3.W, 4.W
tmfWeekMonthNr	tmWeek	1, 2, 3, 4
tmfWeekStartDay	tmWeek	02/04/2017, 09/04/2017
tmfMonthFull	tmMonth	January, Februrary, ..., December

tmfMonthMedium	tmMonth	Jan, Feb, ..., Dec
tmfMonthShort	tmMonth	J, F, M, ..., D
tmfMonthFullYear	tmMonth	January 2017, Februrary 2017, December 2017
tmfMonthMediumYear	tmMonth	Jan 2017, Feb 2017, Dec 2017
tmfMonthShortYear	tmMonth	J2017, F2017, ..., D2017
tmfMonthNumber	tmMonth	1, 2, 3, ..., 12
tmfMonthNumberYear	tmMonth	1 2017, 2 2017, 3 2017, ..., 12 2017
tmfQuarterFull	tmQuarter	1. Quartal, 2. Quartal, 3.Quartal
tmfQuarterShort	tmQuarter	1.Q, 2.Q, 3.Q, 4.Q
tmfQuarterFullYear	tmQuarter	1. Quartal 2017, 2. Quartal 2017
tmfQuarterShortYear	tmQuarter	1. Q 2017, 2.Q 2017, 3. Q 2017
tmfQuarterRoman	tmQuarter	I, II, III, IV
tmfYearFull	tmYear	2017, 2018, 2019, 2020
tmfYearShort	tmYear	17,18,19,20

TimeMode: TGanttChartTimeMode = (tmHour, tmDay, tmWeek, tmMonth, tmQuarter, tmYear)

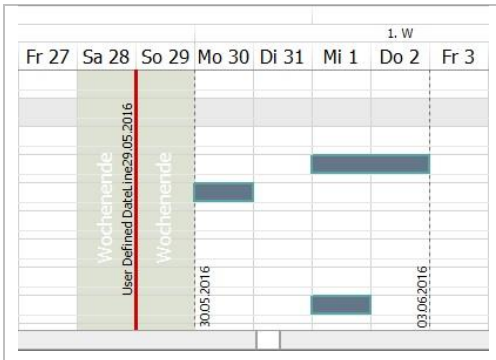
The time mode of the time scale, specifies the time/date range for a time segment.

Visible:Boolean

Defines whether the time scale is visible or not.

DateLines (TGanttDateLines)

If you want to highlight a specific date or time in the gantt chart you can add a DateLine in the DateLines[] array. DateLines can be formatted in various ways and provide some optional built-in-logic for showing the current date or the start date/ end date for all selected bars.



The image on the left shows three date lines. The first (red bold line) shows a user defined date lines on the 29th May 2016. The second and third date lines shows the start and end of the range of the selected bars.

DateLines are highly adaptable. You can define the color, line style, line widh or text orientation.

Accessing date lines

DateLines can be accessed in the TGanttChart.DateLines[] array.

Properties

Caption:String

The caption of the date line.

Color:TColor

The colour value of the date line.

Date:TDateTime

Specifies the Date of the date line. The component will draw the date line at the specified date, when the DateLineType has been set to gdlDateValue.

DateLineType: TGanttDateLineType = (gdlDateValue, gdlCurrentDate, gdlSelectedBarsStart, gdlSelectedBarsEnd)

Specifies whether the date line will be drawn at a user specified date or calculated by built-in-logic of the component.

DateLineType	Effect
gdlDateValue	The date line will be drawn at the date specified by the Date property.
gdlCurrentDate	The date line will be drawn at the current date and the current time.
gdlSelectedBarsStart	The date line will be drawn at the start of the earliest selected bar.
gdlSelectedBarsEnd	The date line will be drawn at the end of the latest selected bar

Font:TFont

The font object of the date line.

PenStyle:TPenStyle (psSolid, psDash, psDot, psDashDot, psDashDotDot, psClear, psInsideFrame, psUserStyle, psAlternate)

Specifies the pen style for the date line.

ShowDateAsCaption:Boolean

Specifies whether the component shows the actual date at the date line or not. You still can assign a value for the caption property. The component will concatenate the date and the caption.

TextHorzAlignment: TAlignment = (taLeftJustify, taRightJustify, taCenter)

The horizontal alignment of the text of the date line. Determines whether the caption will be draw on the left of the date line, right of the date line or centred on the date line.

TextOrientation: TTextOrientation = (toHorizontal, toVertical)

Specifies whether the label will be drawn horizontally or vertically (rotated).

TextVertPosition:TVerticalAlignment = (taAlignTop, taAlignBottom, taVerticalCenter)

The horizontal position of the label.

Visible:Boolean

Specifies whether the date line is visible.

Width:integer

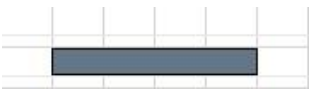
Specifies the width in pixel of the date line.

Bars (TGanttBar)

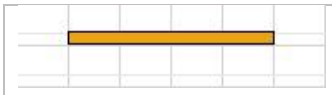
Bars are mainly used to visualize planned tasks or activities and their chronological determination. The component holds various bar types for different types of activities or to provide the user with additional information.

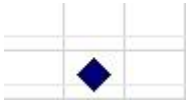
Each row (that has no child rows) can own an unlimited number of bar objects. Bars – with the exception of shadow bars – must not overlap.

See the following table for the different bar types, the component provides:

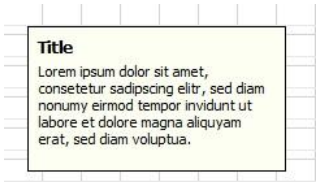
Bar	Description
TGanttTaskBar 	The task bar is the main element of the gantt chart. It is used to visualize a single (planned/target) activity. Task bars can be linked together in form of a single connection. Task bars are basically defined and determined by their StartDate and EndDate as well as the row in which they are displayed.

Bar	Description
TGanttProgressBar	A progress bar is commonly used to visualize the effective progress of a task. Therefore it can be displayed under a task bar in the same row as the task bar. Please not that a progress bar is not limited to be located within the range of a task bar. Progress bars can be linked together.

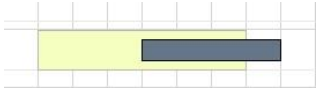


Bar	Description
TGanttMileStoneBar 	Milestones represent special dates with a unique character, for example a deadline where a project has to reach a certain state. A milestone will be represented as a small diamond symbol that visualizes a specific time / date. Milestones do have a fixed size and are not resizable.

Bar	Description
TGanttImageBar 	The TGanttImageBar object can be used to display images. Image bars are resizable. Image bars cannot be linked together with connections.

Bar	Description
TGanttTextBar 	To display any textual information inside the gantt chart, a suitable option would be to use a text bar. The text bar does have an additional title string. Text bars cannot be linked together.

Bar	Description
TGanttSummaryBar 	If the child row of a grouped row contains at least a taskbar, a milestone or a progress bar, a summary bar - shown on the left – will be created automatically. If the start- or enddate of the child bars will be rescheduled the summary bar will update itself immediately. Please note, that it is not possible to create any other bar within a grouped row.

Bar	Description
TGanttShadowTaskBar, TGanttShadowProgressBar 	The TGanttShadowTaskBar and TGanttShadowProgressBar are non-interactive gantt bar objects. The end-user is unable to delete or modify them. Also the shadow bars are the only bar types that may overlap other bars – as shown on the left side. Shadow bars are mainly used to display specific time ranges in the background behind task bars. In some applications shadow bars are used to visualize a previous state of planned activities, the actual state of planned activities is visualized by task bars.

The end user is able to create new bars by holding down the left mouse button and drag in a new object into a row of the gantt chart. By default, all new bars that will be inserted this way are task bars. However you can specify the bar type by assign a value to the GanttChart.GanttChartView.DragNewBarType property.

The following table lists all possible values for the DrawNewBarType property.

Value	Created bar type

nbImageBar	Image bars
nbMileStoneBar	Milestone bars
nbProgressBar	Progress bars
nbTaskBar	Task bars
nbTaskBarOrProgressBar	Progress bars or Task bars. The end-user can create either task bars or progress bars object, depending on the vertical position within a row.
nbTextBar	Text bars.

Accessing bar objects

Bar objects are accessible throughout the Bar[] Array of the GanttRow. If you have accessed a bar object and want to set a specific property you may have to typecast the bar.

TGanttRow.Bars[]:TGanttBaseBar	The Bar array provides access to the bars of a row.
TGanttRow.BarCount:integer	BarCount returns the total number of all bars that are located in the row.
BarCount(ABarSet:TGanttBarTypeSet) :integer;	You can define a set of bartypes as a parameter for the BarCount function. BarCount will then return only the number of bars, defined by the ABarSet set.
TGanttRow.SummaryBar	Provides access to the summary bar if the row has a summary bar, otherwise it will return NIL.

The code example below iterates through all the task bars and progress bars in the third row and set their startdate to the current date.

```
1.  var
2.   i : integer;
3.  begin
4.   if GanttChart1.AbsoluteRow[2].BarCount([btTaskBar, btProgressBar])  > 0 then
5.    for i := 0 to GanttChart1.AbsoluteRow[2].BarCount do
6.      begin
7.        if GanttChart1.AbsoluteRow[2].Bar[i].BarType in [btTaskBar, btProgressBar] then
8.             GanttChart1.AbsoluteRow[2].Bar[i].StartDate := Trunc(Now);
9.      end;
10. end;
```

Adding, Deleting bars

Please note, that if you perform a bunch of data manipulative operations it is important to put the command between a GanttChart.DataController.BeginUpdate and GanttChart.DataController.EndUpdate statement – as this will improve the performance a lot.

TGanttRow.AddTaskBar(AStartDate, AEndDate: TDateTime): TGanttTaskBar

Adds a new task bar to the row and returns a reference on it.

TGanttRow.AddProgressBar(AStartDate, AEndDate: TDateTime): TGanttProgressBar

Adds a new progress bar to the row and returns a reference on it.

TGanttRow.AddMileStoneBar(AStartDate, AEndDate: TDateTime): TGanttMileStoneBar

Adds a new milestone bar to the row and returns a reference on it.

TGanttRow.AddTextBar(AStartDate, AEndDate: TDateTime): TGanttTextBar

Adds a new text bar to the row and returns a reference to it.

TGanttRow.AddImageBar(AStartDate, AEndDate : TDateTIme):TGanttImageBar

Adds a new image bar object to the row and returns a reference to it.

GanttChart.DataController.AddSummaryBar(AStartDate, AEndDate: TDateTime; ARow:TGanttRow): TGanttSummaryBar

Adds a summary bar to the row. Please note, that summary bars are by default controlled by the components logic.

GanttChart.DataController.AddTaskBar(AStartDate, AEndDate: TDateTime; ARow: TGanttRow): TGanttTaskBar

Adds a new task bar to the row and returns a reference to it.

GanttChart.DataController.AddProgressBar(AStartDate, AEndDate : TDateTime; ARow:TGanttRow): TGanttProgressBar

Adds a progress bar to the row and returns a reference to it.

GanttChart.DataController.AddMileStoneBar(AStartDate, AEndDate: TDateTime; ARow: TGanttRow): TGanttMileStoneBar

Adds a milestone bar object to the row and returns a reference to it.

GanttChart.DataController.AddTextBar(AStartDate, AEndDate: TDateTime; ARow: TGanttRow): TGanttTextBar

Adds a text bar object to the row and returns a reference to it.

GanttChart.DataController.AddImageBar(AStartDate, AEndDate : TDateTime; ARow:TGanttRow): TGanttImageBar

Adds an image bar object to the row and returns a reference to it.

GanttChart.DataController.AddShadowTaskBar(AStartDate, AEndDate: TDateTime; ARow: TGanttRow): TGanttShadowTaskBar

Adds a shadow task bar object to the row and returns a reference to it.

GanttChart.DataController.AddShadowProgressBar(AStartDate, AEndDate: TDateTime; ARow:TGanttRow):TGanttShadowProgressBar

Adds a shadow progress bar object to the row and returns a reference to it.

GanttChart.DataController.RemoveBar(ABar: TGanttBaseBar; AKeepConnection:Boolean=false):Boolean

Removes the specified bar (ABar), returns true if successful otherwise false. If a connection is linked to the bar, it will be deleted if AKeepConnection has the value false.

GanttChart.DataController.RemoveBarsFromRow(ARow:TGanttRow; AKeepConnection:Boolean=false):Boolean

Removes all bars from the specified row (ARow). If a connecton is linked to the bar, it will be deleted, if AKeepConnection has been set to true.

The code below adds a task bar at row 2.

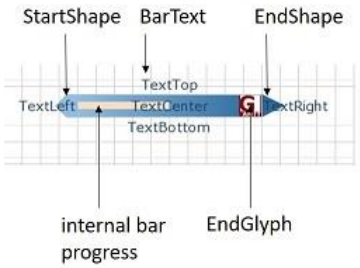
```
1.  // BeginUpdate - EndUpdate.. in case of multiple data manipulative operations
2.  GanttChart1.BeginUpdate;
3.  GanttChart1.AbsoluteRow[2].AddTaskBar(Trunc(Now), Trunc(Now)+10);
4.  GanttChart1.EndUpdate;
```

Task bars (TGanttTaskBar)

The most important bar type– as this is the basic bar type of gantt charts – is the task bar. This chapter handles the properties and the possibilities to change the layout.

A task bar is defined by its start- and end date. Use the TGanttTaskBar.StartDate:TDateTime or TGanttTaskBar.EndDate:TDateTime property to assign a new value or to retrieve the current value. Please note, that when assigning a new time range for taskbars the EndDate must always been larger than the StartDate. For example: If you want to change the duration of a task bar to one day, the EndDate should be EndDate := StartDate + 1.

Properties



The picture on the left shows a taskbar that has a progress value of 50 percent, an EndGlyph, assigned bar texts and a Start- and EndShape.

Progress:integer

For each taskbar you can set an internal bar progress within a range of 0% to 100% - that is visualized as a horizontal line inside the task bar.

BufferTimeLeft:TDateTime

It is possible to fill a range with an optional pattern on the inner left side on the bar as a buffer time. BufferTimeLeft specifies the buffer time in days on the start of the task bar.

BufferTimeRight:TDateTime

It is possible to fill a range with an optional pattern on the inner right side on the bar as buffer time. BufferTimeRight specifies the buffer time in days on the end of the task bar.

StartGlyph:TPicture

Assigns a glyph that is displayed at the start of the task bar.

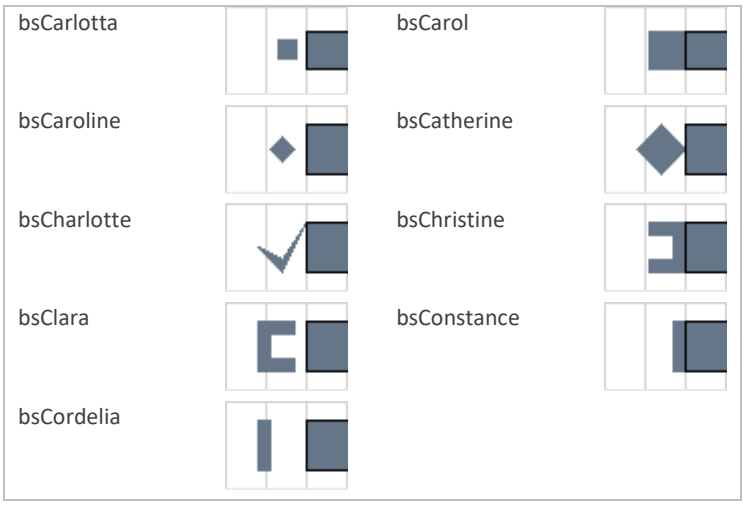
EndGlyph:TPicture

Assigns a glyph that is displayed at the end of the task bar.

StartShape:TBarShape

Assigns a small shape at the start of the task bar. Please refer to the following table for the different possible values:

TBarShape value	Shape	TBarShape value	Shape
bsNothing		bsAbbie	
bsAdrienne		bsAgatha	
bsAlexandra		bsAmanda	
bsAmy		bsAnastasia	
bsAndrea		bsAnne	
bsAntonia		bsAshley	
bsAurora		bsBabette	
bsBetty		bsBeverly	
bsBridget		bsCarla	

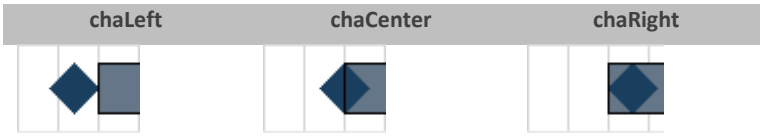


StartShapeColor:TColor

Specifies the color value for the Start shape symbol.

StartShapeAlignment:TCellHorzAlign

The horizontal alignment of the start shape. The following table shows the three possible values.



EndShape:TBarShape

Assigns a small shape at the end of the task bar. Please refer to the table above for the different possible values.

EndShapeColor:TColor

Specifies the color value for the End shape symbol.

EndShapeAlignment:TCellHorzAlign

The horizontal alignment of the end shape.

Data:Pointer

Each task bar holds a Data pointer reference that may be used to point to any application specific user data objects.

StartDate:TDateTime

The start date of the task bar. Settings a new value for the StartDate property will reschedule the task bar.

EndDate:TDateTime

The end date of the task bar. Setting a new value for the EndDate property will reschedule the task bar.

BarType:TGanttBarType = (btBaseBar, btTaskBar, btSummaryBar, btProgressBar, btMileStoneBar, btTextBar, btImageBar, btShadowTaskBar, btShadowProgressBar, btPertBar)

For taskbars the BarType returns btTaskBar.

HotTracked:Boolean

Returns true if the task bar is currently hot tracked.

Selected:Boolean

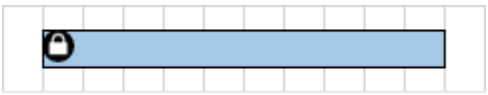
Returns true if the task bar is selected.

GUID:TGUID

The gantt chart component assigns to each task bar a unique Globally Unique Identifier (GUID) in order to distinguish the bar objects.

Locked:Boolean

If the task bar is locked, it will be unmodifiable by the end-user. Also locked task bars are fixed and will not be rescheduled automatically by the component. A small symbol – displaying a lock – is drawn for all locked bars in the upper left corner of the bar.



Height:integer

The height in pixel of a task bar.

Color:TColor

The color of the task bar. Please note that dependant from your draw style, it may be the case that the color property is not used when drawing the task bar. Please refer to this Section – Draw Options – for further details.

Tag:integer

With the help of the Tag property any developer that uses the component can store user defined integer values to each task bar for their own purpose. The Tag property will be ignored by the GanttChart component.

Border:Boolean

Specifies whether a border will be drawn or not.

BorderWidth:integer

Specifies the width of the border in pixel.

BorderColor:TColor

Specifies the border color.

AutomaticSchedulung:Boolean

Read-only property. The value will be determined by the AutomaticScheduling property of the corresponding TGanttRow. When set to false the component will not automatically reschedule the task bar due any changes to a connected bar.

Row:Pointer

Holds a reference pointer to the corresponding bar object.

Owner:Pointer

Holds a reference to the corresponding gantt chart object.

Successors:TList<TGanttBaseBar>

Provides a list of all directly connected bars that are successors.

Predecessor:TList<TGanttBaseBar>

Provides a list of all directly connected bars that are predecessors.

AllowVerticalDrag:Boolean

Read-only property. Task bars can be dragged from one row to another – therefore the property returns true.

AllowHorizontalDrag:Boolean

Read-only property. Task bars can be dragged horizontally in a row.

AllowMultiRow:Boolean

Read-only property. It is not possible for task bars to be larger than the height of a single row, so the property returns false for task bars.

AllowConnections:Boolean

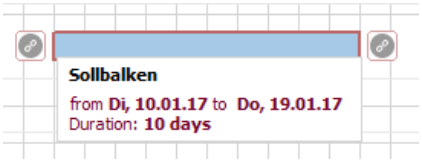
Read-only property. Task bars can be connected to other bar objects – so the property will be true for task bars.

Hint:Boolean

Specifies whether the component should provide a hint for the task bar.

DefaultHint:Boolean

Specifies whether the DefaultHint or a user-defined hint (see HintTitle and HintDescription) will be used.



HintTitle:string

Specifies the title for the user defined hint.

HintDescription:string

Specifies the description for the user defined hint.

GetDuration:Double

Returns the total (working and non-working) duration of the task bar.

GetEffectiveDuration:Double

Returns the work time duration for the task bar.

SetEffectiveDuration(AValue:TDateTime)

Sets the work time duration for the task bar.

IsPartOfCriticalPath:Boolean

The Gantt Control component includes the calculation of the critical path. The calculation of the critical path determines which tasks are critical and non-critical. When a task is critical any delay of this task will result in a delay of the entire project. The critical path is the sequence of all critical tasks.

Returns true if the task bar is part of the critical path, otherwise it will return false.

DrawOptions.DrawStyle: TGanttBarDrawStyle = (dsSolid, dsGradient, dsPattern, ds3D)

There are four different ways a task bar can be drawn. These are a solid fill style (dsSolid), a gradient style fill (dsGradient), a pattern draw style (dsPattern) and a three dimensional draw style (ds3D). The following table shows the draw style and the resulting task bar.

DrawStyle	Taskbar
dsSolid	
dsGradient (horizontal)	
dsGradient (vertical)	
dsPattern	
ds3D	

DrawOptions.GradientStartColor:TColor

Specifies the start color for the gradient draw style.

DrawOptions.GradientEndColor:TColor

Specifies the end color for the gradient draw style.

DrawOptions.GradientDirection:TGradientDirection

Specifies the direction for the gradient. TGradientDirection is defined in the unit **VCL.GraphUtil**. You have to add this to the uses clause.

DrawOptions.BrushStyle:TBrushStyle

Defines the type of pattern that is used when drawing.

DrawOptions.PatternColor:TColor

Specifies the color that is used when drawing the foreground lines of the patterns.

