

OGantt v4.0

by Brainy Data Limited

About OGantt

Introduction

OGantt is a cross-platform GANTT chart component for Omnis Studio. As well as the typical GANTT style chart, OGantt also provides a graph-style resource panel that can be used to display current use of resources.

History of Changes

Installing the Software

A number of components have been provided for the various platforms and versions of Omnis Studio. It is vital that the correct external component is copied to the Omnis tree otherwise the software may not function or may crash.

Different release trees will be provided for different platforms. These are typically called “ogantt_v400r_mac.zip”, “ogantt_demo_software_mac.zip” or “ogantt_v400r_win.zip”, etc.

Within a release tree, you will find unicode and non-unicode folders, and within each of these you will find folders indicating different versions of Omnis Studio. It may be that your version of Studio is not listed in which case you should use the latest version that is not greater than the version of Studio that you are using. If your version of Studio is older than any that are listed, the software may not be compatible with your version of Studio.

To install the software

- Open the appropriate platform folder and navigate to the folder appropriate for your version of Studio.

Within this folder you will find up to three further folders called “xcomp”, “webcomp” and “__webclient”. The *webcomp* and *__webclient* folders are only provided if the software supports the Omnis Web Client.

- Copy the external components from inside *xcomp* and *webcomp* to the Omnis tree. You will find identical named folders inside the Omnis tree. On Mac OSX these can be found inside the *Omnis bundle->Content->MacOS*.
- To install the web client component you can load it into your Omnis download manager. This tool is available of the *Tools Menu->Web Client Tools*.

You can run the example library directly from the provided examples folder. If you move the example library from this folder, you must also move all other files and folders contained within. OGantt Version 4 includes separate examples for Studio 5 as this is the minimum version of Studio that is required to run the web-client implementation of OGantt.

Deploying the software

Please refer to the license agreement for rules on deployment.

Documentation

This documentation describes the functionality provided by OGantt. It is recommended that you at least read the chapter “Designing OGantt”.

As a minimum we recommend that you read the following Chapters and technical notes:

OWrite: Introduction and Designing OGantt.

History

Below is a summary of the most recent enhancements.

Version	Enhancements
4.0.0	<p>This beta release includes web-client and fat-client components for MS Windows and Macintosh. In addition there are new OGantt examples that demonstrate some of the new features. In places where new code has been added to example classes, we have added the comment "CHANGE_V400a1", "CHANGE_V400a2" as well as CHANGE_yyyy_mm_dd where yyyy stands for the year, mm for the numeric month and dd stands for the date. This format allows the searching for changes after a specific date.</p> <p>Web-Client Support</p> <p> See Web-Client Support in Designing OGantt</p> <p>Editing tree list cells</p> <p> See Interacting With OGantt in Designing OGantt</p>
	<p>Drag & Drop</p> <p>The following new properties and constants have been implemented to aid the dropping of Omnis objects or data onto the OGantt component.</p> <p>Properties:</p> <p> \$tddcandrop, \$gddcandrop, \$ddwhere, \$ddresourceid, \$ddphaseid.</p> <p>Constants for \$tddcandrop and \$gddcandrop:</p> <p> kGanttCanDropBetweenRows, kGanttCanDropOnRow, kGanttCanDropOnPhase.</p> <p>Constants for \$ddwhere:</p> <p> kGanttDropBefore, kGanttDropAfter, kGanttDropTreeList, kGanttDropGantt, kGanttDropPhase.</p> <p>Drag & Drop phases between resources</p> <p> New property \$gphasecandrop and new event evPhaseCanDrop. Updated event evGMove.</p> <p>New display scales</p> <p> The following new scales have been added and can be used with the \$gsetscale method for the minor scale.</p> <p> kGanttScale15Minute, kGanttScale2Hour and kGanttScale6Hour.</p>
	<p>New/modified properties:</p> <p> Modified \$vscroll and \$hscroll.</p> <p> New property \$ignorerelations.</p>