DrawOptions.ProgressColor:TColor

Specifies the color, that is used when drawing the internal bar progress.

CPMData.EarliestStart:TDateTime

A datetime property defining the ealiest possible start time of a task/bar. The earliest start time depends on the connection ending at the bar. The task cannot start earlier because other tasks have to be finished/started first.

CPMData.EarliestFinish:TDateTime

A datetime property defining the earliest finish time of a task. The earliest possible finish time of a task is calculated by the earliest possible start time and its duration.

CPMData.LatestStart:TDateTime

A datetime property defining the latest possible start time of a task/bar. The latest start time depends on the connection starting from the bar. If the bar would be moved only one day later then the whole project end date would be delayed.

CPMData.LatestFinish:TDateTime

A datetime property defining the latest possible finish time of a pert bar. The latest possible finish time of a task is calculated by the latest possible start time and its duration.

CPMData.LocalBuffer:TDateTime

A datetime property defining the free buffer of a task. The free buffer is the number of intermediate days between this bar and the earliest connected followed bar.

CPMData.GlobalBuffer:TDateTime

A datetime property defining the global buffer of a task. The global buffer is the number of days that a task can delay without influencing the final project endtime. If the global buffer is 0 then any delay of this bar/task would extends the overall project time. If the global buffer is zero, then the according task/bar is part of the critical path.

TextSettings.Font:TFont

Holds a reference to the font object.

TextSettings.TextLeft:String

Specifies the text that is drawn on the left side of the bar.

TextSettings.TextTop:String

Specifies the text that is drawn on top of the bar.

TextSettings.TextRight:String

Specifies the text that is drawn on the right side of the bar.

TextSettings.TextBottom:String

Specifies the text that is drawn on bottom of the bar.

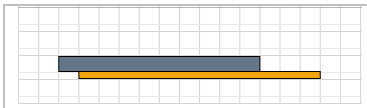
TextSettings.TextCenter:String

Specifies the text that is drawn inside the bar.

Progress bars (TGanttProgressBar)

Progress bars can be used to visualize the actual progress a task has made. For a better comparison – they are directly drawn under the task bar, so they may be applicable for any kinds of target-performance comparisons, as the end-user is able to see the planned time for an activity and the actual time the task has required.

When creating a progress bar you are not restricted in any way – you can create as many progresbars for each row as you want. The amount of progress bars does not depend on the amount of task bars, nor is a progress bar in anyway linked to a task bar. As well as the task bar the progress bar is defined by its StartDate and its EndDate. Use the StartDate:TDateTime and the EndDate:TDateTime properties to define the length and the position of the progress bar.



By default, the progress bar is drawn below the task bar – as shown on the left. However you can specify the vertical position of the progress bar – and the height / ratio of progress bars and task bars and other visual aspects in the ProgressBarView array of the GanttChartView.

GanttChart.GanttChartView.ProgressBarView.Alignment: TProgressBarRowAlignment = (pbaTop, pbaBottom)

Specifies whether progress bars are drawn on top (pbaTop) or below (pbaBottom) the task bars.

GanttChart.GanttChartView.ProgressBarView.HeightMode:TProgressBarHeightMode = (pbmFixedHeight, pbmRatioRowHeight)

Specifies whether the value of the RatioRowHeight property should be handled as absolute pixels (pbmFixedHeight) or as a percentual value of the row height (bpmRatioRowHeight).

GanttChart.GanttChartView. RatioRowHeight:integer

Specifies either the value for the height of the progress bar section within the row - as pixel or as a percentual value of the gantt row.

GanttChart.GanttChartView.ShowGridProgressSplitLine:Boolean

If set to true, a horizontal line will be drawn in the gantt row between the section of the progress bars and the section of the task bars.

GanttChart.GanttChartView.ShowProgressBars:Boolean

Specifies whether progress bars should be shown or not.

Properties

StartDate:TDateTime

The start date of the progress bar. Settings a new value for the StartDate property will reschedule the progress bar.

EndDate:TDateTime

The end date of the progress bar. Setting a new value for the EndDate property will reschedule the progress bar.

BarType:TGanttBarType = (btBaseBar, btTaskBar, btSummaryBar, btProgressBar, btMileStoneBar, btTextBar, btImageBar, btShadowTaskBar, btShadowProgressBar, btPertBar)

For progress bars the BarType returns btProgressBar.

HotTracked:Boolean

Returns true if the progress bar is currently beeing hot tracked.

Selected:Boolean

Returns true if the progress bar is selected.

GUID:TGuid

The gantt chart component assigns to each bar a unique Globally Unique Identifier (GUID) in order to distinguish the bar objects.

Locked:Boolean

If the progress bar is locked, it will be unmodifiable by the end-user. Also locked progress bars are fixed and will not be rescheduled automatically by the component. A small symbol – displaying a lock – is drawn for all locked bars in the upper left corner of the bar.

Height:integer

The height in pixel of a progress bar.

Color:TColor

The color of the progress bar.

Tag:integer

With the help of the Tag property any developer that uses the component can store user defined integer values to each task bar for their own purpose. The Tag property will be ignored by the GanttChart component.

Border:Boolean

Specifies whether a border will be drawn or not.

BorderWidth:Integer

Specifies the width of the border in pixel.

BorderColor:TColor

Specifies the border color.

AutomaticScheduling: Boolean

Read-only property. The value will be determined by the AutomaticScheduling property of the corresponding TGanttRow. When set to false the component will not automatically reschedule the progress bar due any changes to a connected bar.

TextSettings.Font:TFont

Holds a reference to the font object.

TextSettings.TextLeft:String

Specifies the text that is drawn on the left side of the bar.

TextSettings.TextTop:String

Specifies the text that is drawn on top of the bar.

TextSettings.TextRight:String

Specifies the text that is drawn on the right side of the bar.

TextSettings.TextBottom:String

Specifies the text that is drawn on bottom of the bar.

TextSettings.TextCenter:String

Specifies the text that is drawn inside the bar.

Row:Pointer

Holds a reference pointer to the corresponding bar object.

Owner:Pointer

Holds a reference to the corresponding gantt chart object.

Successors:TList<TGanttBaseBar>

Provides a list of all directly connected bars that are successors.

Predecessor:TList<TGanttBaseBar>

Provides a list of all directly connected bars that are predecessors.

AllowVerticalDrag:Boolean

Read-only property. Progress bars can be dragged from one row to another – therefore the property returns true.

AllowHorizontalDraw:Boolean

Read-only property. Progress bars can be dragged horizontally in a row.

AllowMultiRow:Boolean

Read-only property. It is not possible for progress bars to be larger than the height of a single row, so the property returns false for task bars.

AllowConnections:Boolean

Read-only property. Progress bars can be connected to other bar objects – so the property will be true for task bars.

Hint:Boolean

Specifies whether the component should provide a hint for the progress bar.

DefaultHint:Boolean

Specifies whether the DefaultHint or a user-defined hint (see HintTitle and HintDescription) will be used.

HintTitle:String

Specifies the title for the user defined hint.

HintDescription:String

Specifies the description for the user defined hint.

GetDuration:Double

Returns the total (working and non-working) duration of the progress bar.

GetEffectiveDuration:Double

Returns the work time duration for the progress bar.

SetEffectiveDuration(AValue:TDateTime)

Sets the work time duration for the progress bar.

IsPartOfCriticalPath:Boolean

The Gantt Control component includes the calculation of the critical path. The calculation of the critical path determines which tasks are critical and non-critical. When a task is critical any delay of this task will result in a delay of the entire project. The critical path is the sequence of all critical tasks.

Returns true if the progress bar is part of the critical path, otherwise it will return false.

Milestone bars (TGanttMileStoneBar)

Milestones represent a special date in your project for example a deadline. They are visualized as a diamond shape. Milestones do have a fixed size and are not resizable.

Properties

Picture:TPicture

Alternatively, it is possible to display an image instead of the diamond for milestone bars. You can use the Picture property to load/assign an image.

GlyphTag:integer

Could be used by the developer to store an integer value to the milestone bar.

AllowResize:Boolean

Milestone bars are not resizable – therefore the read only property will return false.

GetEffectiveDuration:Double

Returns the work time duration for the milestone bar.

StartDate:TDateTime

The start date of the milestone bar. Assigning a new value for the StartDate property will reschedule the milestone bar.

EndDate:TDateTime

The end date of the milestone bar. Assigning a new value to the EndDate property will reschedule the milestone bar.

BarType:TGanttBarType = (btBaseBar, btTaskBar, btSummaryBar, btProgressBar, btMileStoneBar, btTextBar, btImageBar, btShadowTaskBar, btShadowProgressBar, btPertBar)

For milestone bars the BarType returns btMileStoneBar.

HotTracked:Boolean

Returns true if the milestone bar is currently hot tracked.

Selected:Boolean

Returns true if the milestone bar is selected.

GUID:TGUID

The gantt chart component assigns to each bar a unique Globally Unique Identifier (GUID) in order to distinguish the bar objects.

Locked:Boolean

If the milestone is locked, it will be unmodifiable by the end-user. Also locked milestone bars are fixed and will not be rescheduled automatically by the component. A small symbol – displaying a lock – is drawn for all locked bars in the upper left corner of the bar.

Color:TColor

Specifies the color of the milestone bar.

Tag:integer

With the help of the Tag property any developer that uses the component can store user defined integer values to each milestone bar for their own purpose. The Tag property will be ignored by the GanttChart component.

Border:Boolean

Specifies whether a border will be drawn or not.

BorderWidth:integer

Specifies the width of the border in pixel.

BorderColor:TColor

Specifies the border color.

AutomaticScheduling:Boolean

Read-only property. The value will be determined by the AutomaticScheduling property of the corresponding TGanttRow. When set to false the component will not automatically reschedule the milestone bar due any changes to a connected bar.

Row:Pointer

Holds a reference pointer to the corresponding bar object.

Owner:Pointer

Holds a reference to the corresponding gantt chart object.

Successors:TList<TGanttBaseBar>

Provides a list of all directly connected bars that are successors.

Predecessor:TList<TGanttBaseBar>

Provides a list of all directly connected bars that are predecessors.

AllowVerticalDrag:Boolean

Read-only property. Milestone bars can be dragged from one row to another – therefore the property returns true.

AllowHorizontalDrag:Boolean

Read-only property. Milestone bars can be dragged horizontally in a row.

AllowMultiRow:Boolean

Read-only property. It is not possible for milestone bars to be larger than the height of a single row, so the property returns false for task bars.

AllowConnections:Boolean

Read-only property. Milestone bars can be connected to other bar objects – so the property will be true for task bars.

Hint:Boolean

Specifies whether the component should provide a hint for the milestone bar.

DefaultHint:Boolean

Specifies whether the DefaultHint or a user-defined hint (see HintTitle and HintDescription) will be used.

HintTitle:string

Specifies the title for the user defined hint.

HintDescription:string

Specifies the description for the user defined hint.

Image bars (TGanttImageBar)

If you want to display a graphic within the GanttGraph you can use an image bar object. Image bars are logically linked to a TGanttRow object; however, they may overlap more than one row. Furthermore, you can specify whether the width of the image bar should be defined by a fixed with in pixel or by the end date of the image bar. If you define the width of the image bar by the end date, changing the global time mode of the calendar may stretch the image bar.

The code below shows how to add an image bar object.

```
1.  uses
2.    ... ganttimagebars, ...;
3.
4.  ....
5.
6.  procedure AddImage;
```

```
7.  var
8.   AGanttImageBar : TGanttImageBar;
9.  begin
10.  AGanttImageBar := GanttChart1.AbsoluteRow[2].AddImageBar( Trunc(Now), Trunc(Now) + 5 );
11.  AGanttImageBar.Picture.LoadFromFile( 'C:\SOME\FANCY\Image.png');
12. end;
```

Properties

AllowMultiRow:Boolean

Read-only property. It is possible for image bars to be larger than the height of a single row, so the property returns true.

AllowConnection:Boolean

Read only property. It is not possible for image bars to be connected to other bars, therefore it will return false.

Width:Integer

Specifies the width of the image bar, if the WidthMode property is set to bwmByWidth.

Proportional:Boolean

If set to true, the image bar will keep the height/width ratio while resizing.

StringData1:String

String property that can be used by the developer to store any additional string value to the image bar.

VerticalRowOffset:Integer

Specifies the vertical offset in pixel to the top of the parent row.

WidthMode:TBarWidthMode

Specifies whether the width of the image bar should be calculated by the Width property or calculated by the horizontal position of the EndDate property.

Data:Pointer

Each image bar holds the Data pointer reference that may be used to point to any application specific user data objects.

StartDate:TDateTime

The start date of the image bar. Setting a new value for the StartDate property will reschedule the image bar.

EndDate:TDateTime

The end date of the image bar. Setting a new value for the EndDate property will reschedule the image bar.

BarType:TGanttBarType

For image bars the BarType returns btImageBar.

HotTracked:Boolean

Returns true if the image bar is currently hot tracked.

Selected:Boolean

Returns true if the image bar is selected.

GUID:TGUID

The gantt chart component assigns to each image bar a unique Globally Unique Identifier (GUID) in order to distinguish the bar objects.

Locked:Boolean

If the image bar is locked, it will be unmodifiable by the end-user. Also locked image bars are fixed and will not be rescheduled automatically be the component. A small symbol – displaying a lock – is drawn for all locked bars in the upper left corner of the bar.

Height:integer

The height in pixel of the image bar. When the Proportional property has been set to true, the height of the image bar will be calculated by the component itself.

Tag:integer

With the help of the Tag property any developer that uses the component can store user defined integer values to each image bar for their own purpose. The Tag property will be ignored by the GanttChart component.

Border:Boolean

Specifies whether a border will be drawn or not.

BorderWidth:Integer

Specifies the width of the border in pixel.

BorderColor:TColor

Specifies the color of the border.

TextSettings.Font:TFont

Holds a reference to the font object.

TextSettings.TextLeft:String

Specifies the text that is drawn on the left side of the bar.

TextSettings.TextTop:String

Specifies the text that is drawn on top of the bar.

TextSettings.TextRight:String

Specifies the text that is drawn on the right side of the bar.

TextSettings.TextBottom:String

Specifies the text that is drawn on bottom of the bar.

TextSettings.TextCenter:String

Specifies the text that is drawn inside the bar.

Row:Pointer

Holds a reference pointer to the corresponding bar object.

Owner:Pointer

Holds a reference to the corresponding gantt chart object.

Successors:TList<TGanttBaseBar>

Provides a list of all directly connected bars that are successors.

Predecessor:TList<TGanttBaseBar>

Provides a list of all directly connected bars that are predecessors.

AllowVerticalDrag:Boolean

Read-only property. Image bars can be dragged from one row to another – therefore the property returns true.

AllowHorizontalDrag:Boolean

Read-only property. Image bars can be dragged horizontally in a row.

Hint:Boolean

Specifies whether the component should provide a hint for the milestone bar.

DefaultHint:Boolean

Specifies whether the DefaultHint or a user-defined hint (see HintTitle and HintDescription) will be used.

HintTitle:string

Specifies the title for the user defined hint.

HintDescription:string

Specifies the description for the user defined hint.

Text bars (TGanttTextBar)

Text bars can be used to display larger text information within the gantt chart. A text bar – as well as image bars, can cover more than one row of the chart. When creating a text bar its with is either defined by the Width property or by its EndDate property. To define the vertical dimension of the text bar you have to assign a value to the Height property.

Properties

AllowMultiRow:Boolean

Read-only property. It is possible for text bars to be larger than the height of a single row, so the property returns true.

AllowConnections:Boolean

Read only property. It is not possible for text bars to be connected to other bars, therefore it will return false.

Width:integer

Specifies the width of the text bar, if the WidthMode property is set to bwmByWidth.

Title:String

The title of the text bar.

ShowTitle:Boolean

Specifies whether the text bar should display a title.

Text:string

The text of the text bar.

Font:TFont

The font object used for the text of the text bar.

TitleFont:TFont

The title font object that is used for the title of the text bar.

Transparent:Boolean

When set to true, no solid filled background will be drawn.

VerticalRowOffset:Integer

Specifies the vertical offset in pixel to the top of the parent row.

WidthMode:TBarWidthMode

Specifies whether the width of the text bar should be calculated by the Width property or calculated by the horizontal position of the EndDate property.

Data:Pointer

Each text bar holds the Data pointer reference that may be used to point to any application specific user data objects.

StartDate:TDateTime

The start date of the text bar. Setting a new value for the StartDate property will reschedule the text bar.

EndDate:TDateTime

The end date of the text bar. Setting a new value for the EndDate property will reschedule the text bar.

BarType:TGanttBarType

For text bars the BarType returns btTextBar.

HotTracked:Boolean

Returns true if the text bar is currently hot tracked.

Selected:Boolean

Returns true if the text bar is selected.

GUID:TGUID

The gantt chart component assigns to each bar a unique Globally Unique Identifier (GUID) in order to distinguish the bar objects.

Locked:Boolean

If the text bar is locked, it will be unmodifiable by the end-user. Also locked text bars are fixed and will not be rescheduled automatically by the component. A small symbol – displaying a lock – is drawn for all locked bars in the upper left corner of the bar.

Height:Integer

The height in pixel of the text bar. When the Proportional property has been set to true, the height of the text bar will be calculated by the component itself.

Color:TColor

The background color of the text bar.

Tag:Integer

With the help of the Tag property any developer that uses the component can store user defined integer values to each text bar for their own purpose. The Tag property will be ignored by the GanttChart component.

Border:Boolean

Specifies whether a border will be drawn or not.

BorderWidth:Integer

Specifies the width of the border in pixel.

BorderColor:TColor

Specifies the color of the border.

TextSettings.Font:TFont

Holds a reference to the font object.

TextSettings.TextLeft:String

Specifies the text that is drawn on the left side of the bar.

TextSettings.TextTop:String

Specifies the text that is drawn on top of the bar.

TextSettings.TextRight:String

Specifies the text that is drawn on the right side of the bar.

TextSettings.TextBottom:String

Specifies the text that is drawn on bottom of the bar.

TextSettings.TextCenter:String

Specifies the text that is drawn inside the bar.

Row:Pointer

Holds a reference pointer to the corresponding bar object.

Owner:Pointer

Holds a reference to the corresponding gantt chart object.

Successors:TList<TGanttBaseBar>

Provides a list of all directly connected bars that are successors.

Predecessor:TList<TGanttBaseBar>

Provides a list of all directly connected bars that are predecessors.

AllowVerticalDrag:Boolean

Read-only property. Text bars can be dragged from one row to another – therefore the property returns true.

AllowHorizontalDrag:Boolean

Read-only property. Text bars can be dragged horizontally in a row.

Hint:Boolean

Specifies whether the component should provide a hint for the text bar.

DefaultHint:Boolean

Specifies whether the DefaultHint or a user-defined hint (see HintTitle and HintDescription) will be used.

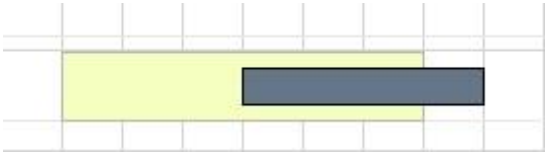
HintTitle:string

Specifies the title for the user defined hint.

HintDescription:string

Specifies the description for the user defined hint.

Shadowbars (TGanttShadowTaskBar, TGanttShadowProgressBar)



Objects of the type TGanttShadowTaskbar and TGanttShadowProgressBar are none interactive gantt bar objects. They can be used to visualize additional information behind any other task bar object. The shadow bar objects have been implemented in the gantt control component in order to be able to display multiple bar objects at the same time range as the task bar object.

Shadow bars do not collide or interfere with other bar objects. When creating shadow bars on a row – it would be recommended to increase the rows height in order to visualize the shadow bar, if it is behind a task bar.

As pointed out before, shadow bars cannot be deleted or modified by the user at runtime.

Properties

AllowResize:Boolean

Read only. Returns false for shadow bars.

AllowVerticalDrag:Boolean

Read only. Returns false for shadow bars.

AllowHorizontalDrag:Boolean

Read only. Returns false for shadow bars.

AllowMultiRow:Boolean

Read only. Returns false for shadow bars.

AllowConnections:Boolean

Read only. Returns false for shadow bars.

Data:Pointer

Each shadow bar holds the Data pointer reference that may be used to point to any application specific user data objects.

StartDate:TDateTime

The start date of the shadow bar. Setting a new value for the StartDate property will reschedule the task bar.

EndDate:TDateTime

The end date of the shadow bar. Setting a new value for the EndDate property will reschedule the task bar.

BarType:TGanttBarType = (btBaseBar, btTaskBar, btSummaryBar, btProgressBar, btMileStoneBar, btTextBar, btImageBar, btShadowTaskBar, btShadowProgressBar, btPertBar)

Returns the corresponding bar type (btShadowTaskBar or btShadowProgressBar) for the shadow bars.

HotTracked:Boolean

Returns true if the shadow bar is currently hot tracked.

Selected:Boolean

Returns true if the shadow bar is selected.

GUID:TGUID

The gantt chart component assigns to each bar a unique Globally Unique Identifier (GUID) in order to distinguish the bar objects.

Locked:Boolean

If the shadow bar is locked, it will be unmodifiable by the end-user. Also locked shadow bars are fixed and will not be rescheduled automatically be the component. A small symbol – displaying a lock – is drawn for all locked bars in the upper left corner of the bar.

Height:integer

Specifies the height of the shadow bar in pixel.

Color:TColor

Specifies the color of the shadow bar.

Tag:integer

With the help of the Tag property any developer that uses the component can store user defined integer values to each shadow bar for their own purpose. The Tag property will be ignored by the GanttChart component.

Border:Boolean

Specifies whether a border will be drawn or not.

BorderWidth:Integer

Specifies the width of the border in pixel.

BorderColor:TColor

Specifies the color of the border.

TextSettings.Font:TFont

Holds a reference to the font object.