	<p>New/modified methods:</p> <ul style="list-style-type: none"> New markers behind parameter for \$setmarkerlist() plus new \$getmarkerlist() method. Date range for \$getbitmap(). Get first visible date \$ggetstartdate(). ISO Week support and change start of week. See \$gsetscale().
	<p>New/modified events</p> <ul style="list-style-type: none"> New event parameters for evGDbClick and evGRClick. Background clicks for evGSelected and evTLSelected.
3.8.0	<ul style="list-style-type: none"> • New properties \$gridhlinestyle, \$gridvlinestyle, \$gridhlinecolor, \$gridvlinecolor and \$grellinestyle control the appearance of the grid and relationship lines. • New properties \$grscalefontname, \$grscalefontsize, \$grscalefontstyle and \$grscalecolor control the appearance of the graph's scale. • New 6th parameter for \$GSetScale() for setting the snap time when moving or resizing phases. • New event parameter pColumnNumber for the evTLDbClick, evTLRClick and evTLSelected events. • The methods \$tlsetselectresource() and \$gsetselectphase() can now be used to deselect the resource and phase.
3.7.3	<ul style="list-style-type: none"> • Ability to specify date formats for tool tips. See \$gsettooltiptext().
3.7.2	<ul style="list-style-type: none"> • New property \$sprintposmode.
3.7.0	<ul style="list-style-type: none"> • New parameters for \$gsetscale(). • New property \$swapclientarea. • New properties \$scalborder, \$scalbordercolor, \$scalfillcolor, \$headborder, \$headbordercolor and \$headfillcolor.

Introduction

Overview

The OGantt software consists of the following parts

1. The OGantt examples in the “Examples” folder.
2. The external component library, “ogantt.dll” on Windows, “ogantt.xcomp” on Macintosh, and “ogantt.so” on Linux.

For an introduction on how to integrate the OGantt software into your library, please read the chapter Designing OGantt

Examples

There are a number of classes that demonstrate the use of the OGantt software. These classes are mainly concerned with the OGantt interface and consist of menus, windows and object classes. For a description of the example classes please read the chapter Examples Reference

External component Library

The external component library consists of the visual window control, a non visual object for managing GANTT data and structures, and a report object for printing GANTT charts. A number of constants are defined for the use with properties and methods of the component objects. These constants can be accessed from the Omnis Catalog -> Constants -> OGantt. For a description of the external components please read the chapter External Component Reference

Designing OGantt

Introduction

You should read this entire chapter before attempting to develop OGantt.

This chapter gives a brief description of what is involved to add OGantt to your libraries. Please use this chapter together with the example library.

For a more detailed description of the example classes and external components please read the chapters Examples Reference and External Component Reference.

Contents

How OGantt Works - a brief description of a GANTT chart and the components that are provided by OGantt.

Building a GANTT chart - a detailed description about the data structures that OGantt requires. These examples can be used to create new empty charts or convert existing project data.

Saving and Loading GANTT charts - strategies for storing GANTT charts in your database.

Interacting with OGantt - about events and how to interact with the window object.

Web-Client Support - a few notes about the web-client support for OGantt.

Translation - a brief description about the use of string tables in the OGantt examples.

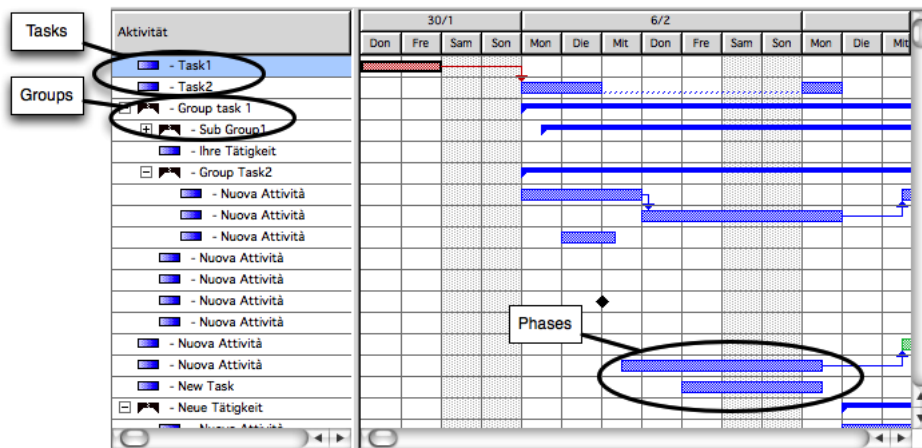
Printing - how to print a GANTT chart.

Further Reading - suggested further reading

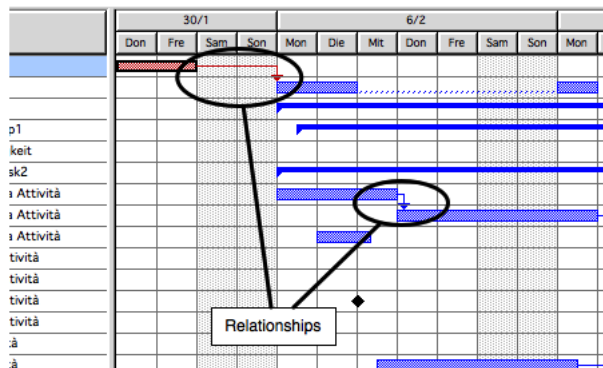
How OGantt Works

The GANTT chart

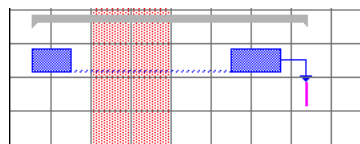
Essentially, a GANTT chart can be divided into categories, resources, and phases or tasks. A category is a group of resource that encapsulates a number of child resources and their phases. Categories and resources are represented as individual rows in the GANTT tree and chart. Phases are represented as individual bars in the GANTT chart and always belong to a resource. A resource may contain more than one phase.