TextSettings.TextLeft:String

Specifies the text that is drawn on the left side of the bar.

TextSettings.TextTop:String

Specifies the text that is drawn on top of the bar.

TextSettings.TextRight:String

Specifies the text that is drawn on the right side of the bar.

TextSettings.TextBottom:String

Specifies the text that is drawn on bottom of the bar.

TextSettings.TextCenter:String

Specifies the text that is drawn inside the bar.

Row:Pointer

Holds a reference pointer to the corresponding bar object.

Owner:Pointer

Holds a reference to the corresponding gantt chart object.

Successors:TList<TGanttBaseBar>

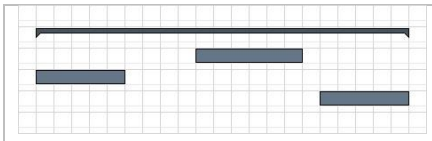
Provides a list of all directly connected bars that are successors.

Predecessor:TList<TGanttBaseBar>

Provides a list of all directly connected bars that are predecessors.

SummaryBar (TGanttSummaryBar)

If one or multiple child rows do own at least a single task bar, progress bar or milestone bar object, a summary bar will be automatically created for the parent row.



The summary bar starts at the earliest start date of all child bars and ends at the latest end of all child bar objects. Changes made to the child bars start date or end date, will force the component to update the summary bar immediately.

It is possible to resize or move the entire summary bar for the end-user. If you delete the last bar object of a summary bar, the summary bar itself will be removed from the component.

Furthermore, the end-user is able to connect a summary bar to another summary bar or a bar object.

Properties

AllowVerticalDrag:Boolean

Read-only property. Summary bars cannot be dragged from one row to another – therefore the property returns false.

AllowHorizontalDrag:Boolean

Read-only property. Summary bars can be dragged horizontally in a row.

Data:Pointer

Each summary bar holds the Data pointer reference that may be used to point to any application specific user data objects.

StartDate:TDateTime

The start date of the summary bar. In order to reschedule an entire summary bar, you may call the RescheduleChilds(ABarStart, ABarEnd: TDateTime; AOpBars:Boolean=False) method.

EndDate:TDateTime

The end date of the summary bar. In order to reschedule an entire summary bar, you may call the RescheduleChilds(ABarStart, ABarEnd: TDateTime; AOpBars:Boolean=False) method.

BarType:TGanttBarType = (btBaseBar, btTaskBar, btSummaryBar, btProgressBar, btMileStoneBar, btTextBar, btImageBar, btShadowTaskBar, btShadowProgressBar, btPertBar)

For summary bars the BarType returns btSummaryBar.

HotTracked:Boolean

Returns true if the summary bar is currently hot tracked.

Selected:Boolean

Returns true if the summary bar is selected.

GUID:TGUID

The gantt chart component assigns to each summary bar a unique Globally Unique Identifier (GUID) in order to distinguish the bar objects.

Color:TColor

Specifies the color of the summary bar.

Tag:Integer

With the help of the Tag property any developer that uses the component can store user defined integer values to each summary bar for their own purpose. The Tag property will be ignored by the GanttChart component.

Border:Integer

Specifies whether a border will be drawn or not.

BorderWidth:Integer

Specifies the width of the border in pixel.

BorderColor:TColor

Specifies the border color.

AutomaticScheduling:Boolean

Read-only property. The value will be determined by the AutomaticScheduling property of the corresponding TGanttRow. When set to false the component will not automatically reschedule the summary bar due any changes made to a connected bar.

TextSettings.Font:TFont

Holds a reference to the font object.

TextSettings.TextLeft:String

Specifies the text that is drawn on the left side of the bar.

TextSettings.TextTop:String

Specifies the text that is drawn on top of the bar.

TextSettings.TextRight:String

Specifies the text that is drawn on the right side of the bar.

TextSettings.TextBottom:String

Specifies the text that is drawn on bottom of the bar.

TextSettings.TextCenter:String

Specifies the text that is drawn inside the bar.

Row:Pointer

Holds a reference pointer to the corresponding bar object.

Owner:Pointer

Holds a reference to the corresponding gantt chart object.

Successors:TList<TGanttBaseBar>

Provides a list of all directly connected bars that are successors.

Predecessor:TList<TGanttBaseBar>

Provides a list of all directly connected bars that are predecessors.

AllowMultiRow:Boolean

Read-only property. It is not possible for summary bars to be larger than the height of a single row, so the property returns false for summary bars.

AllowConnections:Boolean

Read-only property. Task bars can be connected to other bar objects – so the property will be true for task bars.

Hint:Boolean

Specifies whether the component should provide a hint for the summary bar.

DefaultHint:Boolean

Specifies whether the DefaultHint or a user-defined hint (see HintTitle and HintDescription) will be used.

HintTitle:string

Specifies the title for the user defined hint.

HintDescription:string

Specifies the description for the user defined hint.

Selection (TGanttSelection)

When integrating the gantt chart component in your application, it is highly common that it is required to know what bars or rows the user has selected or to select some bars by the application by code. The gantt chart component provides the GanttChart.Selection object to handle and group methods and functions refering to the selection of objects. Below is a list of properties and methods of the GanttChart.Selection object.

The component provides several lists of the selected objects. For rows the component distinguishes wheter a row is fully selected – all cells of a row are selected – or partially selected – only some cells of the rows – are selected. All fully selected rows are also part of the SelectedRowsPartial list.

SelectedCells : TList<TGanttCell>

Contains all selected cells.

SelectedRows : TList<TGanttRow>

Contains all fully selected rows. A row will only be part of the SelectedRows list, when all cells of the rows are selected.

SelectedRowsPartial : TList<TGanttRow>

Contains all partial selected und fully selected rows. A partial selected row is a row that has at least a single selected cell.

SelectedBars : TList<TGanttBaseBar>

Contains all selected bars, without pert bars.

SelectedBarsAndPertBars : TList<TGanttBaseBar>

Contains all selected bars and all selected part bars.

SelectedPertBars : TList<TGanttBaseBar>

Contains all selected pert bars.

The component uses generic list in order to return the selected objects. The code sample below shows, how you can iterate easily through the list of the selected cells and set the cell color for each selected cell to clBlack.

```
1.  procedure TForm1.IterateThroughSelectedCells;
2.  var
3.   aGanttCell : TGanttCell;
4.  begin
5.   for aGanttCell in GanttChart1.Selection.SelectedCells do
6.    begin
7.      aGanttCell.Color := clBlack;
8.    end;
9.  end;
```

Below is a list of methods of the Selection object that allows the developer to manipulate the selection of objects:

Deselect

Deselect all cells and rows.

DeselectBars

Deselect all bars.

SelectAll

Select all rows of the gantt chart.

SelectAllBars(ABarType:TGanttBarTypeSet)

Select all bars of the specified bar type. TGanttBarTypeSet is a set of the TGanttBarType = (btBaseBar, btTaskBar, btSummaryBar, btProgressBar, btMileStoneBar, btTextBar, btImageBar, btShadowTaskBar, btShadowProgressBar, btPertBar).

There are some predefined BarTypeSets defined that you may use.

TGanttBarTypeSet	Elements
btScheduleBars	[btTaskBar, btProgressBar, btMileStoneBar]
btNonScheduleBars	[btTextBar, btImageBar, btShadowTaskBar, btShadowProgressBar]
btShadowBars	[btShadowTaskBar, btShadowProgressBar]
btAllBars	[...all bar types...]

The code sample below selects all taskbars and progress bars:

```
1.  ...
2.  GanttChart1.Selection.SelectAllBars([btTaskBar, btProgressBar]);
3.  ...
```

SelectRange(AFromRow, AFromColumn, AToRow, AToColumn : integer; AddSelectionMode:TSelectionAddMode)

Selects a range of cells, specified by the parameters AFromRow, AFromColumn, AToRow and AToColumn. The parameter AddSelectionMode defines how the selected range handles the prior selected objects. The following table shows all values for the TSelectionAddMode type:

TSelectionAddMode	Description
omDefaultSelect	Replaces the prior selection with new selected objects.
omAddToSelection	Adds the selected objects to the selected Ranges array.
omMergeSelection	The component will integrate the selected range into the previously selected range.

SelectRange(AFromRow, AToRow : integer; AddSelectionMode:TSelectionAddMode)

Selects a range of rows, specified by the parameters AFromRow and AToRow.

SelectColumn(Index:integer; AddSelectionMode:TSelectionAddMode)

Selects a column, specified by the parameter Index.

SelectRow(Index:integer; AddSelectionMode:TSelectionAddMode)

Selects a row, specified by the parameter Index. The Index refers to the AbsoluteIndex of a row.

SelectCell(AGanttCell:TGanttCell; AddSelectionMode:TSelectionAddMode)

Selects a cell.

SelectRow(ACell:TGanttCell; AddSelectionMode:TSelectionAddMode)

Selects the row of a given cell.

SelectRange(AFromRow, AFromColumn, AToRow, AToColumn : integer)

Selects a range of cells, specified by the parameters AFromRow, AFromColumn, AToRow and AToColumn.

SelectColumn(Index:integer)

Selects a column specified by the parameter Index.

SelectRow(Index:integer)

Selects a row specified by the parameter index. The Index refers to the AbsoluteIndex of a row.

SelectCell(AGanttCell:TGanttCell)

Selects a cell.

GetFirstSelectedRow(AAllowPartial:Boolean):TGanttRow

Returns the selected row, with the smallest index – therefore the first selected row. AAllowPartial determines whether only fully selected rows are handled or also partial selected rows.

GetFirstSelectedCell:TGanttCell

Iterates through all selection cells and returns the cell of the topmost selected row and the leftmost selected cell.

GetTopLeftSelectedCell:TGanttCell

Returns the topmost and leftmost selected cell.

GetBottomRightSelectedCell:TGanttCell

Returns the rightmost and bottommost selected cell.

GetFirstSelectedBar:TGanttBaseBar

Returns the first selected bar.

HasSelectionRange:Boolean

Returns true if there is a range of cells selected.

Selected(AGanttRow:TGanttRow):Boolean

Returns true if the gantt row specified by the parameter AGanttRow is fully selected.

Selected(AGanttCell:TGanttCell):Boolean

Returns true, if the specified cell AGanttCell is selected.

Selected(AGanttColumn:TGanttColumn):Boolean

Returns true, if the specified column AGanttColumn has been selected.

SelectedPartial(AGanttRow:TGanttRow):Boolean

Returns true, if the specified row AGanttRow has been partially selected.

SelectedPartial(AGanttColumn:TGanttColumn):Boolean

Returns true, if the column has been selected partial.

SelectionMoveDown(AddToSelection:Boolean)

Moves the selection down one step.

SelectionMoveUp(AddToSelection:Boolean)

Moves the selection up one step.

SelectionMoveRight(AddToSelection:Boolean)

Moves the selection to the right one step.

SelectionMoveLeft(AddToSelection:Boolean)

Moves the selection to the left one step.

SelectionMoveFirstColumn

Moves the selection to the first column.

SelectionMoveLastColumn

Moves the selection to the last column.

SelectionMovePageUp

Moves the selection up one page.

SelectionMovePageDown

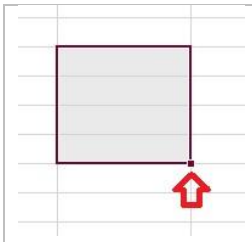
Moves the selection down one page.

SelectedBarsBarType(ABarType:TGanttBarTypeSet): TList<TGanttBaseBar>

Returns the selected bars specified by their type.

Options Selection (TGanttOptionsSelection)

By default the end user is able to select various cells, providing the option to drag the SelectionFrame as shown below:



The user is able to manipulate the range of the selected cells by dragging the small square in the lower right corner.

If you click on a column header you will select all cells of the column. Accordingly if the user clicks a row indicator cell, all cells of this row will be selected. If you click on the indicator columns header all cells of the gantt chart will be selected.

The TGanttOptionsSelection object holds severall properties that makes it possible to customize the selection behaviour of the component.

GanttChart.OptionsSelection.CellSelect:Boolean

If set to true the end user will be able to select single cells, otherwhise only entire rows will be selectable.

GanttChart.OptionsSelection.HeaderColumnQuickSelect:Boolean

If set to true the end user will be able to select cells of multiple columns, as he keeps the left mouse button pressed and moves (drags) the mouse cursor over some column headers.

GanttChart.OptionsSelection.IndicatorRowQuickSelect:Boolean

If set to true the end user will be able to select cells of multiple rows, as he keeps the left mouse button pressed and moves (drags) the mouse cursor over some indicator cells.

GanttChart.OptionsSelection.MultiSelect:Boolean

If set to true, the end-user will be able to select multiple objects. Depending on the GanttChart.OptionsSelection.CellSelect property, he can either select multiple cells or multipl rows.

GanttChart.OptionsSelection.ShowSelectionFrame:Boolean

If set to true, the SelectionFrame is shown.

Undo (TGanttUndoRedoStack)

The component is fully undo- and redoable by default. Whenever an operation has manipulated data, properties of objects or the structure a complete copy of the current component will be pushed on an undo stack. You may call GanttChart.UndoRedoStack.Undo(); to restore the state of the component before the changes have been applied. Furthermore you may be able to set the component to any prior stored state of the undo stack immediately.

If you have undone any changes, you may redo the changes.

By default the undo- and redo stack only stores the last 20 states of the component. Please keep in mind that a single entry in the undo stack holds the entire copy of the component including all elements and their properties. So it might be possible that a scenario where the gantt chart does have a large amount of rows or bars, may require some more memory. You can specifiy the undo and redo stack size. Also it is recommended to disable the TGanttUndoRedoStack if you require the best performance for the component.

Below are the properties and methods of the TGanttUndoRedoStack of the component:

GanttChart.UndoRedoStack.MaxStackSize:Integer

The size of the undo and redo stack. Specifies how many actions can be undone or redone.

GanttChart.UndoRedoStack.Enabled:Boolean

Specifies whether the undo- and redo mechanism should be active or not. If the application that integrates the gantt chart component does not provide and uses the inbuildt undo and redo methods, you can disable the UndoRedoStack for a better performance.

GanttChart.UndoRedoStack.Clear

Clears the entire undo- and redo stack.

GanttChart.UndoRedoStack.BeginGroup

If you want to call multiple data sensitive operations at once, and you want the user to undo the changes at once, you have to group the operations between a BeginGroup and EndGroup section.

GanttChart.UndoRedoStack.EndGroup

If you want to call multiple data sensitive operations at once, and you want the user to undo the changes at once, you have to group the operations between a BeginGroup and EndGroup section.

GanttChart.UndoRedoStack.BlockUndo

If you want to make changes to the data of the component, and don’t want to make them undoable you have to group the operation(s) between a BlockUndo() and EndBlockUndo() section.

GanttChart.UndoRedoStack.EndBlockUndo

EndBlockUndo() marks the end of an undo blocking section, started with BlockUndo().

GanttChart.UndoRedoStack.EndBlockUndoRestoreOldValue

EndBlockUndoRestoreOldValue() marks the end of an undo blocking section, started with BlockUndo(). If you use EndBlockUndo() in a recursive call stack, it is recommended to use EndBlockUndoRestoreOldValue() instead of EndBlockUndo().

GanttChart.UndoRedoStack.PushUndo(ACaption:String)

Stores the current state of the component in the undo stack of the component. Call PushUndo(ACaption:String) before you apply any changes to the component, so that it is possible to undo the changes and restore the prior state of the component. Caption describes the action you want to apply.

GanttChart.UndoRedoStack.PushRedo(ACaption:String)

Pushes the current state of the component on the RedoStack.

GanttChart.UndoRedoStack.Undo

Reverts the last command and restores the state of the component before the last operations have made changes to the component.

GanttChart.UndoRedoStack.Undo(AIndex:integer)

Reverts the last commandos and set the component to a prior state. AIndex specifies the index in the UndoStack that the component should be restored to.

GanttChart.UndoRedoStack.Redo

Reverts the effect of the last undo commando.

GanttChart.UndoRedoStack.Redo(AIndex:integer)

Reverts the effect of a last undo commando. AIndex specifies the index in the ReDoStack that the component will be restored to.

GanttChartView (TGanttChartView)

The GanttChartView record holds multiple basic options refering to visual aspects, the time mode, the scroll range and the appearance of the progress bars. The time mode describes the time span the time scale will be segmented into.

TimeMode: TGanttChartTimeMode = (tmHour, tmDay, tmWeek, tmMonth, tmQuarter, tmYear)

The global time mode of the gantt chart. Whenever you change the time mode of the component – the time mode of single time scales are readapted. Example: If you create three time scales and set their time modes to tmWeek, tmMonth and tmYear and change the time mode of the component from tmWeek to tmDay, the time modes of the time scales will change to tmDay, tmWeek and tmMonth.

The following table shows the possible values for the time mode:

TimeMode	Time unit
tmHour	Hours
tmDay	Days
tmWeek	Weeks
tmMonth	Months
tmQuarter	Quarters
tmYear	Years

GanttCellWidth:integer

GanttCellWidth defines the width of the smallest time unit segment in pixel.

The image on the left side demonstrates different gantt cell widths. The gantt chart on the left side, does have a GanttCellWidth of 20 pixel and the gantt chart on the right side does have a GanttCellWidth of 40 pixel.

StartDate:TDateTime

The start date of the gantt chart view. Assigns a new start date in order to scroll the gantt chart horizontally.

EndDate:TDateTime (ReadOnly)

Depends on the GanttCellWidth and the width of the gantt chart itself. Use the read only EndDate property to retrieve the start date of the (fully) last visible time segment that is displayed in the view range.

ScrollRangeStart:TDateTime

By default the end user is able to scroll horizontally without any boundaries throughout the gantt chart view. However you can set a fixed scroll range. ScrollRangeStart specifies the start of the scroll range.

ScrollRangeEnd:TDateTime

By default the end user is able to scroll horizontally without any boundaries throughout the gantt chart view. However you can set a fixed scroll range. ScrollRangeEnd specifies the end of the scroll range.

UseFixScrollRange:Boolean

If you want to set up a fixed scroll range, set UseFixScrollRange to true. Horizontally scrolling will then only be possible within the specified scroll range.

SmoothScrolling:Boolean

If set to true, the component scrolls a little smoother horizontally.

ShowTimeScaleAtBottom:Boolean

If set to true, the defined time scales will be shown below the last gantt row.

ProgressBarView.Alignment: TProgressBarRowAlignment = (pbaTop, pbaBottom)

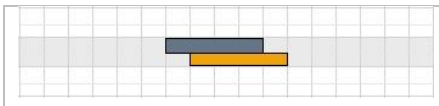
Specifies the position of progress bars within a gantt row.

ProgressBarView.Height:integer

Specifies the height in pixel of the section that displays the progress bars within a row. Requires that the ProgressBarView.HeightMode has been set to pbmFixedHeight.

ProgressBarView.HeightMode:TProgressBarHeightMode = (pbmFixedHeight, pbmRatioRowHeight)

Specifies whether the absolute Height or the RatioRowHeight value will be used to determine the height of progress bar section within a row.



The image shows a gantt row, where progress bars are shown below task bars.

ProgressBarView.RatioRowHeight:integer

Specifies the ratio (0-100) between the height of the progress bar section and the task bar section. Requires that the ProgressBarView.HeightMode has been set to pbmRatioRowHeight.

ProgressBarView.ShowGridProgressSplitLine:Boolean

If set to true a small horizontal line will be drawn between the progress bar section and the task bar section.

ProgressBarView.ShowProgressBars:Boolean

Specifies whether progress bars are shown within a gantt row.

DragNewBarType:TCreateNewBarType

The end user is able to create new bars by holding the left mouse button down and dragging a new object into a row of the gantt chart. By default all new bars that will be inserted this way (by the user) are task bars. You can specify a new value for the DrawNewBarType property in order to enable the end user to create another bar object. The following table summarizes all possible bar types and their corresponding values:

TCreateNewBarType	Description
nbTaskBar	Task bars

nbProgressBar	Progress bars
nbMileStoneBar	Milestones
nbTextBar	Text bars
nbImageBar	Image bars
nbTaskBarorProgressBar	Task bars and Progress Bars. Based on the y-position of the mouse cursor either task bars or progress bars will be created.

OptionsBehaviour (TGanttOptionsBehaviour)

The OptionsBehaviour record holds multiple properties that defines how the end-user can interact with the gantt chart component and what operations can be triggered or not.

OptionsBehaviour.ColumnResizing:Boolean

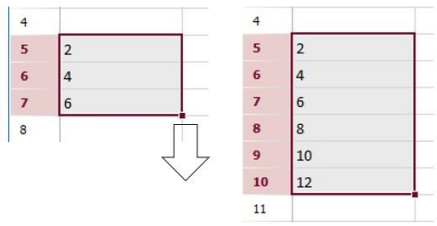
Specifies whether the end-user is able to resize the width of columns.

OptionsBehaviour.RowResizing:Boolean

Specifies whether the end user is able to resize the height of rows.

OptionsBehaviour.AutoFillCells:Boolean

If the user has selected a range of cells and expands the range vertically the component tries to auto fill the content for the cells, if AutoFillCells has been set to true.

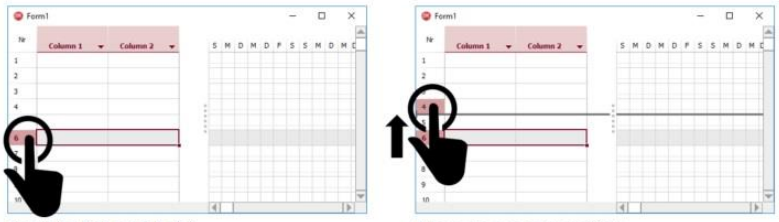


Please have a look at the example on the left: If the range of the selected cells do have the sequence of '2', '4', '6' as cell values and the user expands the range, the component automatically fills in the values '8', '10', 12'.

The component tries to detect linear series for numeric and date values, day and month names. If no serie could be detected it will copy the cell values for the new selected cells.