Individual phases can be assigned a relationship to other phases within the same resource or phases of other resources. There are different types of relationships, such as finish-to-start, start-to-start, finish-to-finish, etc. Changing the time frame of one phase may effect other phases that are related.



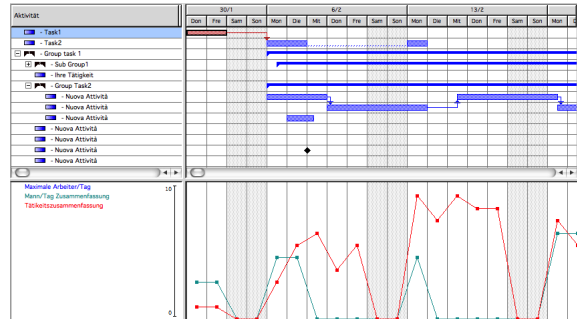
An individual phase may also be suspended for a period of time. This period is displayed as a dotted line between the broken bar.



In addition, phases can specify extra data that is represented in a line graph or as totals below the GANTT chart area. It is entirely up to you what this data represents. For example, one could

www.brainydata.com

specify the cost of a phase on a daily basis, as well as required resources, i.e. workman, goods, etc.



External Objects

There are three objects contained in the GANTT component. They will be referred to as the Chart object (*Gantt.GanttUI*), which is a visual component that can be added to a window to display a GANTT chart, the Layer object (*Gantt.GanttObject*), which is a non-visual object that holds the data to be represented in the chart, and the Report object (*Gantt.GanttPrint*) which is used to print GANTT charts in an Omnis report.

The first step is to set up the visual Chart object with headings and working hours, then to set up a Layer object with lists in the appropriate format to hold data. Once these lists have been populated, the Layer object is passed to the Chart object for display, and the GANTT Chart Component is redrawn.

Building a GANTT chart

This section describes in detail the structure of the data required by a GANTT chart. It does not describe how one what typically load or save a GANTT chart, but the knowledge covered here together with the example code may prove useful if you already have project data in a structure that does not conform to the structure required by OGantt.

Preparing and loading data into the GANTT component involves building a number of lists and assigning them to the non-visual layer object and the GANTT window object.

To simplify this process, we have divided it into these distinct tasks.

1. Load the custom text for the interface
2. Initialise the calendar details of the GANTT component, such as working hours, and holidays.
3. Preparing and loading data for the left-hand tree list.
4. Preparing and loading data for the right-hand GANTT chart.
5. Assign layer object to the GanttUI window object and build the chart.

Please use the GanttExample.lbs in conjunction with this text. In this library we have sub-classed the external non-visual *Gantt.GanttObject* (see object class oGanttObject) and implemented the code described here. If possible, use this object class in your own library to aid with the loading and saving of a GANTT chart.

1. Load the custom text for the interface

Before displaying data in the GANTT chart, it is necessary to set terminology for the day and month headings, how totals are calculated, the legend list for totals and graph display, and the tool-tip text in the right-hand chart.

This should be done just once for each window object during construct. So the ideal place for placing this call is the construct method of the window or a method that it calls.

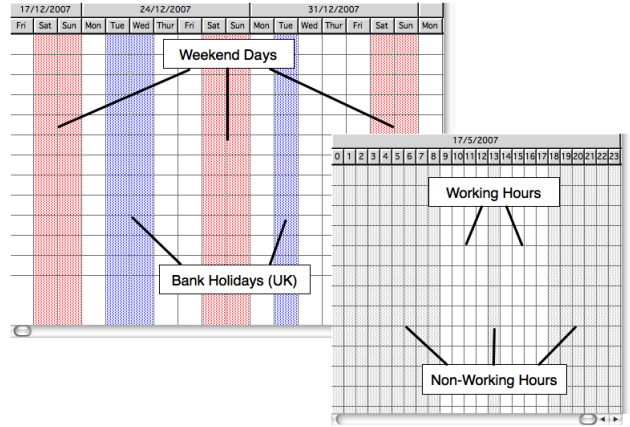
In our examples, the legend list and total calculations are hard-coded and therefore the same for all charts. However, there is no reason why these cannot be user defined and stored in the DB as part of the GANTT chart data.