OptionsBehaviour.RowDragging:Boolean

The end-user is able to reorder rows within its level of hierarchy. In order to drag a row to a new position. You have to doubleclick in the indicator cell, hold the mouse button down and move the row to a new position.



OptionsBehaviour.ResizeMultipleRows:Boolean

Specifies whether the row height of all selected rows can be resized at once.

OptionsBehaviour.TreeCellEditOnTyping:Boolean

Specifies whether a key stroke will immediately set a cell in edit mode.

OptionsBehaviour.TreeCellEditOnEnter:Boolean

Specifies whether a cell will be set in edit mode when the end-user presses the [ENTER] key.

OptionsBehaviour.AddBars:Boolean

Specifies whether end-users are allowed to add new bar objects.

OptionsBehaviour.DeleteBars:Boolean

Specifies whether end-users are allowed to delete bar objects.

OptionsBehaviour.DeleteRows:Boolean

Specifies whether end-users are allowed to delete rows by selecting a row and pressing the [DEL] key.

OptionsBehaviour.InsertRows:Boolean

Specifies whether end-users are allowed to insert rows by selecting a row and pressing the [INSERT] key.

OptionsBehaviour.ModifyBars:Boolean

Specifies whether end-users are allowed to modify and reschedule bars.

OptionsBehaviour.AddConnections:Boolean

Specifies whether end-users are allowed to add connections between bar objects.

OptionsBehaviour.DeleteConnections:Boolean

Specifies whether end-users are allowed to delete connections between bar objects.

OptionsBehaviour.DeleteCellContent:Boolean

Specifies whether end-users are allowed to delete the content of selected cells by pressing then [DEL] key.

OptionsBehaviour.ApplyRowBestFitOnIndicatorDoubleClick:Boolean

Specifies whether end-users are allowed to apply a new best-fit heights for rows, by double clicking at the row resizer in the indicator cell.

OptionsSchedule (TGanttOptionsSchedule)

OptionsSchedule holds some properties regarding the manipulation and rescheduling of bars.

SnapToTime : TSnapToTimeMode = (sttHours, sttDays)

When the SnapToTime property has been set to sttDays the bars will be automatically snapped to the start and the end of a day. When set to sttHous bars can be reassigned to any time.

UpdateCriticalPath:Boolean

Specifies whether the critical path should be updated.

ScheduleBarsToWorkingTimes:Boolean

When a bar is rescheduled and ScheduleBarsToWorking has been set to true, the component will automatically reassign bars to always start on working times.

OptionsView (TGanttOptionsView)

The OptionsView record holds multiple options and properties regarding the visual appearance of the gantt chart and its elements.

GridLineColor : TColor

The basic color for the grid lines. Grid lines are drawn between the cells and the day units of the gantt chart.

GridProgressSplitLineColor : TColor

The color for the progress bar split line within a gantt row.

GridLineBorderColor : TColor

The color for the outer border line of the gantt chart component.

DrawGridLinesSoft : Boolean

When set to true the component uses GridLineSoftColor to draw a color gradient for the start and the end of the grid lines to apply a smoother look.

GridLineSoftColor : TColor

GridLineSoftColor is used as color value when drawing a color gradient for the start and the end of the grid lines. Requires to set DrawGridLinesSoft to true.

SelectionRangeColor : TColor

The color that is used for drawing the border of the selected cell range and the focused column caption and focused row numbers.

DefaultTaskBarOptions : TGanttDefaultTaskBarOptions

Specifies the visual settings for new created task bars. Requires DefaultTaskBarOptions.ApplyOptions has been set to true.

DefaultProgressBarOptions : TGanttDefaultProgressBarOptions

Specifies the visual settings for new created progress bars. Requires DefaultProgressBarOptions.ApplyOptions has been set to true.

DefaultMileStoneBarOptions : TGanttDefaultMileStoneBarOptions

Specifies the visual settings for new created milestone bars. Requires DefaultMileStoneBarOptions.ApplyOptions has been set to true.

DefaultTextBarOptions : TGanttDefaultTextBarOptions

Specifies the visual settings for new created text bars. Requires DefaultTextBarOptions.ApplyOptions has been set to true.

DefaultImageBarOptions : TGanttDefaultImageBarOptions

Specifies the visual settings for new created image bars. Requires DefaultImageBarOptions.ApplyOptions has been set to true.

DefaultSummaryBarOptions : TGanttDefaultSummaryBarOptions

Specifies the visual settings for new created summary bars. Requires DefaultSummaryBarOptions.ApplyOptions has been set to true.

DefaultCellOptions : TGanttDefaultCellOptions

Specifies the default color when a cell is created.

TaskBarMaxHeight : Boolean

Specifies the maximal height in pixel for task bars.

ProgressBarMaxHeight : Boolean

Specifies the maximal height in pixel for progress bars.

CriticalPathOptions : TCriticalPathOptions

Holds options for the appearance of the critical path.

ShowBarButtons : Boolean

Specifies whether quick buttons on selected bars should be drawn or not.

NoDataToDisplay : String

Specifies the string that is displayed when the gantt chart component does not have any rows.

DefaultRowHeight : integer

Specifies the default row height.

ColorHot : TColor

Specifies the hot tracked color. The color is used to highlight objects that are currently under the mouse cursor.

ColorHotDarker : TColor

A secondary color for the ColorHot property.

ColorSelected : TColor

A Color that is used to draw the background of column headers for selected columns and the background of indicator cells for selected rows.

ShowVerticalGridLines : Boolean

Specifies whether a vertical grid line should be drawn between each time unit of the gantt chart.

Calendar (TGanttCalendar)

The gantt chart control provides a calendar module that can be used to define and display special (working and non-working) date exceptions, such as weekends or holidays. The calculation of the duration and the effective duration (working times) of bars, is based on the definition of date exceptions of the calendar.

The component holds several calendar modules. Date Exceptions defined in the base calendar of the component (GanttChart.BaseCalendar) are applied to all rows by default. However, it is possible to apply a unique RowCalendar to a single row or a set of rows. In this case, the RowCalendar has a higher priority and the BaseCalendar will not be interpreted. This provides the possibility to have applied different date exceptions to different rows.

A TGanttCalendar object holds an array of Categories (TGanttCalendar.Categorie[]). Each DateCategory provides access to the DateExceptions and holds methods to add and delete DateExceptions.

Please note, that a single calendar cannot hold more than 255 date exceptions!

There are different date exception types, as it could be seen in the following table:

DateException	Description
TDayException	Define working or non-working hours for a day.
TWeekException	Define working or non-working days or DayExceptions for a week.
TMonthException	Define working or non-working days, DayExceptions for a month. Defines working or non-working weeks or WeekExceptions for a month.
TYearException	Defines working or non-working days or DayExceptions for a given date. Defines working or non working weeks, WeekException or working/non-working months.
TRecurringException	Provides the possibility to define patterns for recurring DateExceptions.

A very import aspect of Date Exceptions is, that it is possible to define severall DateExceptions within other DateExceptions. For example, you can define different MonthExceptions within a YearException – and the MonthException holds 4 different WeekExceptios that are defined by DayExceptions. By structuring and using recurring exceptions, it is possible to define complex scenarios.

The following code shows how to add a new DateCategory to the BaseCalendar and add a new YearException to it and declare the 01.01.0217 as a non-working time.

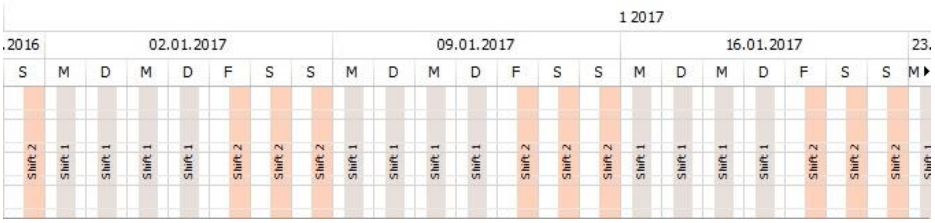
```
1.  GanttChart1.BaseCalendar.Add.AddYearException.DeclareDate('Special date', StrToDate('01.01.2017')
    , StrToDate('01.01.2017'), False);
```

A more complex scenario could look like the following sample of code:

```
1.  procedure TForm1.AddExceptions;
2.  var
3.   WeekException1,
4.   WeekException2     : TWeekException;
5.   RecurringException : TRecurringException;
6.
7.   DayException1,
8.   DayException2 : TDayException;
9.  begin
10.  WeekException1 := TWeekException.Create( GanttChart1.BaseCalendar );
11.  DayException1 := WeekException1.AddDayException_DayOfTheWeek('Shift 1', 1, 4);
12.  DayException1.DeclareHour('Shift 1', 7, 17, true);
13.
14.  DayException1.ShowCaption := true;
15.  DayException1.Font.Color := clBlack;
16.
17.  WeekException2 := TWeekException.Create( GanttChart1.BaseCalendar );
18.  DayException2 := WeekException2.AddDayException_DayOfTheWeek('Shift 2', 5, 7);
```

```
19.
20.   DayException2.DeclareHour('Shift 2', 12, 23, true);
21.   DayException2.ShowCaption := true;
22.   DayException2.Font.Color := cLBlack;
23.   DayException2.Color := $00BBD1FF;
24.
25.   // Add the Exeception to the Category
26.   with GanttChart1.BaseCalendar.Add do
27.   begin
28.    Name := 'DateCategory1';
29.    AddException( WeekException1 );
30.    AddException( WeekException2 );
31.   end;
32. end;
```

The code above will add two different shifts to the gantt chart as shown below:



The following chapters describe the different DateExceptions.

DayExceptions (TDayException)

The day exception declares hourly working and non-working times.

Properties and Methods

DeclareTime(ACaption:String; AStartTime, AEndTime : TTime; AWorking : Boolean)

Defines a working or non-working time period for a day – specified by the AStartTime and AEndTime parameter. The label defined by ACaption will be shown in the gantt chart if the flag ShowCaption has been set to true and if there is enough space to be shown.

DeclareHour(ACaption:String; AHourStart, AHourEnd : integer; AWorking : Boolean)

Defines a working or non-working time period for a day – specified by the AHourStart and AHourEnd parameter. The label defined by ACaption will be shown in the gantt chart if the flag ShowCaption has been set to true and if there is enough space to be shown.

ExceptionListCount:integer

Returns the total amount of exceptions defined by the DateException.

ExceptionList(ID:integer): TExceptionListElement

Provides access to an exception element.

GetCaption(const ADateTime:TDateTime):String

Returns the caption for a given time.

IsWorking(const ADateTime:TDateTime; const ADefault:Boolean):Boolean

Returns true, if the time specified by ADateTime is defined as working time.

GetException(const ADateTime:TDateTime): TBaseDateException

Returns the TBaseDateException element for a given time, otherwise it will return NIL.

Color:TColor

Defines the color for the exception.

Font:TFont

Holds a reference to the font object – that is used when drawing the caption.

ShowCaption:Boolean

Specifies whether the caption should be visible or not.

Visible:Boolean

Defines whether the DateException is visible or not. When set to false, the DateException still defines working and non-working times for the gantt chart component.

WeekException (TWeekException)

Properties and Methods

DeclareDayOfTheWeek(ACaption:String; AFromDayOfTheWeek, AToDayOfTheWeek : integer; AWorking : Boolean)

Defines working and non-working days within a week, specified by the AFromDayOfTheWeek and AToDayOfTheWeek parameter. For the parameters AFromDayOfTheWeek and AToDayOfTheWeek, the value '1' represents Monday and '7' represents 'Sunday'. The label defined by ACaption will be shown in the gantt chart if the flag ShowCaption has been set to true and if there is enough space to be shown.

DeclareDayOfTheWeek(ACaption:String; AFromDayOfTheWeek, AToDayOfTheWeek : integer; ADayException: TDayException)

Sets working and non-working hours, defined by a DayException for a single day, or a range of days within a week. For the parameters AFromDayOfTheWeek and AToDayOfTheWeek, the value '1' represents Monday and '7' represents 'Sunday'. The label defined by ACaption will be shown in the gantt chart if the flag ShowCaption has been set to true and if there is enough space to be shown.

AddDayException_DayOfTheWeek(ACaption:String; AFromDayOfTheWeek, AToDayOfTheWeek : integer):TDayException

Creates a new DayException for the WeekException and returns a reference to it.

ExceptionListCount:integer

Returns the total amount of exceptions defined by the DateException.

ExceptionList(ID:integer): TExceptionListElement

Provides access to an exception element.

GetCaption(const ADateTime:TDateTime):String

Returns the caption for a given time.

IsWorking(const ADateTime:TDateTime; const ADefault:Boolean):Boolean

Returns true, if the time specified by ADateTime is defined as working time.

GetException(const ADateTime:TDateTime): TBaseDateException

Returns the TBaseDateException element for a given time, otherwise it will return NIL.

Color:TColor

Defines the color for the exception.

Font:TFont

Holds a reference to the font object – that is used when drawing the caption.

ShowCaption:Boolean

Specifies whether the caption should be visible or not.

Visible:Boolean

Defines whether the DateException is visible or not. When set to false, the DateException still defines working and non-working times for the gantt chart component.

MonthExceptions (TMonthException)

Properties and Methods

DeclareDayOfTheMonth(ACaption:String; AFromDayOfTheMonth, AToDayOfTheMonth : integer; AWorking : Boolean)

Defines working and non-working days within a month, specified by the AFromDayOfTheMonth and AToDayOfTheMonth paremeter. The label defined by ACaption will be shown in the gantt chart, if the flag ShowCaption has been set to true and if there is enough space to be shown.

DeclareDayOfTheMonth(ACaption:String; AFromDayOfTheMonth, AToDayOfTheMonth : integer; ADayException: TDayException)

Sets working and non-working hours defined by a DayException for a single day, or a range of days within a month. The label defined by ACaption will be shown in the gantt chart, if the flag ShowCaption has been set to true and if there is enough space to be shown.

DeclareWeekOfTheMonth(ACaption:String; AFromWeekOfTheMonth, AToWeekOfTheMonth : integer; AWorking : Boolean)

Defines working and non-working weeks within a month, specified by the AFromWeekOfTheMonth and AToWeekOfTheMonth parameter. The label defined by ACaption will be shown in the gantt chart, if the flag ShowCaption has been set to true and if there is enough space to be shown.

DeclareWeekOfTheMonth(ACaption:String; AFromWeekOfTheMonth, AToWeekOfTheMonth : integer; ADayException: TDayException)

Sets working and non-working hours defined by a DayException for weeks within a month.

AddDayException_DayOfTheMonth(ACaption:String;AFromDayOfTheMonth, AToDayOfTheMonth : integer):TDayException

Adds a DayException for a single day or a range of days within a month and returns a reference to it.

AddDayException_WeekOfTheMonth(ACaption:String;AFromWeekOfTheMonth, AToWeekOfTheMonth : integer):TDayException

Adds a DayException for weeks within a month and returns a reference to it.

ExceptionListCount:integer

Returns the total amount of exceptions defined by the DateException.

ExceptionList(ID:integer): TExceptionListElement

Provides access to an exception element.

GetCaption(const ADateTime:TDateTime):String

Returns the caption for a given time.

IsWorking(const ADateTime:TDateTime; const ADefault:Boolean):Boolean

Returns true, if the time specified by ADateTime is defined as working time.

GetException(const ADateTime:TDateTime): TBaseDateException

Returns the TBaseDateException element for a given time, otherwise it will return NIL.

Color:TColor

Defines the color for the exception.

Font:TFont

Holds a reference to the font object – that is used when drawing the caption.

ShowCaption:Boolean

Specifies whether the caption should be visible or not.

Visible:Boolean

Defines whether the DateException is visible or not. When set to false, the DateException still defines working and non-working times for the gantt chart component.

YearException (TYearException)

Properties and Methods

DeclareDate(ACaption:String; AFromDate, AToDate : TDateTime; AWorking : Boolean)

Defines an exception for a time period, defined by the parameters AFromDate and AToDate.

DeclareDate(ACaption:String; AFromDate, AToDate : TDateTime; ADayException: TDayException)

Sets working and non-working hours, defined by a DayException for a time period, specified by the parameters AFromDate and AToDate.

DeclareDayOfTheYear(ACaption:String; AFromDayOfTheYear, AToDayOfTheYear : integer; AWorking : Boolean)

Defines a range of days withing a year as working or non-working.

DeclareDayOfTheYear(ACaption:String; AFromDayOfTheYear, AToDayOfTheYear : integer; ADayException: TDayException)

Sets working and non-working hours, defined by a DayException for a range of days withing a year.

DeclareDate(ACaption:String; AFromDate,AToDate: TDateTime; AWeekException: TWeekException)

Sets an existing WeekException for a time period.

DeclareWeekOfTheYear(ACaption:String; AFromWeekOfTheYear, AToWeekOfTheYear : integer; AWorking : Boolean)

Defines a range of weeks of the year as working or non working.

DeclareWeekOfTheYear(ACaption:String; AFromWeekOfTheYear, AToWeekOfTheYear : integer; ADayException: TDayException)

Sets an existing DayException for a range of weeks of a year.

DeclareWeekOfTheYear(ACaption:String; AFromWeekOfTheYear, AToWeekOfTheYear : integer; AWeekException: TWeekException)

Sets an existing WeekException for a range of weeks within a year.

AddDayException_WeekOfTheYear(ACaption:String;AFromWeekOfTheYear, AToWeekOfTheYear : integer):TDayException

Creates a new DayException for a range of weeks and returns a reference to it.

AddWeekException_WeekOfTheYear(ACaption:String;AFromWeekOfTheYear, AToWeekOfTheYear : integer):TWeekException

Creates a new WeekException for a range of weeks and returns a reference to it.

DeclareMonthOfTheYear(ACaption:String; AFromMonthOfTheYear, AToMonthOfTheYear : integer; AWorking : Boolean)

Defines working and non-working times for a range of months of a year.

DeclareMonthOfTheYear(ACaption:String; AFromMonthOfTheYear, AToMonthOfTheYear : integer; ADayException: TDayException)

Sets an existing DayException for a range of months within a year.

DeclareMonthOfTheYear(ACaption:String; AFromMonthOfTheYear, AToMonthOfTheYear : integer; AWeekException: TWeekException)

Sets an existing WeekException for a range of months within a year.

DeclareMonthOfTheYear(ACaption:String; AFromMonthOfTheYear, AToMonthOfTheYear : integer; AMonthException: TMonthException)

Sets an existing MonthException for a range of months within a year.

AddDayException_MonthOfTheYear(ACaption:String;AFromMonthOfTheYear, AToMonthOfTheYear : integer): TDayException

Adds a new DayException for a range of months within a year and returns a reference to it.

AddWeekException_MonthOfTheYear(ACaption:String;AFromMonthOfTheYear, AToMonthOfTheYear : integer): TWeekException

Adds a new WeekException for a range of months within a year and returns a reference to it.

AddMonthException_MonthOfTheYear(ACaption:String;AFromMonthOfTheYear, AToMonthOfTheYear : integer): TMonthException

Adds a new MonthException for a range of months within a year and returns a reference to it.

ExceptionListCount:integer

Returns the total amount of exceptions defined by the DateException.

ExceptionList(ID:integer): TExceptionListElement

Provides access to an exception element.

GetCaption(const ADateTime:TDateTime):String

Returns the caption for a given time.

IsWorking(const ADateTime:TDateTime; const ADefault:Boolean):Boolean

Returns true, if the time specified by ADateTime is defined as working time.

GetException(const ADateTime:TDateTime): TBaseDateException

Returns the TBaseDateException element for a given time, otherwise it will return NIL.

Color:TColor

Defines the color for the exception.

Font:TFont

Holds a reference to the font object – that is used when drawing the caption.

ShowCaption:Boolean

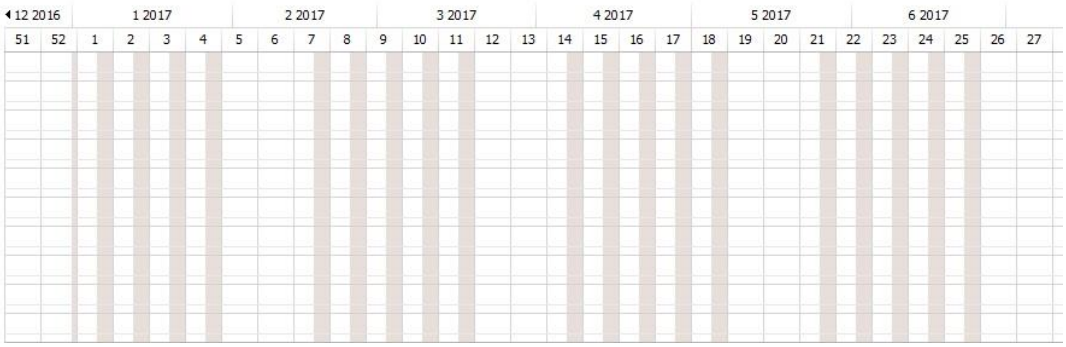
Specifies whether the caption should be visible or not.

Visible:Boolean

Defines whether the DateException is visible or not. When set to false, the DateException still defines working and non-working times for the gantt chart component.

RecurringException (TRecurringException)

For Exceptions that repeats in a certain chronological pattern you may use RecurringExceptions to define them. RecurringException are always defined within a given time range.



In the image above you see a calendar where a non-working time from Friday till Sunday is created and repeated for the year 2017. The week exception repeats 5 times and is followed by a pause of 2 weeks. In order to create such a calendar please use the code shown below:

```
1.  procedure TForm1.AddExceptions;
2.  var
3.   WeekException : TWeekException;
4.   RecException  : TRecurringException;
5.  begin
6.
7.   WeekException := TWeekException.Create(GanttChart1.BaseCalendar);
8.   WeekException.DeclareDayOfTheWeek('Week1', 5,7, false);
9.
10.  RecException := TRecurringException.Create(GanttChart1.BaseCalendar, StrToDate('01.01.2017'), S
    trToDate('01.01.2017'), StrToDate('31.12.2017'));
```

```
11.   RecException.Declare(WeekException,  5, 2,  dmWeek);
12.
13.   with GanttChart1.BaseCalendar.Add do
14.   begin
15.     AddException(RecException)
16.   end;
17.  end;
```

Properties and Methods

Declare(AException:TBaseDateException)

Adds an Exception to the recurring exception. The Exception will repeat itself.