Example Code: `oGanttObject.$load_custom_text`

2. Initialising the Calendar details

Before displaying data in the GANTT chart, it is necessary to supply information about which hours and days to display as working time and which days are holidays.

The method `oGanttObject.build_calendar` demonstrates how we set-up lists for working hours, public holidays, and exceptions to working hours for specific days of the year.



Working Hours

Setting up working hours is a simple matter of creating a list with two columns that specify the start and end times for each working period in a day.

List definition of the working hours list. One row per range of working hours.		
Column	Type	Description
From	Short Time	Specifies the starting time of this time range
To	Short Time	Specifies the end of this time range

Example Code: `oGanttObject.build_calendar`

Working Hours Exceptions

We can also set exceptions to working hours for specific days. For example, it may be company policy to always finish work at 1pm on the 24th of December. To set exceptions we create another list that takes a date and a working hours list for each individual day that has different working hours.

List definition of the working hours exception list. One row per day.		
Column	Type	Description
Date	Date	The date for the exceptions
WorkingHours	List	List of working hours for that day. As defined above.

Example Code: `oGanttObject.build_calendar`

Holidays

Weekend days and other holidays are specified by another list. The structure of this list is a little more complex.

List definition of the holidays list. One row per day.		
Column	Type	Description
Type	Long Integer	Specifies the type of holiday and can be one of the kGanttHol... constants.
Date	Date	Specifies the date for fixed day holidays when Type is kGanttHolFixedDay.
Day	Short Integer	Specifies the day number 0..6 (Sunday..Saturday) for weekly non-working days when Type is kGanttHolWeekDayRep.
Year	Date	Specifies the date for annually repeated holidays that always fall on the same day when Type is kGanttHolYearDayRep.
Color	Long Integer	Specifies the background color for the holiday.

Example Code: `oGanttObject.build_calendar`

Finally

Once all the lists are build, we assign them to the layer object using the method `$gsetcalendar`.

Example Code: `oGanttObject.build_calendar`

3. Loading the Left-Hand Tree

The left-hand tree list of the window component is capable of displaying a number of custom columns as well as the required tree list column. To prepare the left hand tree we have to tell it about the columns it is to display and provide it with the data to be displayed. The method `oGanttObject.build_tree` demonstrates how we can load the tree.

Categories	Start Date	End Date	Description
Main Project	1 SEP 2008 09:...	12 SEP 2008 17:...	Description for m...
Preparation	1 SEP 2008 09:...	3 SEP 2008 17:...	Description for p...
Task 1	1 SEP 2008 09:...	1 SEP 2008 17:...	Description for t...
Task 2	2 SEP 2008 09:...	2 SEP 2008 17:...	Description for t...
Task 3	2 SEP 2008 09:...	3 SEP 2008 17:...	Description for t...
Venue	4 SEP 2008 09:...	10 SEP 2008 17:...	Description for v...
Opening	4 SEP 2008 10:...	4 SEP 2008 16:...	Description for o...
Part 1	5 SEP 2008 10:...	7 SEP 2008 16:...	Description for p...
Part 2	8 SEP 2008 10:...	9 SEP 2008 16:...	Description for p...
Close	10 SEP 2008 10:...	10 SEP 2008 16:...	Description for cl...
Vacate	11 SEP 2008 09:...	12 SEP 2008 17:...	Description for v...
Task 1	11 SEP 2008 09:...	11 SEP 2008 17:...	Description for t...
Task 2	12 SEP 2008 09:...	12 SEP 2008 17:...	Description for t...
Non-Venue Activity	1 SEP 2008 09:...	12 SEP 2008 17:...	Description for n...

Column Headers

First we build a list with details about the column headers for the tree. In that list we specify things such as the default, maximum and minimum column width, justification, icon, text color, and the text.

List definition of the column headers list. One row per column.		
Column	Type	Description
Width	Long Integer	Default width of column in pixels
MinWidth	Long Integer	Minimum width of column in pixels
MaxWidth	Long Integer	Maximum width of column in pixels
Icon	Long Integer	Icon ID (0 = no icon)
Align	Short Integer	kLeftJst, kCenterJst or kRightJst
TextColor	Long Integer	RGB color value or one of the Omnis color constants
Text	Character 100	Text to be displayed

Once the list is build we call the method `$lsetheader` to pass the information to the layer object.

Example Code: `oGanttObject.build_tree`

Tree Data

Next we can load the actual data for the tree. We are required to build a list for each column that the tree displays. This is because, as well as the number of columns being optional, together with the data that the tree will display, each cell has its own set of properties such as enterable, text color, alignment etc.

The benefit of this approach is flexibility. Potentially, each individual cell can be displayed using different text color or different icons, and some cells can be edited while others can not, regardless of row or column structure.

The first column of our tree is reserved for the actual tree (Categories) that allow us to expand and collapse groups of resources. So the information that we must provide differs to that of the subsequent custom columns.

List definition of the first tree column. One row per group , sub-group and task.		
Column	Type	Description
Icon	Long Integer	Icon ID (0 = no icon)
Name	Character 100	The text to be displayed
Enterable	Boolean	If kTrue, the Name can be edited (see Editing tree list columns)
IsCategory	Boolean	If kTrue, the item is a group or sub-group
TextColor	Long Integer	RGB color value or one of the Omnis color constants
ParentID	Long Integer	The ResourceID of the parent group or sub-group
ResourceID	Long Integer	The ID of this group, sub-group or task
Order	Number floating dp	The number that determines the order based on which the rows are sorted. The ordering relates only to the siblings of a node and is not absolute for all the nodes of a tree. If not specified, the node is inserted after the last sibling of the specified parent and a order number is automatically assigned.

Once the list is build, we send it to the layer object using the method `$tlsetlist(theList,kGanttListTree)`.

Example Code: `oGanttObject.build_tree`

Custom Column Data

For all subsequent tree columns we must specify one list for each column with the following information.

List definition of the custom tree columns. One row per group , sub-group or task.		
Column	Type	Description
Icon	Long Integer	Icon ID (0 = no icon)
Data	Character 100	The text to be displayed
Enterable	Boolean	If kTrue, the Name can be edited
TextColor	Long Integer	RGB color value or one of the Omnis color constants
ResourceID	Long Integer	The ID of the resource this item belongs to.
Align	Constant (kLeftJst, kCenterJst or kRightJst)	Alignment of the Data in the cell.

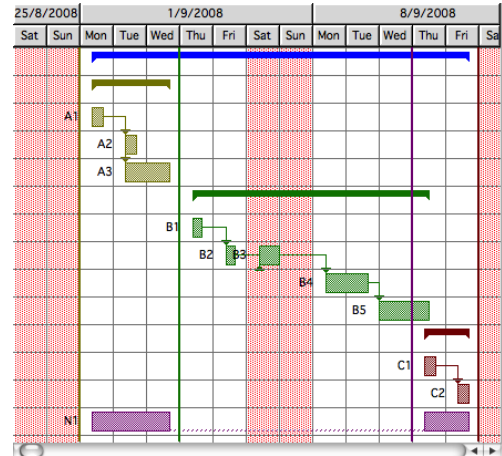
Once all the lists are build, we send them to the layer object using the method `$tlsetlist(theList,kGanttListColumn)`. We must do this once for every column in the correct order.

Example Code: `oGanttObject.build_tree`

4. Loading the Right-Hand Chart

The right-hand chart requires a number of lists to be build.

- The resources list which has the same number of rows and contains some of the same data as the tree list of the left-hand tree.
- The phase list that contains the data for all the phases (bars) for the chart.
- The relation list for specifying the numerous possible relations between all the phases.
- And the suspension list for specifying periods during which phases are suspended.
- Create a legend list for the graph/totals legend.
- Create a marker list for important dates in a project and set the initial display date



The method `oGanttObject.$build_chart` demonstrates how these lists are build and assigned.

Resource List

The Resource List resembles (a little) the Tree List that we build earlier. It represents the hierarchical structure of the GANTT chart and specifies the data for the category groups, sub-groups and resources for that chart.

List definition of the main resource list. One row per group , sub-group or task.		
Column	Type	Description
ParentID	Long Integer	The ResourceID of the parent group or sub-group
ResourceID	Long Integer	The ID of this group, sub-group or task
Order	Number floating dp	The number that determines the order based on which the rows are sorted. The ordering relates only to the siblings of a node and is not absolute for all the nodes of a tree. If not specified, the node is inserted after the last sibling of the specified parent and a order number is automatically assigned.
IsCategory	Boolean	If kTrue, the item is a category group or sub-group
WorkingHours	List	Working hours for this resource (can be NULL)
Holidays	List	Holidays for this resource (can be NULL)
WorkingHoursExceptions	List	Exceptions to working hours (can be NULL)

Once the list is build we call the method `$gsetlist(theList,kGanttListResource)` to send it to the layer object.