Declare(AException:TBaseDateException; ARecInfo : TRecurringInfo)

Adds an Exception to the recurring exception.

Declare(AException:TBaseDateException; FRepetition, FDistanceToPrior : integer; FDistanceUnit:TDistanceMode)

Adds an Exception to the recurring exception. FRepetition specifies the total number the exception will be repeated. FDistanceToPrior specifies the distance in units set by the FDistanceUnit parameter, after the exception has been repeated.

RecurringStartDate:TDateTime

Defines the start date of the first exception.

RangeEndDate:TDateTime

Specifies the end of the range for the recurring exception.

RangeEndDateAssigned:Boolean

Returns True, when an EndRange has been defined for the recurring exception.

RangeStartDate:TDateTime

Specifies the start of the range for the recurring exception.

RangeStartDateAssigned:Boolean

Returns True, when a StartRange has been defined for the recurring exception.

Repetition:integer

Specifies the number of repetitions of the exception.

ExceptionListCount:integer

Returns the total amount of exceptions defined by the DateException.

ExceptionList(ID:integer): TExceptionListElement

Provides access to an exception element.

GetCaption(const ADateTime:TDateTime):String

Returns the caption for a given time.

IsWorking(const ADateTime:TDateTime; const ADefault:Boolean):Boolean

Returns true, if the time specified by ADateTime is defined as working time.

GetException(const ADateTime:TDateTime): TBaseDateException

Returns the TBaseDateException element for a given time, otherwise it will return NIL.

Color:TColor

Defines the color for the exception.

Font:TFont

Holds a reference to the font object – that is used when drawing the caption.

ShowCaption:Boolean

Specifies whether the caption should be visible or not.

Visible:Boolean

Defines whether the DateException is visible or not. When set to false, the DateException still defines working and non-working times for the gantt chart component.

DateCategorie(TDateCategory)

DateCategories are designed to manage and hold the DateExceptions. The list below shows the properties and methods of the DateExceptions. If there are DateExceptions in different categories that overlap or share the same date it is important to consider the order of the categories in the calendars category list. The date exceptions of the category that is last in the category list will be displayed topmost within the gantt chart. Generally you should try to avoid assigning a same date in different categories, as it is possible that there are contrary properties (working and non-working) in each category.

Properties and Methods

AddDayException:TdayException

Adds a new DayException to the category.

AddWeekExcprion:TWeekException

Adds a new WeekException to the category.

AddMonthException:TMonthException

Adds a new MonthException to the category.

AddYearException:TyearException

Adds a new YearException to the category.

AddException(ADateException:TbaseDateException):Boolean

Adds a DateException to the category.

IsWorking(ADateTime:TDateTime):Boolean

Returns true, if the specified ADateTime is defined as working time, otherwhise it will return false.

IsWorking(const ADateTime:TDateTime; var AException:TBaseDateException):Boolean

Returns true, if the specified ADateTime is defined as working time and will assign the corresponding DateException to the var parameter Aexception.

GetException(ADateTime:TDateTime):TBaseDateException

Will return the Exception for a specified ADateTime, if the exception includes definitions for the time, otherwhise it will return NIL.

Exception(Index:integer):TBaseDateException

Provides access to an Exception.

Count : integer

Returns the total amount of Exceptions.

Tag:integer

Provides to possibility to store an additional integer value to the Tag property.

Name:String

The name of the date category.

Working:Boolean

Defines whether exceptions are defined as working or not. Please note, that each DateException does define whether their relevant times are working or non-working times.

Calendar (TGanttCalendar)

The calendar holds one or multiple DateCategories and provides methods for the calculation and checking of time, regarding working- and non working times. The calendar uses an internal cache in order to precalculate working-

and non working times. By default the calendar has a cache size of 3650, whereby 1 entry hold information for a single day and requires 52 byte per day.

The following table shows the properties of the Calendar object.

Properties and Methods

Categorie[I:Integer]:TdateCategory

Returns the DateCategorie given by the index i.

Delete(I:Integer; AReleaseExceptions:Boolean=true)

Deletes the specified DateCategory. If AReleaseException has been set to false, the component will not Release the exceptions the categorie holds.

Add:TDateCategory

Adds a new DateCategory and returns a reference to it.

Add(Name:String):TDateCategory

Adds a new DateCategory, defines its name and returns a reference to it.

CategoryByName(AName: String): TDateCategory

Returns the first matching DateCatory specified by its name.

Clear

Clears the calendar and deletes all its content.

Count:integer

Returns the number of the DateCategorie objects.

GetWorkingTime2(AFrom:TDateTime; ATo:TDateTime):Double

Returns the effective working time for the date range AFrom till AFrom.

AddEffective2(AStart:TDateTime; AEffDuration:Double):TDateTime

Adds an effective duration to the date AStart and returns the resulting date.

SubtEffective2(AStart:TDateTime; AEffDuration:Double):TDateTime

Substracts an effective time duration from the date AStart and returns the resulting date.

GetException(ADay:integer; AHour:Byte):TBaseDateException

Returns the DateException for the Day and the hour AHour.

DayHasException(ADay:integer):Boolean

Returns true if there is a matching date exception for the given day, otherwise the calendar will return false.

IsWorking(ADate:TDateTime):Boolean

Returns true, if the date ADate is defined as working, otherwhise it will return false.

IsWorking(ADate:TDateTime; var AException:TBaseDateException):Boolean

Returns true, if the date specified by the parameter date is defined as a working time and sets the corresponding DateException as var parmeter AException.

IsWorkingFullDay(ADate:TDateTime):Boolean;

Returns true if the full day ‘ADate’ is defined as working time.

IsWorkingFullDay(ADate:TDateTime; var AException:TBaseDateException):Boolean

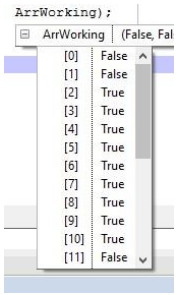
Returns true if the full day ‘ADate’ is defined as working time.

GetHours2(ADay:TDateTime; var ADayExeptionInfoArr:TDayExeptionInfoArr; var ADayExWorkingInfoArr:TDayExWorkingInfoArr)

GetHours2 returns detailed information, it will set the ADayExceptionInfoArr and the ADayExWorkingInfoArr for the given day ‘ADay’. Each DayException holds a unique integer ID from 0 to 255. The ADayExWorkingInfoArr does have a length of 24 byte elements. Each element of the array holds the ID for the Exception that defines the hour. The ADayExworkingInfoArr holds 24 boolean elements. If an element is false the corresponding hour is defined as working time. If an element does have the value ‘true’ it is defined as a non-working hour. The first element in both arrays represent the first hour of a day where the last element represent the last hour of a day.

```
1.  uses ...ganttcalendar...;
2.
3.  var
4.   aDayException : TDayException;
5.   ArrExceptions : TDayExeptionInfoArr;
6.   ArrWorking    : TDayExWorkingInfoArr;
7.   i : integer;
8.
9.  begin
10.
11.  // define  an exceptin
12.  aDayException := TDayException.Create(GanttChart1.BaseCalendar);
13.  aDayException.DeclareHour('hours non working', 2,10, false);
14.  GanttChart1.BaseCalendar.Add('Exception A').AddYearException.DeclareDate('Ex1', trunc(now), tru
     nc(now)+1, adayException);
15.
16.  // initialize arrays
17.  for i := 0 to 23 do
18.   begin
19.      ArrExceptions[i] := 0;
20.      ArrWorking[i] := false; // false = 'Working'; True = 'Non-Working';
21.   end;
22.
23.  GanttChart1.BaseCalendar.GetHours2(trunc(now)+1, ArrExceptions, ArrWorking);
24.
25. end;
```

The code sample above creates a day exception for a given date (now) and defines the hours from 02.00 AM till 10.00 AM as non working times (code line 13). After that the arrays ArrException and ArrWorking are initilized and the procedure GetHours2 is called for the given date. The array ArrException will hold the unique ID of the exception. The array ArrWorking will return true for the element 2 up to 10 as shown below:

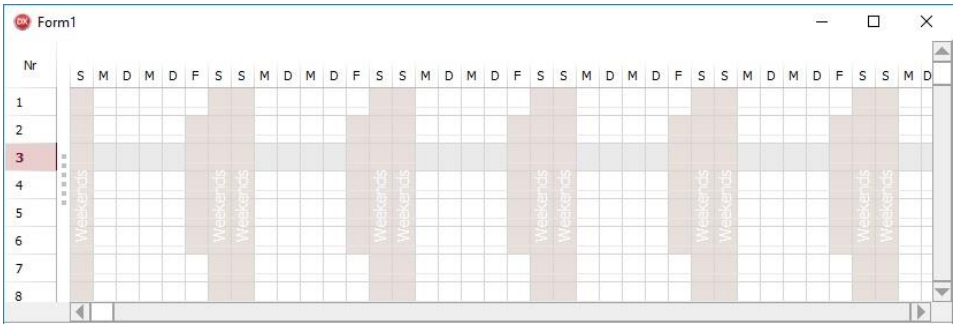


GetException(const ADateTime:TDateTime):TBaseDateException

Returns the exception for a specified DateTime value.

RowCalendarRepository:TGanttCalendarRepository

The gantt chart component enables the developer to define different calendars with different exceptions and apply them to a single or a set of rows. The RowCalendarRepository is a place to hold and store the calendar objects you want to create and define.



The image above shows a scenario where a base calendar is defined and additional a row calendar has been set up, where the rows 2 till 6 are linked to the row calendar. The code below shows the creation process of the row calendar object.

```
1.  procedure DefineRowCalendar;
2.  var
3.   gRowCal : TGanttRowCalendar;
4.   i:integer;
5.  begin
6.   gRowCal := GanttChart1.RowCalendarRepository.Add;
7.   gRowCal.Add.AddWeekException.DeclareDayOfTheWeek('Fr,Sa,Su',5,7,false);
8.   for i := 1 to 5 do
9.     GanttChart1.AbsoluteRow[i].RowCalendar := gRowCal;
10.
11.  GanttChart1.Repaint;
12. end;
```

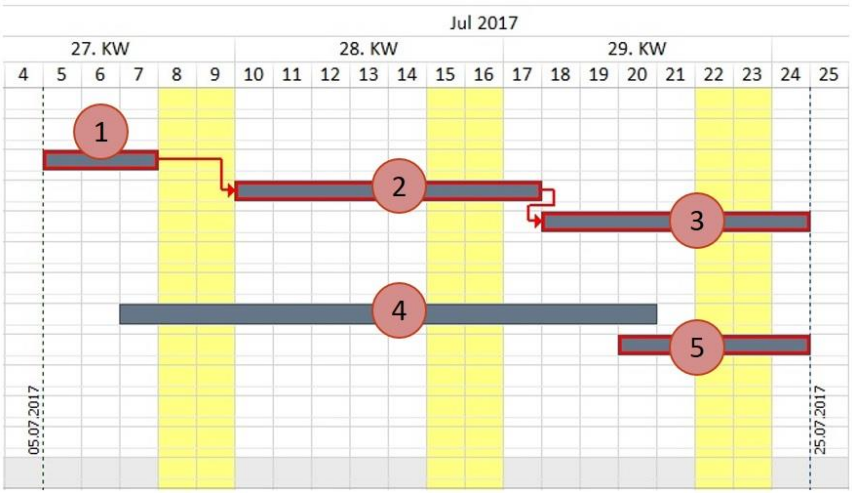
Each row object provides the function **GetCalendar**, that returns a reference to the calendar that is currently being used by the component. If a RowCalendar has been assigned to a row it has a higher priority and will be used instead of the BaseCalendar.

Critical Path

The gantt control component includes the calculation of the critical path. The calculation of the critical path determines which tasks are critical and non-critical. When a task is critical any delay of this task will result in a delay of the entire project. The critical path is the sequence of all critical tasks. Note that there can be more critical paths than one.

A typical critical path contains usually a set of connected bars. As soon as the first task, that is represented by the first bar, is delayed the second connected bar will be rescheduled, finally resulting in the rescheduling of the project end date that is determined by the end of the last bar.

The following image displays a gantt chart that contains two critical paths (1,2,3) and (5) visualized by red framed bars. In this example you can see that the only bar that will not reschedule the project end is bar (4) and therefore is not part of the critical path.



In order to enable or disable the calculation of the critical path you can set the boolean UpdateCriticalPath property of the OptionsSchedule record.

GanttChart.OptionsSchedule.UpdateCriticalPath:Boolean

The UpdateCriticalPath property will enable or disable the calculation of the underlying critical path data.

If you want to change the appearance of the critical path you can use the following properties:

GanttChart.OptionsView.CriticalPathOptions.Color:TColor

Specifies the color of the critical path.

GanttChart.OptionsView.CriticalPathOptions.PenStyle:TPenStyle

Specifies the pen style for the drawing of the critical path.

GanttChart.OptionsView.CriticalPathOptions.Visible:Boolean

Specifies whether the critical path should be visible or not.

By default the calculation and the visualisation of the critical path is disabled. To decide whether a bar or a connection is part of the critical path or not, you can access the boolean read only property **IsPartOfCriticalPath:Boolean**

Furthermore the critical path includes the calculation of the earliest possible start date, the earliest possible finish date, the latest possible start date and the latest possible end date without rescheduling any following bars. The following properties summarizes those critical path data. Please note that you have to set the **GanttChart.OptionsSchedule.UpdateCriticalPath** to true, in order to have those critical path data calculated.

TGanttBar.CPMData.EarliestStart:TDateTime

A datetime property defining the ealiest possible start time of a task/bar. The earliest starttime depends on the connection ending at the bar. The task cannot start earlier because other tasks have to be finished/started first.

TGanttBar.CPMData.EarliestFinish:TDateTime

A datetime property defining the earliest finish time of a task. The earliest possible finish time of a task is calculated by the earliest possible starttime and its duration.

TGanttBar.CPMData.LatestStart:TDateTime

A datetime property defining the latest possible start time of a task/bar. The latest starttime depends on the connection starting from the bar. If the bar would be moved only one day later then the whole project ending date would be delayed.

TGanttBar.CPMData.LatestFinish:TDateTime

A datetime property defining the latest possible finish time of a pert bar. The latest possible finish time of a task is calculated by the latest possible start time and its duration.

TGanttBar.CPMData.LocalBuffer:TDateTime

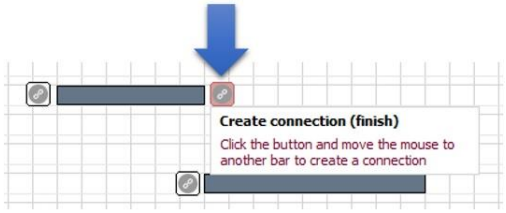
A datetime property defining the free buffer of a task. The free buffer is the number of intermediate days etween this bar and the earliest connected followed bar.

TGanttBar.CPMData.GlobalBuffer:TDateTime

A datetime property defining the global buffer of a task. The global buffer is the number of days that a task can delay without influencing the final project endtime. If the global buffer is 0 then any delay of this bar/task would extends the overall project time. If the global buffer is zero, then the according task/bar is part of the critical path.

Connections between bars (TBarConnection)

For modelling temporal and causal relations between two tasks/activities you can connect two bars with each other and create a connection. Connections are represented as TBarConnection objects. If you reschedule a bar by default, all connected bars are rescheduled automatically, so that the condition of the connection will not be violated.



To create a connection at runtime, the user has to click on the bar button and move the mouse cursor inside the target bar object. The end user can specify the type of the connection by clicking either on the start or on the end of a bar.

Please note, that it is not possible to create cyclic connections between bars. No successor can at anytime be a predecessor of a bar.

The code below, demonstrates how to create a connection between two bars:

```
1.  procedure MakeConnection;
2.  var
3.   Bar1,
4.   Bar2 : TGanttBaseBar;
5.   Con  : TBarConnection;
6.  begin
7.   Bar1 := GanttChart1.AbsoluteRow[0].Bar[0];
8.   Bar2 := GanttChart1.AbsoluteRow[1].Bar[0];
9.   // create a connection
10.  GanttChart1.BeginUpdate;
11.  try
12.    Con := GanttChart1.DataController.AddConnection( Bar1, Bar2, bcFinishToStart, true);
13.    Con.Color := clLime;
14.  finally
15.    GanttChart1.EndUpdate;
16.  end;
17. end;
```

Properties

The following list gives an overview of the properties of the TBarConnection object:

BarFromGUID: TGUID

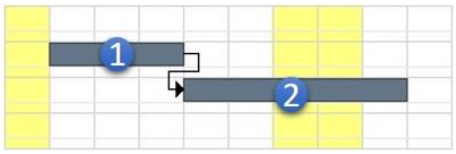
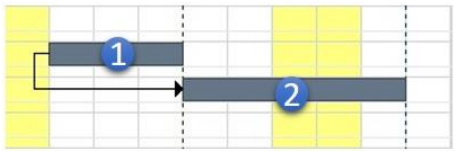
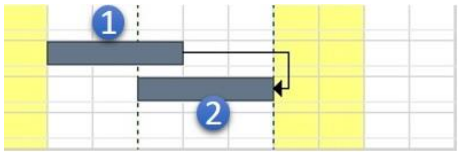
Returns the GUID of the connected predecessor bar object.

BarToGUID : TGUID

Returns the GUID of the connected successor bar object.

ConnectionType : TBarConnectionType = (bcFinishToStart, bcStartToFinish, bcStartToStart, bcFinishToFinish)

Specifies the type of the connection. Basically there are four different types of connections, describing different causal relations between two bars:

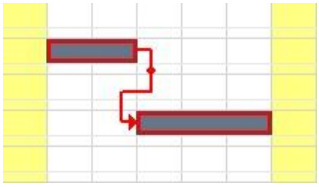
bcFinishToStart 	The task #2 cannot start before its predecessor #1 ends, although it may start later. This is the most common connection type.
bcStartToFinish 	Task #2 ends always after the start of task #1.
bcStartToStart 	Task #2 starts always after the start of task #1.
bcFinishToFinish 	Task #2 ends always after the end of task #1.

MinDistance : TDateTime

The minimal distance between two bars.

DistanceFixed : Boolean

Specifies whether the distance between the two connected bars should stay fixed when scheduling one of the connected bars. As you can see below, fixed connections are visualized with a small dot.



HotTracked : Boolean

Returns true if the connection is currently beeing hot tracked.

Valid : Boolean

Returns true, if the two bars do not violate the causal restriction of a connection.

IsPartOfCriticalPath : Boolean

Returns true if the connection is part of the critical path.

Color : TColor

Specifies the color of the connection.

BarFrom:TGanttBaseBar

Returns the connected predecessor bar object.

BarTo:TGanttBaseBar

Returns the connected successor bar object.

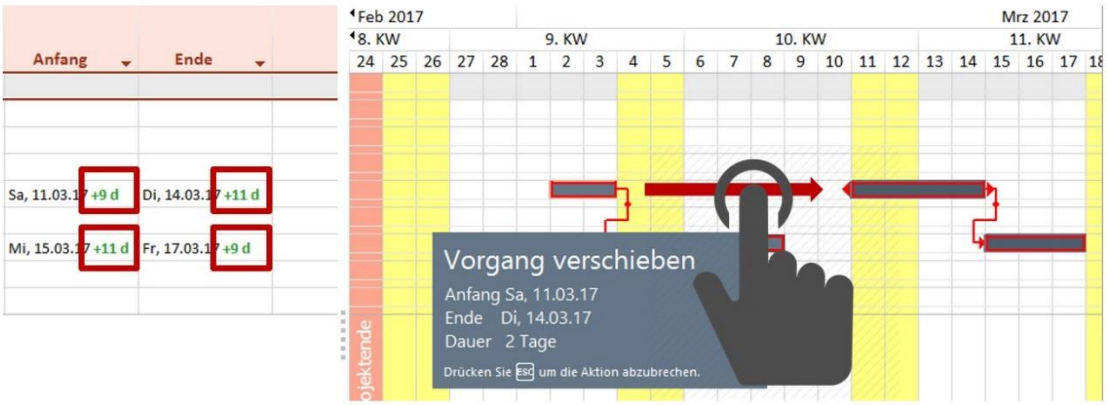
Delete

Deletes the connection.

Create(AOwner:TComponent; ABarFrom, ABarTo : TGanttBaseBar; AConnectionType:TBarConnectionType)

The constructor of the TBarConnection object.

When the end user reschedules a bar in a row by horizontally dragging it to a new position, all connected bars are automatically rescheduled immediately and the end user is able to see the resulting effects. While dragging a bar the changes for the start, end and duration column are shown in the table for all affected bars. You can cancel the rescheduling if you press the 'ESC' key at anytime.



By default all connected bars are rescheduled automatically when a bar will be repositioned. However you can set the **AutomaticScheduling** property for a TGanttRow to false – in this case a bar will never be rescheduled automatically by the component.