Example Code: `oGanttObject.build_chart`

Phase List

The Phase list defines the bars which appear in the chart on the right-hand side of the component. It should contain one line per individual bar in the GANTT chart, as well as one line per category bar. If multiple, separately selectable bars should appear on the same row of the GANTT chart, there should be one line for each. However, if an individual bar is to be subdivided (in which case it is still selectable as a single entity, although subdivisions can be resized individually) then there should be only one line for the whole bar. Divisions are specified using suspensions.

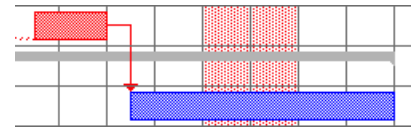
List definition of the phase list. One row per phase.		
Column	Type	Description
ResourceID	Long Integer	ID of the task in the resource list
PhaseID	Long Integer	ID of this phase
StartDate	Date Time	Date and time for start of bar
EndDate	Date Time	Date and time for end of bar (use depends on Duration)
PhaseColor	Long Integer	Fill color of the bar
TextColor	Long Integer	Color of any text associated with the bar
TextLeft	Character	Text to the left of the bar
TextRight	Character	Text to the right of the bar
TextTop	Character	Text to the top of the bar
TextBottom	Character	Text to the bottom of the bar
Text	Character	Text within the bar
Progress	Long Integer	Progress of the task in seconds
Duration	Long Integer	Duration of the task expressed in seconds: If Duration < 0 (-1): ignore duration and use StartDate and EndDate for the duration. If Duration = 0: ignore EndDate, use StartDate for both start and end. If Duration > 0: it specifies the duration. EndDate is calculated from StartDate plus Duration
DurationType	Long Integer	Type of duration: kGanttTypeDurFix: fixed duration (bars + suspension periods). kGanttTypeDurEff: effective duration (sum of the bars excluding suspension periods).
IsLocked	Boolean	If kTrue the task is locked and it is not possible to move it.
IsCategory	Boolean	If kTrue it represents a phase with all attributes of a category but this phase isn't shown in the graph.

FieldList	List	<p>List of the fields for the calculations of the graph and totals. The number of lines is not limited. The list must be defined as follows:</p> <p>Value (long): value of the field.</p> <p>ToSplitValue (bool): if kTrue the value is evenly distributed for each day of the task. If kFalse each day of the task is assigned the given value.</p>
-----------	------	--

Once the list is build we call the method \$gsetlist(theList,kGanttListPhase) to send it to the layer object.

Example Code: `oGanttObject.build_chart`

Relation List



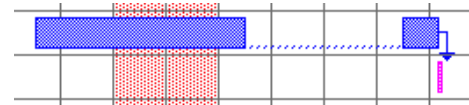
The relation list specifies how the phases are related. Phases can be related in a variety of ways. A relationship can be specified by using the constants kGanttRel.... A phase may be related to more than one other phase, but there can be only one relationship between the same two phases.

List definition of the relation list. One row per relationship between two phases.		
Column	Type	Description
RelationID	Long Integer	ID of the relation-ship
StartPhaseID	Long Integer	ID of the first phase
EndPhaseID	Long Integer	ID of the second phase
RelationType	Constant	<p>Type of relation-ship:</p> <p>kGanttRelFinishFinish - both phases finish at the same time.</p> <p>kGanttRelFinishStart - first phase has to finish before the second phase can start.</p> <p>kGanttRelStartFinish - second phase must finish before first phase can start.</p> <p>kGanttRelStartStart - both phases must start at the same time.</p> <p>The constants kGanttRelFinishFinishBlock, kGanttRelFinishStartBlock, kGanttRelStartFinishBlock and kGanttRelStartStartBlock are the same but without any delay between the phases. The column RelationDelay will be ignored and phases will be moved so they start/finish together.</p>
RelationDelay	Long Integer	Delay of the relationship expressed in seconds.

Once the list is build we call the method \$gsetlist(theList,kGanttListRelation) to send it to the layer object.

Example Code: `oGanttObject.build_chart`

Suspension List



The suspension list specifies periods during which a phase is suspended. This period is represented in the chart by a dotted line between the active periods of a phase. A phase may have more than one suspension period.

List definition of the suspension list. One row per suspension period.		
Column	Type	Description
ResourceID	Long Integer	Resource ID of the Phase
PhaseID	Long Integer	Phase ID
SuspensionID	Long Integer	ID of this suspension period
StartDate	Date Time	Start date and time of the suspension period
EndDate	Date Time	End date and time of the suspension period

Once the list is build we call the method `$gsetlist(theList,kGanttListSuspension)` to send it to the layer object.

Example Code: `oGanttObject.build_chart`

Legend List

The legend list specifies the text and color for the Graph/Totals pane legend. The legend is displayed in the bottom left pane when showing the Graph or Totals pane.

List definition of the legend list. One row per legend.		
Column	Type	Description
Text	Character	The legend text
TextColor	Long Integer	The text color

Once the list is build we call the method `$lsetstringlist(theList)` to send it to the GanttUI window object.

Marker List and Display Date

The marker list can be used to visually display important dates in a project. An entry in the marker list represents a vertical line in a specified thickness and color that is drawn in the right-hand chart at the specified date and time. You could use this list to show the start and end dates of the project and maybe important mile stones.

List definition of the marker list. One row per marker.		
Column	Type	Description
Date	Date Time	Date and time where to draw the marker
Color	Long Integer	Color of the line
Width	Long Integer	Width of the marker in pixels

Once the list is build we call the method `$setmarkerlist(theList)` to send it to the GanttUI window object.

While we are setting important dates for the project we will also need to set the initial display date. We do this by calling the method `$gsetstartdate`.

Example Code: `oGanttObject.build_chart`

5. Assign Layer Object And Build

Finally we can assign the layer object for display in the GanttUI window object and build the chart.

We assign and show a layer object using the methods `$addlayer`, `$tluselayer` and `$guselayer`. These are methods of the GanttUI window object.

We also must build the tree and chart and we do this by calling the methods `$tlbuild` and `$gbuild` of the layer object.

Example Code: `oGanttObject.load_chart`

Saving and Loading GANTT charts

You will need to decide how you will store a chart in your database. Essentially there are two choices.

1. Single Record. You store all the GANTT data as a single record.
2. Multi Record. Create matching tables for all the various lists in a GANTT chart.

We will briefly discuss these options with appropriate references to sample code.

1. Single Record

Storing all the GANTT data as a single record is easy. You create a single table with a column for each of the OGantt lists plus any additional columns that you require. Your database may contain multiple GANTT charts, that you may wish to load individually or load two or more charts at the same time using multiple layer objects. The single-record approach is easier and quicker to implement and faster to load when viewing an entire chart. But the downside is that you need to load the entire GANTT data if you just want to look at some phases in your chart, and you need to save the entire GANTT data during a save.

Example Code: `oGanttDataObject.$save_chart`
`oGanttDataObject.$load_chart`
`oGanttDataObject.$new_chart`

2. Multi Record

For this strategy you will need to create tables for all the various lists in a GANTT chart and store each row as a single record in your database. Using this approach also allows you to maintain multiple charts. In addition to the tables for the GANTT data you can create a header table that has a unique ID, and all your GANTT records can maintain a relational link to the header record.

Example Code: `oGanttDataObject.$save_chart`
`oGanttDataObject.$load_chart`
`oGanttDataObject.$new_chart`

Tracking Changes

When saving changes that a user has made, it is not desirable to write all data to the database. Ideally you only want to write those records that are effected by the changes. OGantt can help with this.

As the user makes changes to the chart, OGantt remembers these changes and the various lists that record these changes can be accessed using the methods `$ggetmodlistresource`, `$ggetmodlistphase`, `$ggetmodlistsusp`, and `$ggetmodlistrel`.

You have two choices here. You can either save all changes when the user clicks the save button, or you can save changes as the user makes the changes. While the user interacts you will receive a number of events, such as `evGMove`, `evGResize`, `evGCreate`, `evGAddSusp`, etc. During any of these events you can retrieve the modification lists and write the records to the database.

After you have updated your database use the \$gfreemodlist... methods to clear these lists, ready for the next set of changes.

However, some changes that are made by Omnis notation are not recorded via the OGantt modification lists. These you must track your self. The examples implement a messaging system that allows the oGanttDataObject to receive all messages related to changes.

Example Code: `oGanttDataObject.$begin_update`
`oGanttDataObject.$end_update`

Interacting With OGantt

As the user interacts with OGantt, the component will generate a number of events that you can intercept in your fields \$event method. You may also provide a number of menus, toolbars and windows to manipulate other GANTT attributes. It typically involves setting properties or calling methods of the OGantt layer object or window control, or both. The provided examples are designed around a messaging system that allows both the layer object and window control to receive messages from other interface objects and deal with requests at the appropriate levels. There are comments throughout the code in the examples to help you understand the interface. Please also read the chapter Examples Reference. Essentially, there are five classes that bring together the OGantt interface. The window class wGanttUI for editing a chart, the object classes oGanttLayerObject and oGanttDataObject for loading and saving charts, and the object classes oGanttMessage and oGanttUI for handling the message system and multi-lingual translation.