Pert chart

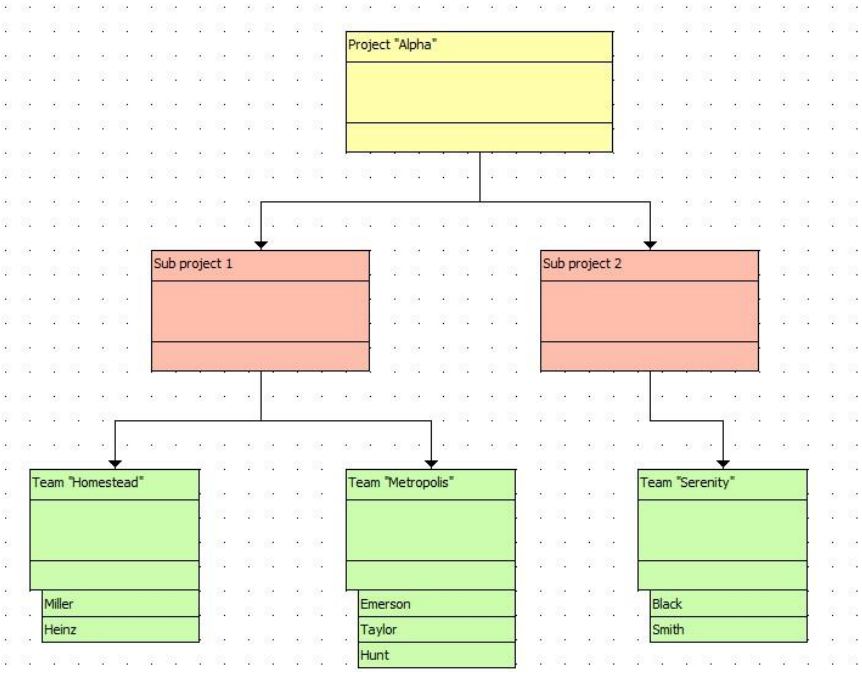
Overview

The pert chart is another viewmode of the gantt chart that visualizes single tasks (rows) as pert bars. One Pert bar represents a single row. The pert chart area does not have a time scale nor is it structured in rows. Two pert bars can be connected with each other. The gantt chart view and the pert chart view share the same data source. Therefore changes made in one view are represented in the other.

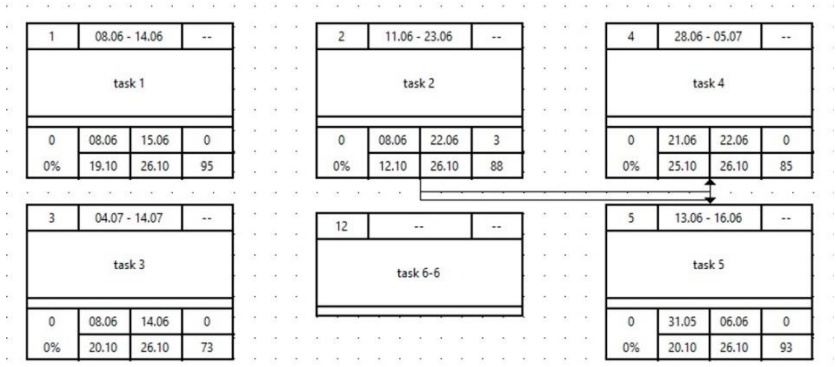
Within the pert chart the duration of the rows task bars as well as the earliest start date, the earliest end date, the latest start date and the latest end date are visualized in the pert bars.

To model existing structures of a project, pert bars can be used too. Each pert bar is divided into three segments and an optional list displaying additional items. Note: You can use the pert chart for structuring and segmenting projects. Higher level project targets, sub ordinate project targets and work packages can be visualized and modelled as pert bars. For example pert bars that represent working packages can be extended by a list of responsible employees. To model relations between targets and working packages the pert chart supports connecting pert bars with each other.

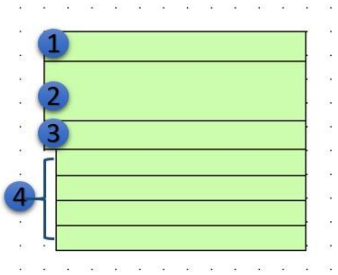
Please have a look of the following picture for an example of a hierarchical project structuring using pert bars.



Note: All pert bars shown in the example above have its **PertBarType** property set to pbtNoTimeSpan. If you intend to use the pert bars for displaying critical time requirements, as the earliest start date, the earliest end date, the latest start date and the latest end date of a task, you have to assign pbtTimeSpan for the PertBarType property. The picture below shows pert bars displaying additional time span information.

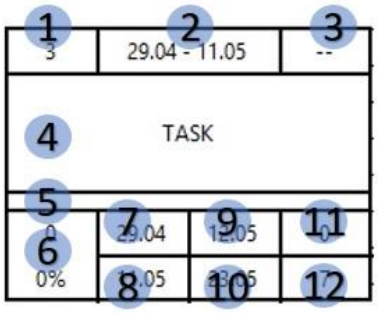


PertBarType ‘pbtNoTimeSpan’



Pertbars that have set the PertBarType to pbtNoTimeSpan consists of the top section (1), the main section(2), the bottom section(3) and an optional AdditionalSection(4).

PertBarType ‘pbtTimeSpan’



Pertbars that have set the PertBarType to pbtTimeSpan have the following elements:

(1) The according row number of the pert bar

(2) The overall duration of the pert bar

(3) The number of the parent row of the pert bar.

(4) The name of the task

(5) A visualization of the progress of the pert bar

(6) The numeric value of the pert bars progress

(7) The earliest start of the pert bar

(8) The latest start of the pert bar

(9) The earliest finish/end of the pert bar

(10) The latest finish/end of the pert bar

(11) The local free buffer time (in days) of a pert bar

(12) The global free buffer time (in days) of a pert bar

Properties / Methods

X:Integer

The x-position in pixel of the top left point of the pert bar.

Y:Integer

The y-position in pixel of the top left point of the pert bar.

Width:Integer

The width in pixel of the pert bar.

PertBarType : TPertBarSection= (pbtTimeSpan, pbtNoTimeSpan)

The type of the pert bar.

Detailed : Boolean

If detailed is set to true the additional setion of a pert bar will be shown.

Text : String

The text that is displayed in the main section of the pert bar.

TopSection : TPertBarSection

A reference to the top section of the pert bar.

MainSection: TPertBarSection

A reference to the main section of the pert bar.

BottomSection: TPertBarSection

A reference to the bottom section of the pert bar.

AdditionalSection: TPertBarSection

A reference to the additional section of the pert bar.

SetWidthAll(AWidth:integer)

Sets the width of the pert bar.

VisibleHeight:integer

Returns the visible height of the pert bar.

A TPertBarSection object does hold the following properties:

Settings.Font: TFont

The font object for the section.

Color:TColor

The background color of the pert bars section.

Height:integer

The height in pixel of the pert bars section.

Width:integer

The width in pixel of the pert bars section.

Alignment:TAlignment

The alignment of the pert bars section.

TextAlignment:TAlignment

The text alignment of the pert bar section.

Text:String

The text of the pert bar section.

Adding pert bars

There are different ways how to add a pert bar. Please keep in mind that pert bars and task bars are only different visual entities of the same tasks. This means a pert bar will automatically be created if you (or the enduser) add a taskbar to the gantt chart, when the boolean flag **TGanttChart.PertChartView.AutoCreatePertBars** has been set to true.

```
1.  ..
2.   try
3.     GanttChart1.BeginUpdate;
4.     GanttChart1.DataController.AddPertbar( GanttChart1.AbsoluteRow[2], nil);
5.   finally
6.     GanttChart1.EndUpdate;
7.   end;
8.  ..
```

The code example above shows, how to add a pert bar to a row.

Accessing/Deleting pert bars

Pert bars can be accessed by using the **PertBar:TPertBar** property of a TGanttRow. The resulting PertBar of a row will be nil, if it does not have a pert bar.

In order to delete a pert bar by code you may use the RemoveBar method as shown below:

```
1.  ..
2.   try
3.     GanttChart1.BeginUpdate;
4.     GanttChart1.DataController.RemoveBar( GanttChart1.AbsoluteRow[2].PertBar );
5.   finally
6.     GanttChart1.EndUpdate;
7.   end;
8.  ..
```

Connecting pert bars

The code below uses the AddConnection method to connect two pert bars.

```
1.  ..
2.   GanttChart1.DataController.AddConnection(GanttChart1.AbsoluteRow[2].PertBar,GanttChart1.Absolute
    Row[3].PertBar,bcFinishToStart,false);
3.  ..
```

GanttChart.PertChartView:TPertChartView

The gantt chart component holds several settings that are related to the pert chart view and are accessible by the PertChartView class.

MaxWidth: Integer

The with of the pert chart area in pixel.

MaxHeight: Integer

The height of the pert chart area in pixel.

SpacingX: Integer

You can set up a pattern grid. SpacingX determines the horizontal distance between single dots of the pattern grid.

SpacingY:Integer

SpacingY determines the vertical distance between single dots of the pattern grid.

AutoCreatePertBars : Boolean

When set to true, the component will automatically add a new pert bar when a task bar is created.

AutoCreateNewRow : Boolean

When set to true, the component will automatically add a new gantt row, if the user creates a new pert bar and there is no 'empty' ganttrow.

ShowPertGrid:Boolean

Determines whether the pattern grid will be shown or not.

PertGridColor:TColor

The color of the pattern grid.

PertCaption : string

A caption that is shown above the pert chart.

PertSubCaption:string

A secondary caption that is shown above the pert chart.

DragPertBarType:TCreateNewPertBarType = (nbPertBarTimeSpan, nbPertBarNoTimeSpan)

Specifies the new type of the pert bar that will be created.

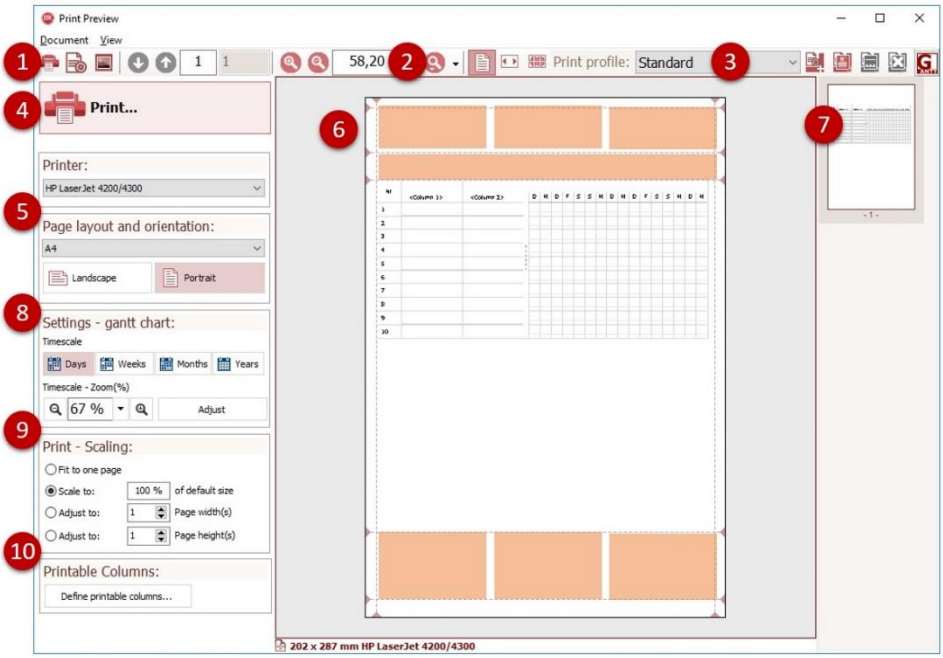
Print Preview

Overview

The gantt chart component comes along with a print preview component The print preview is highly adabtable and allows the specification of lots of parameters (zoom, pagination, page title, background image...).

In addition a legend that is segmented into rows and columns can be attached to the gantt chart for printing. Each cell can contain text or images. Furthermore the print preview provides the possibility to set up a page header and page footer, that is segmented into three areas.

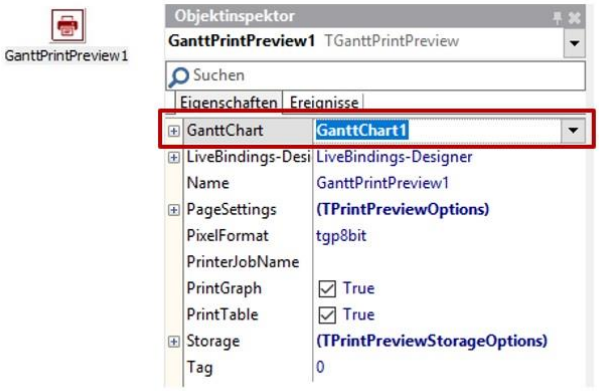
The following image gives an overview of the Print Preview and describes the most relevant parts:



Item	Description
1	The tool button section including buttons for printing, setup the page, add additional legends and images as well as navigating through the pages.
2	A section where the user can adjust the zoom factor of the print preview. This zoom factor is only relevant for the preview itself and does not effect the printed result.
3	Additionaly the user can save the current settings into a print profile. At this section the user is able to load/save/delete an existing print profile.
4	The print button, that initiates the print process.
5	A section where the user can specify the printer, the page layout and the orientation.
6	The preview of the gantt chart component. Here you can see the areas for the page header, the title area and the page footer.You may double click directly at a section, to edit its properties. In addition you can drag the small dotted lines and adjust the margins for the page and for the different sections.
7	A section where thumbnails for single pages are shown.
8	A section where basic settings of the gantt chart can be adjusted to influence the print.
9	Scaling options for the document.
10	A button that invokes a dialog where the user can specify the printable columns.

Set up the Print Preview

To set up the Print Preview, please place a new TGanttPrintPreview component from the Tool panel - section ‘Gantt Suite 3’ onto your form and link the TGanttChart component to the TGanttPrintPreview component by assigning a value to the GanttChart property.



In order to invoke the Print Preview you may call the statement that is shown below:

```
1.  ..
2.  GanttPrintPreview1.ShowPreview();
3.  ..
```

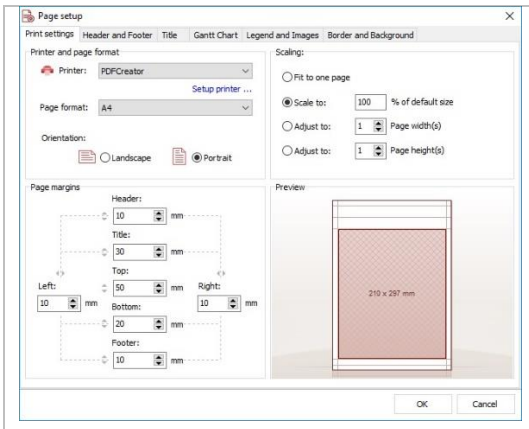
If you want to print the content without having shown any preview you may call the ShowPrintDlg() method as shown below.

```
4.  ..
5.  GanttPrintPreview1.ShowPrintDlg();
6.  ..
```

Page setup dialog

The page setup dialog has the tab pages “Print settings”, “Header and Footer”, “Title”, “Gantt Chart”, “Legend and Images” and “Border and Background”

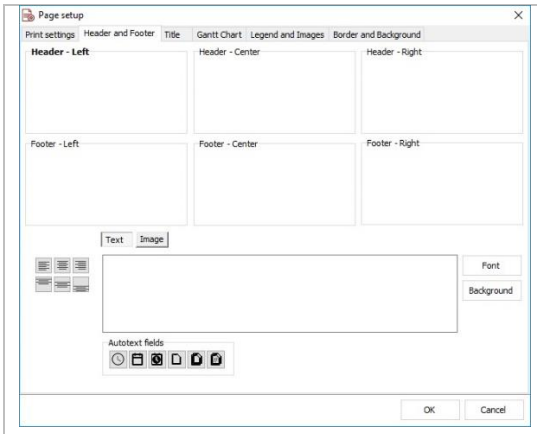
Print Settings



Here you can see the selected printer, the page format and its orientation. On the lower left the user can specify the page borders.

On the top right section the user can specify the scaling options, inlcuding to set a scale factor, adjust to n page widths/ page heights or to fit the content to a single page.

Header and footer

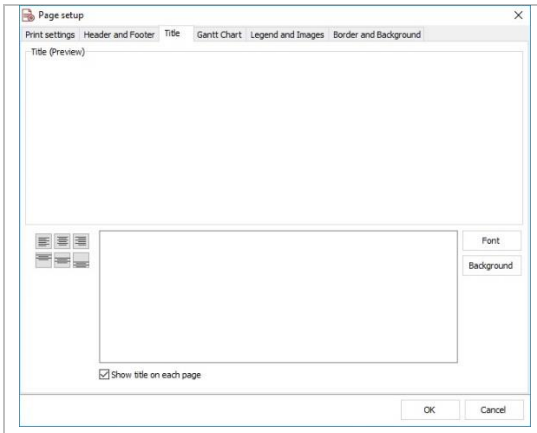


At this point the page header and the page footer can be defined. Header and footer are divided into three sections. At the top of the dialog is a preview of the header and footer.

For each segment of the page header and footer you can either type in some text or load an image that is displayed inside the segment. Also you can insert auto text fields into single segments. Auto text fields are responsible for displaying the current page number, the current time and so on.

Whenever the area for the page header and footer is to small to display the entire content, the area will be drawn cross hatched.

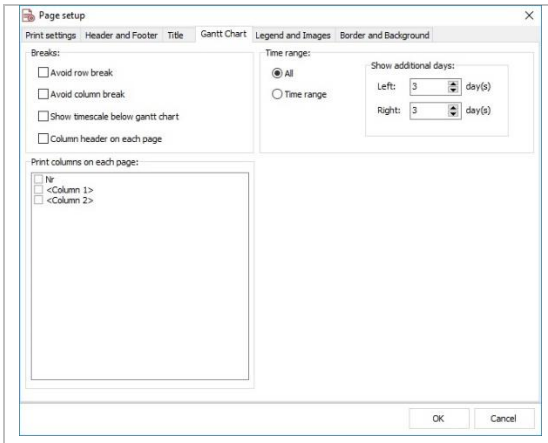
Title



The user is able to apply a page title (caption). The page title is displayed below the page header section. A background color, the font and the alignment can be adjusted for the page title.

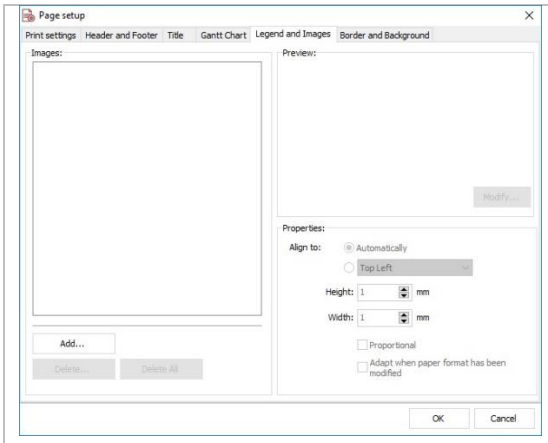
Furthermore the user can specify whether the page title should be printed for each page or not.

Gantt Chart



In the section “Gantt Chart”, you can specify the time range of the gantt chart, the printable columns and whether the component should avoid row and column breaks at page breaks.

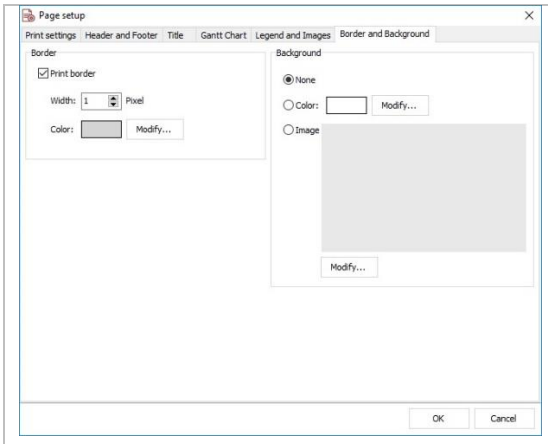
Legend and images



The user can add images to the print preview. Images can be fully scaled and positioned everywhere on the print area.

For each image it may be possible to define a page corner where the image is aligned to.

Border and Background



It is possible to print a border that surrounds the entire content. A color and the width of the border can be specified here.

If the user selects the option “Image” he can define a background image for the entire print content.

TGanttPrintPreview.PageSettings

GanttRangeStart : TDateTime

The start date of the gantt chart.

GanttRangeEnd : TDateTime

The end date of the gantt chart.

BorderColor : TColor

Specifies the color of the border.

BorderWidth : Integer

Specifies the width of the border.

BorderVisible: Boolean

Specifies whether the border is visible or not.

GanttRangeAuto:Boolean

If the GanttRangeAuto property has been set to true the start date and the end date of the gantt chart will be automatically calculated by the earliest and the latest bar in the gantt chart.

GanttRangeAutoDaysLeft:Integer

If GanttRangeAuto has been set to true, GanttRangeAutoDaysLeft specifies the additional left time buffer in days.

GanttRangeAutoDaysRight:Integer

If GanttRangeAuto has been set to true, GanttRangeAutoDaysRight specifies the additional right time buffer in days.

FitToPage:Boolean

If FitToPage has been set to true, the gantt chart is stretched to fit one page.

Zoom:Double

Sets the zoom for the print content. A value of 1 represents 100 percent.

ScaleToPagesWidth:Boolean

If set to true the print content will be stretched to **ScaleToPageWidthCount** pages widths.

ScaleToPagesHeight:Boolean

If set to true the print content will be stretched to **ScaleToPageHeightCount** pages heights.

ScaleToPageWidthCount : integer

The number of page widths the print content will be stretched to.

ScaleToPageHeightCount : integer

The number of page heights the print content will be stretched to.

PreviewZoom : double

The zoom factor for the print preview.

PreviewZoomMode : TPreviewZoomMode = (pzmOnePage, pzmFitToPageWidth, pzmMultiplePages)

Specifies how the preview renders single or multiple pages. When the PreviewZoomMode has been set to pzmOnePage, the zoom factor PreviewZoom will be used to render a single page. When the PreviewZooMode has been set to pzmFitToPageWidth, the preview will adjust the zoom factor so that a page width is fully visible. When the PreviewZoomMode has been set to pzmMultiplePages the print preview will display multiple pages at once.

MarginLeftMM: integer

The left page margin in millimetres.

MarginTopMM: integer

The top page margin in millimetres.

MarginRightMM: integer

The right page margin in millimetres.

MarginBottomMM: integer

The bottom page margin in millimetres.

MarginHeaderMM: integer

The header margin in millimetres.

MarginFooterMM: integer

The footer margin in millimeters.

MarginTitleMM: integer

The title margin in millimetres.

PageBreak: Boolean

If set to true the component will try to prevent breaking the content of rows when a page break occurs.

ColumnBreak: Boolean

If set to true the component will try to prevent breaking the content of a column when a page break occurs.

ShowTimeScaleAtBottom : Boolean

If set to true the timescale(s) will be shown below the gantt chart.

Header.LeftSection: TFooterHeaderSectionSettings

Holds properties of the left page header section.

Header.CenterSection: TFooterHeaderSectionSettings

Holds properties of the center page header section.

Header.RightSection: TFooterHeaderSectionSettings

Holds properties of the right page header section.

Footer.LeftSection: TFooterHeaderSectionSettings

Holds properties of the left page footer section.

Footer.CenterSection: TFooterHeaderSectionSettings

Holds properties of the center page footer section.

Footer.RightSection: TFooterHeaderSectionSettings

Holds properties of the right page footer section.

Title : TTitleSettings

Title properties.

Background: TBackgroundSettings

Properties defining the layout and style of the background.

ColumnsOnEachPage : TStrings

A list of columns that are displayed on each page.

HeaderOnEachPage : Boolean

Specifies whether the header should be shown on each page.

OverlayImages : TOverlayImages

Provides access to a list to overlay images.

SaveProfileOnPreviewClose : Boolean

Saves the page settings when the print preview is closed.

ProjectName : String

The project name.

ProjectNr : String

The project number.

Date : String

Current date.

TFooterHeaderSectionSettings

IsImage : Boolean

Returns true if the content of the section is an image.

HorzAlignment : TAlignment

The horizontal alignment of the content.

VertAlignment : TVerticalAlignment

The vertical alignment of the content.

Font : TFont

The font object of the page header or page footer section.

Color : TColor

The background color of the section.

ImageMode : TFooterHeaderImageDisplayMode = (idmStretch, idmDoNotStrech, idmTile)

The ImageMode determines how the images are displayed in the section. Images can be drawn strechted (idmStretch), unstretched (idmDoNotStretch) or tiled.

Picture : TPicture

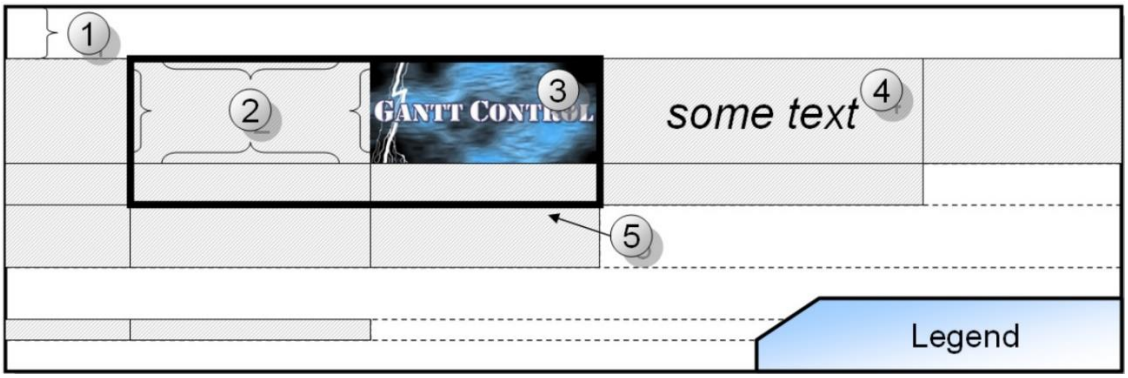
A reference to the picture.

Text: string

The text of the header and footer section.

Legend:TGanttLegend

To display any textual and graphical information you can create a legend. The legend will be displayed below or right next to the gantt chart. The architecture of a legend is based on a grid-like structure, including a set of rows (**#1**) and cells (**#2**) that are added to a row as shown in the picture below.



The interface provides access to cells canvas – so it's possible to draw some images (**#3**) to a cell or to draw some textual information (**#4**). Please note that a row does not need to have cells at all. Also each row can own a different amount of legend cells. Furthermore there is the possibility to assign a different horizontal alignment for each row. Also you can define borders (**#5**) surrounding cells. A border can be applied to a range of legend cells that is defined by the start row, the start cell, the end row and the end cell.

The following table shows the most relevant operations and properties of the legend.

Position : TGanttLegendPosition=(glposBelow, glposRight)

Specifies the position of the legend.

Visible : Boolean

Specifies whether the legend is visible or not.

Color : TColor

The basic background color of rows of the legend. Each cell can hold its own color value.

BorderColor : TColor

The color of the border.

BorderWidth : integer

The width in pixel of the border.

Rows[i:integer] : TGanttLegendRow

Provides access to the rows of a legend.

Border[i:integer] : TGanttLegendBorder

Provides access to the borders of the legend.

AddRow:TGanttLegendRow

Adds a new row to the legend and returns a reference to it.

AddBorder:TGanttLegendBorder

Adds a new border to the legend.

RowCount : integer

Returns the total number of rows in the legend.

BorderCount : integer

Returns the total number of borders of the legend.

GetExtension:TPoint

Returns the overal dimension of the legend.

GetRowRect(ARect:TRect; ARowID:integer):TRect

Returns the rect of a legend row.

TGanttLegendRow

Height : integer

Returns the height of a legend row.

Alignment : TGanttRowAlignment = (tgRALeft, tgRARight, tgRACenter, tgRAStretch)

The horizontal alignment of all cells within this row.

TGanttRowAlignment	Alignment
tgRALeft	Left justify
tgRARight	Right justify
tgRACenter	Center
tgRAStretch	The cells are splitted; specified by the SplitAt position.

SplitAt : integer

All cells that's ID is lesser or equal then the SplitAt property, are displayed left aligned the rest of the cell is right aligned. You have to set tgRAStretch to the row alignment.

RowIndex : integer

Returns the row index.

Clear

Removes all cells of the row.

Width : integer

Returns the width of the row.

GetCellRect(ID:integer; ARect:TRect):TRect

Returns the cell rect of the specified cell.

AddCell:TGanttLegendCell

Adds a new cell to the row and returns a reference to it.

Cell[i:integer] : TGanttLegendCell

Provides access to the cells of a legend row.

CellCount:integer

Returns the total number of cells the row owns.

TGanttLegendCell

Width : integer

Specifies the width of a cell.

Color : TColor

Background color of the cell.

Font : TFont

The font that is used by this cell.

VertAlignment : TCellVerticalAlign

The vertical text alignment of the cell.

HorzAlignment : TCellHorzAlign

The horizontal text alignment of the cell.

WordWrap : Boolean

When set to true, the text of the cell will be automatically word wrapped.

Text : String

The text of the cell.

Picture : TPicture

Picture of the cell.

Tag : integer

With the help of the tag property any developer that uses the component can store user defined values to the legend cell.

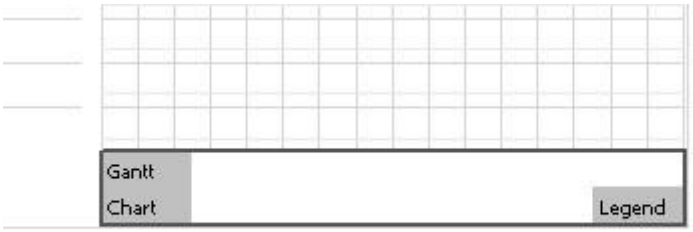
RowIndex : integer

Returns the index of the row.

CellIndex : integer

Returns the index of the cell within the row.

The following code example demonstrates, how to create a simple legend add two new rows and add a border. The resulting legend is shown below.




```
1.  ...
2.  GanttChart1.Legend.Clear;
3.
4.  with GanttChart1.Legend.AddRow do
5.  begin
6.    Color := clWhite;
7.    Alignment := tgRALeft;
8.    with AddCell do
9.    begin
10.     Color := clSilver;
11.     Text := 'Gantt';
12.   end;
13. end;
14.
15. with GanttChart1.Legend.AddRow do
16. begin
17.   Alignment := tgRAStretch;
18.   SplitAt  := 0;
19.
20.   with AddCell do
21.   begin
22.     Color := clSilver;
23.     Text := 'Chart';
24.   end;
```

```
25.
26.     with AddCell do
27.     begin
28.       Color := clSilver;
29.       Text := 'Legend';
30.     end;
31.   end;
32.
33.   with GanttChart1.Legend.AddBorder do
34.   begin
35.     StartRow := 0;
36.     StartCell := 0;
37.     EndRow := 1;
38.     EndCell := 1;
39.     Color := $00585858;
40.     Thickness := 2;
41.   end;
42. ...
```

Localization

The Gantt Control Component version 3.0 is available for the both languages German and English. The following table shows all functions and methods defined in the ganttresources.pas unit.

If you want to localize the component for an english user interface simply add the unit ganttresources to the uses clause of your main form and call the CreateDefaultRessources_ENG method.

Function	Description
SetRessourceString(ResStringRec: pResStringRec; NewStr: string);	Localizes a single language depentent string value.
CreateDefaultRessources_GER	Localizes the Gantt Control Component for the German language.
CreateDefaultRessources_ENG	Localizes the Gantt Control Component for the English languages.

Internally – both the methods CreateDefaultRessources_GER and CreateDefaultRessources_ENG call the SetRessourceString(ResStringRec: pResStringRec; NewStr: string) method to define the string values for all language specific user interface elements.

You can overwrite any existing RessourceString element with a new translation anytime.

The following table summarizes all language dependent ressource strins and their value for the English language.

```
1.  SetRessourceString(@grs_rownoteedit, 'Edit row note');
2.  SetRessourceString(@grs_input, 'Input');
3.  SetRessourceString(@grs_sort, 'Sort');
4.  SetRessourceString(@grs_filter, 'Filtering');
5.  SetRessourceString(@grs_inserttask, 'Insert task');
```

```
6.   SetRessourceString(@grs_addcolumn, 'Add column');
7.   SetRessourceString(@grs_insertcolumn, 'Insert column');
8.   SetRessourceString(@grs_deletecolumn, 'Remove column');
9.   SetRessourceString(@grs_movecolumn, 'Reposition column');
10.  SetRessourceString(@grs_clearallcolumns, 'Remove all columns');
11.  SetRessourceString(@grs_clearcontent, 'Clear content');
12.  SetRessourceString(@grs_deleterow, 'Remove task');
13.  SetRessourceString(@grs_deleterows, 'Remove tasks');
14.  SetRessourceString(@grs_addchildrow, 'Add child row');
15.  SetRessourceString(@grs_rowchangeparent, 'Change hierarchy');
16.  SetRessourceString(@grs_rowsgroup, 'Group tasks');
17.  SetRessourceString(@grs_rowheight, 'Row height');
18.  SetRessourceString(@grs_columnwidth, 'Column width');
19.
20.  SetRessourceString(@grs_all, 'All');
21.  SetRessourceString(@grs_empty, '(empty)');
22.  SetRessourceString(@grs_cancelsort, 'Cancel sort');
23.  SetRessourceString(@grs_cancelfilter, 'Cancel all filter');
24.
25.  SetRessourceString(@grs_btn_cancel, 'Cancel');
26.  SetRessourceString(@grs_btn_ok, 'OK');
27.
28.  SetRessourceString(@grs_sort_AZ, 'Sort A to Z');
29.  SetRessourceString(@grs_sort_ZA, 'Sort Z to A');
30.  SetRessourceString(@grs_sort_dateAsc, 'Sort Date ascending');
31.  SetRessourceString(@grs_sort_dateDesc, 'Sort Date descending');
32.  SetRessourceString(@grs_sort_09, 'Sort 0 to 9');
33.  SetRessourceString(@grs_sort_90, 'Sort 9 to 0');
34.
35.  SetRessourceString(@grs_calendarweekshort, 'CW');
36.  SetRessourceString(@grs_weekshort, 'W');
37.  SetRessourceString(@grs_quarter, 'Quarter');
38.  SetRessourceString(@grs_quartershort, 'Q');
39.
40.  SetRessourceString(@grs_hour, 'Hour');
41.  SetRessourceString(@grs_day, 'Day');
42.  SetRessourceString(@grs_week, 'Week');
43.  SetRessourceString(@grs_month, 'Month');
44.  SetRessourceString(@grs_quarter, 'Quarter');
45.  SetRessourceString(@grs_year, 'Year');
46.
47.  SetRessourceString(@grs_worktime, 'Working Time');
48.  SetRessourceString(@grs_nonworktime, 'Non-working Time');
49. // TfrmGanttPrintForm
```

```
50.  SetRessourceString(@grs_noprinteravailable,'no printer available');
51.  SetRessourceString(@grs_printer,'Printer');
52.  SetRessourceString(@grs_printrange , 'Print range');
53.  SetRessourceString(@grs_currentpage , 'Current page');
54.  SetRessourceString(@grs_pages , 'Pages');
55.  SetRessourceString(@grs_pagesadd , '(e.g. 1-4 or 1,4)');
56.  SetRessourceString(@grs_settings , 'Settings');
57.  SetRessourceString(@grs_copies , 'Copies');
58.
59.  // TfrmPrintProgress
60.  SetRessourceString(@grs_printprogress , 'Printing - Progress');
61.
62.  // TfrmRenameProfile
63.  SetRessourceString(@grs_renameprintprofile , 'Renamne print profile');
64.  SetRessourceString(@grs_printprofile , 'Print profile');
65.
66.  // TfrmSaveProfil
67.  SetRessourceString(@grs_saveAsPrintprofile , 'Save print profile as');
68.  SetRessourceString(@grs_existingprofile , 'Existing profile');
69.  SetRessourceString(@grs_newprofile , 'New profile');
70.
71.  //TfrmColumnsSelect
72.  SetRessourceString(@grs_printablecolumns , 'Printable columns');
73.  SetRessourceString(@grs_printablecolumnsubtext , 'Please select the printable columns:');
74.
75.  //TPageSetupDlg
76.  SetRessourceString(@grs_pagesetupdlg , 'Page setup');
77.  SetRessourceString(@grs_ps_printsettings , 'Print settings');
78.  SetRessourceString(@grs_ps_printerandformat , 'Printer and page format');
79.  SetRessourceString(@grs_ps_printersetup , 'Setup printer ...');
80.  SetRessourceString(@grs_ps_format , 'Page format:');
81.  SetRessourceString(@grs_ps_orientation , 'Orientation:');
82.  SetRessourceString(@grs_ps_landscape , 'Landscape');
83.  SetRessourceString(@grs_ps_portrait , 'Portrait');
84.  SetRessourceString(@grs_ps_scaling , 'Scaling:');
85.  SetRessourceString(@grs_ps_adjusttosinglepage , 'Fit to one page');
86.  SetRessourceString(@grs_ps_scaleto , 'Scale to:');
87.  SetRessourceString(@grs_ps_scaletosuffix , '% of default size');
88.  SetRessourceString(@grs_ps_scaletosuffix2 , 'of default size');
89.  SetRessourceString(@grs_ps_adjustto , 'Adjust to:');
90.  SetRessourceString(@grs_ps_pagewidths , 'Page width(s)');
91.  SetRessourceString(@grs_ps_pageheights , 'Page height(s)');
92.  SetRessourceString(@grs_ps_pagemargins , 'Page margins');
93.  SetRessourceString(@grs_ps_header , 'Header:');
```

```
94.  SetRessourceString(@grs_ps_title , 'Title:');
95.  SetRessourceString(@grs_ps_top , 'Top:');
96.  SetRessourceString(@grs_ps_left , 'Left:');
97.  SetRessourceString(@grs_ps_right , 'Right:');
98.  SetRessourceString(@grs_ps_bottom , 'Bottom:');
99.  SetRessourceString(@grs_ps_footer , 'Footer:');
100. SetRessourceString(@grs_ps_mm  , 'mm');
101. SetRessourceString(@grs_ps_preview , 'Preview');
102.
103. SetRessourceString(@grs_ps_headerandfooter , 'Header and Footer');
104. SetRessourceString(@grs_ps_headerleft , 'Header - Left');
105. SetRessourceString(@grs_ps_headercenter , 'Header - Center');
106. SetRessourceString(@grs_ps_headerright , 'Header - Right');
107. SetRessourceString(@grs_ps_footerleft , 'Footer - Left');
108. SetRessourceString(@grs_ps_footercenter , 'Footer - Center');
109. SetRessourceString(@grs_ps_footerright , 'Footer - Right');
110. SetRessourceString(@grs_ps_text , 'Text');
111. SetRessourceString(@grs_ps_image , 'Image');
112. SetRessourceString(@grs_ps_font , 'Font');
113. SetRessourceString(@grs_ps_background , 'Background');
114. SetRessourceString(@grs_ps_autotextfields , 'Autotext fields');
115. SetRessourceString(@grs_ps_viewmode , 'View mode:');
116. SetRessourceString(@grs_ps_select , 'Select...');
117. SetRessourceString(@grs_ps_remove , 'Remove');
118.
119. SetRessourceString(@grs_ps_cbHdrFtrImageMode1 , 'Stretch');
120. SetRessourceString(@grs_ps_cbHdrFtrImageMode2 , 'Align');
121. SetRessourceString(@grs_ps_cbHdrFtrImageMode3 , 'Tile');
122.
123. SetRessourceString(@grs_pst_title , 'Title');
124. SetRessourceString(@grs_ps_titlerow , 'Title (Preview)');
125. SetRessourceString(@grs_ps_titleoneachpage , 'Show title on each page');
126.
127. SetRessourceString(@grs_ps_ganttchart , 'Gantt Chart');
128. SetRessourceString(@grs_ps_wrap , 'Breaks:');
129. SetRessourceString(@grs_ps_rowwrap , 'Avoid row break');
130. SetRessourceString(@grs_ps_colwrap , 'Avoid column break');
131. SetRessourceString(@grs_ps_timescalebelow , 'Show timescale below gantt chart');
132. SetRessourceString(@grs_ps_colheaderoneachpage , 'Column header on each page');
133. SetRessourceString(@grs_ps_printcolumnoneachpage , 'Print columns on each page:');
134. SetRessourceString(@grs_ps_timerange , 'Time range:');
135. SetRessourceString(@grs_ps_timerangeall , 'All');
136. SetRessourceString(@grs_ps_timerangetimerange , 'Time range');
137. SetRessourceString(@grs_ps_timerangestart , 'Start:');
```

```
SetRessourceString(@grs_ps_timerangeend , 'Finish:');

SetRessourceString(@grs_ps_legendimages , 'Legend and Images');
SetRessourceString(@grs_ps_images , 'Images:');
SetRessourceString(@grs_ps_preview2 , 'Preview:');
SetRessourceString(@grs_ps_add , 'Add...');
SetRessourceString(@grs_ps_delete , 'Delete...');
SetRessourceString(@grs_ps_delete2 , 'Delete');
SetRessourceString(@grs_ps_deleteall , 'Delete All');
SetRessourceString(@grs_ps_modify , 'Modify...');
SetRessourceString(@grs_ps_properties , 'Properties:');
SetRessourceString(@grs_ps_properties2 , 'properties');
SetRessourceString(@grs_ps_align , 'Align to:');
SetRessourceString(@grs_ps_aligntopt1 , 'Top Left');
SetRessourceString(@grs_ps_aligntopt2 , 'Top Right');
SetRessourceString(@grs_ps_aligntopt3 , 'Bottom Right');
SetRessourceString(@grs_ps_aligntopt4 , 'Bottom Left');
SetRessourceString(@grs_ps_alignauto , 'Automatically');
SetRessourceString(@grs_ps_height , 'Height:');
SetRessourceString(@grs_ps_width , 'Width:');
SetRessourceString(@grs_scaleproportional , 'Proportional');
SetRessourceString(@grs_adjusttopagesize , 'Adapt when paper format has been modified');

SetRessourceString(@grs_borderbackground , 'Border and Background');
SetRessourceString(@grs_background , 'Background');
SetRessourceString(@grs_backgroundnone , 'None');
SetRessourceString(@grs_backgroundfarbe , 'Color:');
SetRessourceString(@grs_image , 'Image');
SetRessourceString(@grs_border , 'Border');
SetRessourceString(@grs_printborder , 'Print border');
SetRessourceString(@grs_pixel , 'Pixel');
SetRessourceString(@grs_color , 'Color:');

//TfrmGanttPrintPreview_2
SetRessourceString(@grs_ppv_printpreview , 'Print Preview');
SetRessourceString(@grs_ppv_menu_document , 'Document');
SetRessourceString(@grs_ppv_menu_view , 'View');
SetRessourceString(@grs_ppv_menu_print , 'Print...');
SetRessourceString(@grs_ppv_menu_print2 , 'Print');
SetRessourceString(@grs_ppv_menu_pagesetup , 'Setup page...');
SetRessourceString(@grs_ppv_menu_close , 'Close');
SetRessourceString(@grs_ppv_menu_firstpage , 'First page');
SetRessourceString(@grs_ppv_menu_previouspage , 'Previous page');
SetRessourceString(@grs_ppv_menu_nextpage , 'Next page');
```

```
182. SetRessourceString(@grs_ppv_menu_lastpage , 'Last page');
183. SetRessourceString(@grs_ppv_menu_thumbnails , 'Thumbnails');
184. SetRessourceString(@grs_ppv_paperformatandorientation , 'Page layout and orientation:');
185. SetRessourceString(@grs_ppv_settingsganttchart , 'Settings - gantt chart:');
186. SetRessourceString(@grs_ppv_timescale , 'Timescale');
187. SetRessourceString(@grs_ppv_timescalezoom , 'Timescale - Zoom(%)');
188. SetRessourceString(@grs_ppv_timescaleadjust , 'Adjust');
189. SetRessourceString(@grs_ppv_printer , 'Printer:');
190.
191. SetRessourceString(@grs_ppv_timescale_days , 'Days');
192. SetRessourceString(@grs_ppv_timescale_days2 , 'day(s)');
193. SetRessourceString(@grs_ppv_timescale_weeks , 'Weeks');
194. SetRessourceString(@grs_ppv_timescale_months , 'Months');
195. SetRessourceString(@grs_ppv_timescale_years , 'Years');
196.
197. SetRessourceString(@grs_ppv_printscaling , 'Print - Scaling:');
198. SetRessourceString(@grs_ppv_printablecolumns , 'Printable Columns:');
199. SetRessourceString(@grs_ppv_defineprintablecolumns ,'Define printable columns...');
200. SetRessourceString(@grs_ppv_increasezoom , 'Increase zoom');
201. SetRessourceString(@grs_ppv_decreasezoom , 'Decrease zoom');
202. SetRessourceString(@grs_ppv_adjustzoom , 'Adjust zoom');
203. SetRessourceString(@grs_ppv_zoomtopagewidth , 'Zoom to page width');
204. SetRessourceString(@grs_ppv_zoomallpages , 'All pages');
205. SetRessourceString(@grs_ppv_singlepage , 'Single page');
206. SetRessourceString(@grs_ppv_printprofile , 'Print profile:');
207. SetRessourceString(@grs_ppv_newprofile , 'Create new print profile');
208. SetRessourceString(@grs_ppv_saveasnewprofile , 'Save current print settings');
209. SetRessourceString(@grs_ppv_renameprofile , 'Rename print profile');
210. SetRessourceString(@grs_ppv_deleteprofile , 'Delete print profile');
211.
212. SetRessourceString(@grs_ppv_showadditionaldays , 'Show additional days:');
213. SetRessourceString(@grs_ppv_deleteprintprofilecnf_pr , 'Do you realy want to delete the profile
     ');
214. SetRessourceString(@grs_ppv_deleteprintprofilecnf_sf , ' ?');
215. SetRessourceString(@grs_ppv_copyof , 'Copy of ');
216.
217. SetRessourceString(@grs_ppv_toomanypages1 , 'Too many pages');
218. SetRessourceString(@grs_ppv_toomanypages2p , 'There will be ');
219. SetRessourceString(@grs_ppv_toomanypages2s , ' pages, with to current settings.');
220. SetRessourceString(@grs_ppv_toomanypages3 , 'Please select a larger paper format or minimize the
      page borders.');
221.
222. // ganttbarconnections.pas
223. SetRessourceString(@grs_ganttbarconnections_updateerr1 , 'Task in row');
224. SetRessourceString(@grs_ganttbarconnections_updateerr2 , ' Cannot be moved.');
```

```
225.
226. // ganttbarinfopanel.pas
227. SetRessourceString(@grs_ganttbarinfopanel_cancelaction, '          Press        to cancel the
     action.');
228.
229. // ganttbasebars.pas
230. SetRessourceString(@grs_gbb_barfixed, '(fixed)');
231. SetRessourceString(@grs_fbb_from, 'from ');
232. SetRessourceString(@grs_fbb_till, ' to  ');
233. SetRessourceString(@grs_fbb_dauer, 'Duration');
234. SetRessourceString(@grs_fbb_criticalpath, 'Is part of the @b@1critical path');
235.
236. //ganttcolumns.pas
237. SetRessourceString(@grs_currency_default, '$');
238.
239. //ganttconsts.pas
240. SetRessourceString(@grs_calendarweek,'. CW');
241.
242. //ganttdatacontroler.pas
243. SetRessourceString(@grs_collision, 'Task in row %s can not be rescheduled.');
244. SetRessourceString(@grs_collision2, 'Task in row %s could not be extended.');
245. SetRessourceString(@grs_warn_downgrade, 'By downgrading, the row %s will be converted into a sum
     mary row. This row already contains planed activities. Do you want to replace them with a summar
     y row?');
246.
247. //ganttdrawer.pas
248. SetRessourceString(@grs_nopertbarmsg1, 'Drag a rectangle');
249. SetRessourceString(@grs_nopertbarmsg2, 'to create a new task.');
250.
251. // ganttprintpreview.pas
252. SetRessourceString(@grs_noprinteravailable_long, 'No printers found on your system.');
253.
254. // ganttrows.pas
255. SetRessourceString(@grs_lc_hour,  'hour' );
256. SetRessourceString(@grs_lc_hours, 'hours' );
257. SetRessourceString(@grs_lc_day,   'day' );
258. SetRessourceString(@grs_lc_days,  'days' );
259. SetRessourceString(@grs_lc_week,  'week' );
260. SetRessourceString(@grs_lc_weeks, 'weeks' );
261.
262. //ganttsuite.pas
263. SetRessourceString(@grs_create_new_task, 'Create a new task' );
264. SetRessourceString(@grs_cannotcreatetask, 'Task in row %s can not be created.');
265. SetRessourceString(@grs_hintinfodesc, 'Double-Click to edit row note.');
266. SetRessourceString(@grs_movetask, 'Reschedule Task');
```

```
267. SetRessourceString(@grs_movetaskstart, 'Start        ');
268. SetRessourceString(@grs_movetaskfinish, 'Finish    ');
269. SetRessourceString(@grd_movetaskduration, 'Duration');
270. SetRessourceString(@grs_RescheduleStart,'Reschedule Start');
271. SetRessourceString(@grs_RescheduleFinish, 'Reschedule Finish');
272. SetRessourceString(@grs_pertchart,'PERTCHARTVIEW');
273. SetRessourceString(@grs_pertchartsub,'Pert view of the gantt chart');
274.                                     //
275. SetRessourceString(@grs_nobaronsummrow, 'No bars can be inserted on summary rows.');
276.
277. // Connections Bar Button
278. SetRessourceString(@grs_create_con1_cap,'Create connection (start)');
279. SetRessourceString(@grs_create_con1_hnt,'Click the button and move the mouse to another bar to c
     reate a connection');
280. SetRessourceString(@grs_create_con2_cap,'Create connection (finish)');
281. SetRessourceString(@grs_create_con2_hnt,'Click the button and move the mouse to another bar to c
     reate a connection');
```

Events

The Gantt Control Components defines severall events that are accessible by Delphi. To access an event please select the TGanttChart component in the delphi formular designer and press [F11] to focus the Object inspector. After that please select the tab sheet “events”. Here you can see all the events to component provides.

The following table shows all events of the gantt chart component and a brief description.

OnAfterBarAdd(Bar: TGanttBaseBar)
The OnAfterBarAdd event is fired after a bar has been added to the component by code or per user interaction.

OnAfterBarChangeRow(Bar: TGanttBaseBar)
The event OnAfterBarChangeRow is fired when a bar has been moved vertically to a new row.

OnAfterBarDelete(Sender: TObject)
The event OnAfterBarDelete will be triggered after a bar has been deleted.

OnAfterBarMove(Bar: TGanttBaseBar)
The event OnAfterBarMove will be triggered after a bar has been moved.

OnAfterBarResize(Bar: TGanttBaseBar)
The event OnAfterBarResize will be triggered after a bar has been resized.

OnAfterConnectionAdd(Connection: TBarConnection)
The event OnAfterConnectionAdd will be triggered after a connection has been added.

OnAfterConnectionDelete(Sender: TObject)
The event OnAfterConnectionDelete will be triggered after a connection has been deleted.

OnAfterDragNewBar(Bar: TGanttBaseBar)

The event OnAfterDragNewBar will be raised after the user has created and dragged a new bar.

OnAfterDragNewConnection (Sender: TObject; BarFrom, BarTo: TGanttBaseBar; AConnectionType: TBarConnectionType; var ACreateConnection, AAutoResolveConnection: Boolean)

The event OnAfterDragNewConnection will be raised after the end-user has created a new connection between bars.

OnBarMouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer)

The event OnBarMouseDown will be raised after the user has clicked on a gantt bar object. The parameter Sender contains a reference to the clicked bar.

OnBarMouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer)

After the user releases a pressed mouse button while the cursor is over a gantt bar the OnBarMouseUp event will be fired. The parameter Sender specifies the bar that is under the mouse cursor while the mouse button is released. The parameter Button (mbLeft, mbRight, mbDouble) determines the mouse button that was released.

OnBeforeBarAdd(Bar: TGanttBaseBar; var Cancel: Boolean)

The event OnBeforeBarAdd will be raised before a new bar object will be added to the gantt chart component. The parameter Bar holds a reference to the bar object that will be added. If you set true to the var parameter Cancel, the bar will not be added to the component.

OnBeforeBarChangeRow(Bar: TGanttBaseBar; var Cancel: Boolean)

The event OnBeforeChangeRow will be raised before the bar object will be reassigned to a new row. You can cancel the action if you set true to the Cancel parameter.

OnBeforeBarDelete(Bar: TGanttBaseBar; var Cancel: Boolean)

The event OnBeforeBarDelete will be raised before a bar object will be deleted. You can cancel the deletion of the bar, if you set true to the Cancel parameter.

OnBeforeBarMove(Bar: TGanttBaseBar; var Cancel: Boolean)

The event OnBeforeBarMove will be raised before a bar will be rescheduled by the end-user. You may prevent the rescheduling of the bar if you set the parameter Cancel to true.

OnBeforeBarResize(Bar: TGanttBaseBar; var Cancel: Boolean)

The event OnBeforeBarResize will be raised before a bar will be resized. You can cancel the action if you set true to the Cancel parameter.

OnBeforeConnectionAdd(Connection: TBarConnection; var Cancel: Boolean)

The event OnBeforeConnectionAdd will be raised before a connection will be added between two bar objects. You can cancel the creation of the connection if you set the Cancel parameter to true.

OnBeforeConnectionDelete (Connection: TBarConnection; var Cancel: Boolean)

The event OnBeforeConnectionDelete will be raised before a connection will be deleted. You can cancel the deletion of the bar if you set the Cancel parameter to true.

OnCellAllowEdit (Cell: TGanttCell; var CanEdit: Boolean)

The event OnCellAllowEdit will be raised before a cell enters the edit mode. Within this event you can specifiy whether the cell should be editable by assigning true or false to the CanEdit parameter.

OnCellClick (Cell: TGanttCell)

The event OnCellClick will be raised when the user clicks on a cell. The parameter Cell contains the clicked cell object.

OnCellFinishEditing (Cell: TGanttCell)

The event OnCellFinishEditing will be raised after a cell value has been edited. If the user cancels the editing of a cell by pressing [ESC] the event will not be triggered. The parameter Cell contains a reference to the edited cell object of the gantt chart component.

OnCellMouseDown (Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer)

The event OnCellMouseDown will be raised when the users clicks a cell. The parameter Sender holds a reference to the cell.

OnCellMouseUp (Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer)

The event OnCellMouseUp will be raised when the users releases a mouse button over a cell.

OnCellStartEditing (Cell: TGanttCell)

The event will be raised when the user starts the editing of a cell. Cell holds a reference to the cell object.

OnCellValidate (Cell: TGanttCell; var AValue: Variant; var AText: string)

The OnCellValidate event will be raised after the user has edited a cell. The event provides the edited value – both as a variant type and the cells text. You may modify the cells value and display text within this event.

OnClick (Sender: TObject)

The OnClick event is raised when the user clicks on the gantt chart component.

OnColumnCancelFilter (Sender: TObject)

The event OnColumnCancelFilter will be raised when the filtering of rows based on a column filter will be canceled.

OnColumnCancelSort (Sender: TObject)

The event OnColumnCancelSort will be raised when the column sorting will be canceled.

OnColumnFiltered (Sender: TObject)

The event OnColumnFiltered will be raised when a column filter has been applied.

OnColumnHeaderMouseDown (Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer)

The event OnColumnHeaderMouseDown will be raised when the end-user clicks on a column header.

OnColumnHeaderMouseUp (Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer)

The event OnColumnHeaderMouseUp will be raised when the end-user releases the mouse button over a gantt column header.

OnColumnResized (Sender: TObject)

The event OnColumnResized will be raised after the user has resized the width of a column. The parameter Sender contains the gantt column.

OnColumnSort (Column: TGanttColumn; ASortOrder: TColumnSortOrder)

The event OnColumnSort will be raised after a sorting has been applied to a column.

OnColumnTitleEdited (Column: TGanttColumn; var NewTitle: string; OldTitle: string)

The event OnColumnTitleEdited will be raised when the title caption has been edited by the end-user.

OnConnectionMouseDown (Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer)

The event OnConnectionMouseDown will be raised when the user clicks on a bar connection. The parameter Sender provides access to the clicked connection.

OnCustomDrawBar (Bar: TGanttBaseBar; OpBar: Boolean; ACanvas: TCanvas; ARect: TRect; var ADone: Boolean)

The event OnCustomDrawBar will be raised when a bar object is drawn by the component. You can use the OnCustomDrawBar event to customize and implement your own drawing mechanism for bar objects. Bar holds a reference to the bar object that is drawn. OpBar is true for the drawing of temporal bars that visualizes a preview of modified bars. For example if you move a bar with the mouse to a new start date both a temporal bar and the bar itself will be drawn. The ACanvas parameter provides access to the bars canvas. ARect specifies the extension of the bar in pixel. ADone must be set to true after you have finished your own drawing routine.

OnCustomDrawCell (Cell: TGanttCell; AColumnIndex, AbsoluteRowIndex: Integer; ACanvas: TCanvas; ARect: TRect; var ADone: Boolean)

The event OnCustomDrawCell will be raised when a cell object is drawn by the component. You can use the OnCustomDrawCell event to customize and implement your own drawing mechanism for cells. Cell holds a reference to the cell object that is drawn. AColumnIndex specifies the index of the refering column. AbsoluteRowIndex specifies the absolute row index of the corresponding row. ACanvas provides access to the cells drawing canvas. ARect specifies the cells extension. ADone must be set to true after you have finished your own drawing mechanism.

OnCustomDrawHeader (Column: TGanttColumn;ACanvas: TCanvas; ARect: TRect; var ADone: Boolean)

The event OnCustomDrawHeader will be raised when a header object is drawn by the component. You can use the OnCustomDrawHeader event to customize and implement your own drawing mechanism for column headers.

OnDataModified (Sender: TObject; Action: string)

The event OnDataModified will be raised as soon as any data of the gantt chart component has been modified.

OnDblClick (Sender: TObject)

The event OnDblClick will be raised when a user double clicks on the gantt chart component.

OnDragDrop (Sender, Source: TObject; X, Y: Integer)

The event will be raised if the user drag drops an object to the gantt chart component.

OnDragOver (Sender, Source: TObject; X, Y: Integer; State: TDragState; var Accept: Boolean)

The event will be raised if the user drags an object over the gantt chart component.

OnEndDock (Sender, Target: TObject; X, Y: Integer)

The event will be raised when the dragging of an objects ends.

OnEndDrag (Sender, Target: TObject; X, Y: Integer)

The event will be raised when the dragging of an objects ends.

OnEnter (Sender: TObject)
The event will be raised when the gantt chart component gets the focus.

OnExit (Sender: TObject)
The event will be raised when the gantt chart components loses his focus.

OnGanttHorizontalScroll (Sender: TObject; Position: Integer)
The event OnGanttHorizontalScroll will be raised when the users scrolls the gantt chart view horizontally.

OnGanttObjectMouseDown (Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer)
The event OnGanttObjectMouseDown will be raised when the users clicks a sub element of the gantt chart component.

OnGanttVerticalScroll (Sender: TObject; Position: Integer)
The event OnGanttVerticalScroll will be raised when the users scrolls the gantt chart view vertically.

OnIndicatorActionAutoSchedule (Sender: TObject)
The event OnIndicatorActionAutoSchedule will be fired when the user clicks an indicator item thats AutoBehaviour has been set to giaAutoSchedule.

OnIndicatorActionEditNote (Sender: TObject)
The event OnIndicatorActionEditNode will be fired when the user clicks an indicator item thats AutoBehaviour has been set to giaNote.

OnIndicatorActionGoToBar (Sender: TObject)
The event OnIndicatorActionGoToBar will be fired when the user clicks an indicator item thats AutoBehaviour has been set to giaGoToBar.

OnIndicatorColumnHeaderMouseDown (Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer)
The event OnIndicatorColumnHeaderMouseDown will be fired when the user clicks in the indicator columns header.

OnIndicatorColumnMouseDown (Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer)
The event OnIndicatorColumnMouseDown will be fired when the user clicks on a cell of the indicator column.

OnIndicatorColumnResized (Sender: TObject)
The event OnIndicatorColumnResized will be fired when the user resizes the width of the indicator column.

OnIndicatorItemGetHint (AIndicatorItem: TIndicatorItem; ARow: TGanttRow; AColumn: TGanttColumn; var AHintTitle, AHintCaption: string)
The event OnIndicatorGetHint will be fired before the gantt chart component displays the hint and the description of the gantt chart component.

OnIndicatorItemMouseDown (AIndicatorItem: TIndicatorItem; ARow: TGanttRow; AColumn: TGanttColumn; Button: TMouseButton; Shift: TShiftState; X, Y: Integer)
The event OnIndicatorMouseDown will be raised when the user clicks on an IndicatorItem of an Indicator column.

OnKeyDown (Sender: TObject; var Key: Word; Shift: TShiftState)

The event OnKeyDown will be raised when the user presses any key while the gantt chart has the focus.

OnKeyPresss (Sender: TObject; var Key: Char)

The event OnKeyPress will be raised when a key corresponding to an ASCII character has been pressed and the gantt chart component is focused.

OnKeyUp (Sender: TObject; var Key: Word; Shift: TShiftState)

The event OnKeyUp will be raised when the user releases a key.

OnMouseDown (Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer)

The event OnMouseDown will be raised when the user presses a mouse button on the gantt chart component.

OnMouseEnter (Sender: TObject)

The event OnMouseEnter will be raised when the user moves the mouse cursor above the gantt chart component.

OnMouseLeave (Sender: TObject)

The event OnMouseLeave will be raised when the mouse cursor leaves the area of the gantt chart component.

OnMouseMove (Sender: TObject; Shift: TShiftState; X, Y: Integer)

The event OnMouseMove will be raised when the mouse cursor moves.

OnMouseUp (Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer)

The event OnMouseUp will be raised when a pressed mouse button will be released.

OnMouseWheel (Sender: TObject; Shift: TShiftState; WheelDelta: Integer; MousePos: TPoint; var Handled: Boolean)

The event OnMouseWhell will be raised when the mouse wheel has been scrolled/modifed.

OnMouseWheelDown (Sender: TObject; Shift: TShiftState; MousePos: TPoint; var Handled: Boolean)

The event OnMouseWheelDown will be raised when the mouse wheel will be scrolled downwards.

OnMouseWheelUp (Sender: TObject; Shift: TShiftState; MousePos: TPoint; var Handled: Boolean)

The event OnMouseWheelUp will be raised when the mouse wheel will be scrolled upwards.

OnPertHorizontalScroll (Sender: TObject; Position: Integer)

The event OnPertHorizontalScroll will be raised when the pert chart will be scrolled horizontally.

OnPertVerticalScroll (Sender: TObject; Position: Integer)

The event OnPertVerticalScroll will be raised when the pert chart will be scrolled vertically.

OnPushRedo (Sender: TObject)

The event OnPushRedo will be trigerred when the current state of the component will be pushed to the RedoStack.

OnPushUndo (Sender: TObject)

The event OnPushUndo will be triggered when the current state of the component will be pushed to the UndoStack.

OnRedo (Sender: TObject)

The OnRedo event will be triggered when the gantt chart component restores the changes made from the last undo call.

OnResize (Sender: TObject)

The event OnResize will be called when the component will be resized.

OnRowAdd (Row: TGanttRow)

The event OnRowAdd will be triggered when a new row will be added to the component.

OnRowCollapse (Row: TGanttRow)

The event OnRowCollapse will be triggered when the user collapses an expanded row.

OnRowDelete (Sender: TObject; AbsoluteIndex: Integer)

The event OnRowDelete will be raised when a row has been deleted.

OnRowExpand (Row: TGanttRow)

The eventOnRowExpand will be triggered when the users expands a collapsed row.

OnRowResized (Sender: TObject)

The event OnRowResized will be triggered when the user resizes a row.

OnSelectionChanged (Sender: TObject; SelectedCells: TList<ganttcell.TGanttCell>; SelectedRows: TList<ganttrows.TGanttRow>)

The event OnSelectionChanged will be raised whenever the current selection of objects has been changed. It provides access to both the generic lists SelectedCells and SelectedRows that contain the selected cells and rows.

OnShowIndicatorItem (AIndicatorItem: TIndicatorItem; ARow: TGanttRow; AColumn: TGanttColumn; var AShow: Boolean)

The event OnShowIndicatorItem will be triggered when an indicator item will be renderd. You can specifiy whether it should be visible or not with the AShow parameter.

OnSplitterMoved (Sender: TObject)

The event OnSplitterMoved will be raised after the splitter between the tree grid and the gantt chart area has been moved.

OnStartDock (Sender: TObject; var DragObject: TDragDockObject)

The event OnStartDock will be raised when the user begins to drag a control.

OnStartDrag (Sender: TObject; var DragObject: TDragObject)

The event OnStartDrag occurs when the user begins to drag a control by left-clicking the control and holding the left mouse button down.

OnTimeScaleMouseDown (Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer)

The event OnTimeScaleMouseDown occurs when the user clicks on a time scale object.

OnTreeScroll (Sender: TObject; Position: Integer)
The event OnTreeScroll will be triggered when the user scrolls the tree grid horizontally.

OnUndo (Sender: TObject)
The event OnUndo will be triggered when the component restores the component to a prior state and revert the (last) changes.

OnUndoRedoStackChanged (Sender: TObject)
The event OnUndoRedoStackChanged will be triggered whenever the component adds or deletes elements to the undo stack and the redo stack.

OnViewModified (Sender: TObject; Action: string)
The event OnViewModifed will be triggered when the user changes the layout of the gantt chart, e.g. scrolls the gantt chart or modifies the splitter position.