Editing tree list cells

It is possible for users to directly edit the cells in the tree list portion of the OGantt control. You can specify if a cell can be edited by assigning kTrue in the 'Enterable' column of a row when building the tree list data (see "3. Loading the Left-Hand Tree"). The cell can be edited using all the usual arrow keys and edit menu options. To commit a change the user can use the tab key or click another cell. The only action that cancels the text entry is the escape key. The three special events evTLTextEditStart, evTLTextChanged and evTLTextEditEnd allow the developer to control and test user input.

Web-Client Support

Those familiar with the OGantt component will know that two OGantt objects are required to load and display a chart. The non-visual layer object and the visual window component. As you may be aware, the web-client architecture does not support non-visual objects, but we still require a layer object on the client for use by the remote form. To work around this limitation we have implemented the layer object as a invisible remote form object that must be placed on a remote form like any other form control. Once placed on a remote form it can be used very much like the non-visual layer object in the fat-client.

The Studio 5 only OGantt example library includes a sample remote form and task that loads a Gantt chart. The remote form does not fully implement an interface in the way the fat-client version does, but it should be sufficient to serve as a good starting point for further development. We will continue to work on these examples time permitting.

Important note regarding calling external methods on the web-client. When calling methods with reference parameters (parameters that return values), these parameters must be passed by name and they must reference instance variables.

Example:

```
Do $cinst.$objs.Gantt.$getmarkerlist (nam(ivMarkerList),nam(ivMarkersBehind))
```

Translation

The OGantt examples have been designed from the ground up to take into account multi-lingual implementations. The examples implement English and German text. To add additional languages simply edit the OGantt string table in the example's resource folder.

We slightly deviated from the convention when using string tables in an Omnis interface. Traditionally, one would enter calculations, typically inside square brackets, for the various interface items, such as menu lines, window controls, toolbar objects, etc. However, this requires the developer to create string tables during the development stages to do any useful on-the-fly testing of an interface. It is impossible to get a feel for the design of a window or menu without seeing the real text.

This approach we found very cumbersome, in fact so cumbersome that it negatively impacted the developers creativity. It also doesn't address the problems of multi-lingual web-client interfaces.

So we designed a system that allows the developer to just go and design an interface class in the traditional way, before string tables, and let someone else worry about string tables, or create the strings for a class once the design has been completed. This system relies on the same messaging system used throughout. Any class that requires translation or messages from other interface objects, simply drops a message object into the instance variables, and it is done.

Please see the object classes oGanttMessage and oGanttUI for more details.

Printing

OGantt provides a external report component that can be added to an Omnis report class. A GANTT chart can be printed by loading the data using a layer object just as you would when loading it for the window object. There are some small differences and additional properties, related to printing. These are fully documented throughout the code.

Print Status

In order to print a chart, you must respond to a printing status returned into an instance variable in report class. You specify the name of the variable in the \$resultvar property of the report object. OGantt will return one of the kGanttPrintResult... status codes. The reason is simple. When you print a GANTT chart in your record section it may not all fit on the page. So while there is still more to print, you must continue printing the record section until OGantt returns a value other than kGanttPrintResultContinue in your variable.

This simple loop will print an entire chart, across several pages if necessary.

```
Repeat
  Do $cinst.$printrecord()
Until ivPrintResult<>kGanttPrintResultContinue
Do $cinst.$sendprint()
```

You must not load new chart data until the current chart data has finished printing.

Positioning

In principal, OGantt prints using the entire paper using the four OGantt margin properties for the paper margin.

The property `$sprintposmode` (introduced in version 3.7.2) provides some additional control over how the report objects co-ordinates are used together with the `$marginup`, `$margindown`, `$marginleft` and `$marginright` properties.

Further Reading

We recommend that you also read the following technical note(s) available from our public support website prior to working with OGantt.

TN0022 ‘External Component Version Numbers’ explains how to programmatically check the version numbers of our software.

Examples Reference

This reference serves as a guide to the classes of the OGantt examples. Only a brief description is provided here as there are many comments throughout the code.

Contents

Window Classes

Object Classes

File Classes

Toolbar Classes

Report Classes

Window Classes

The examples provide a number of windows that make up the interface to OGantt. Most of these windows behave like dialogs that are opened for a short time while they are required for input to an action.

wGanttDlgEditPhase

This class is used as a sub-window in the main window wGanttUI. It provides a point and click interface to setting phase attributes.

Super-class: wGanttDlgSuper

wGanttDlgGetDate

A simple dialog with a calendar and time field for returning a date and time.

Super-class: wGanttDlgSuper

wGanttDlgLoadChart

A simple dialog for choosing a chart to load. It lists all charts in the database .

Super-class: wGanttDlgSuper

wGanttDlgNewChart

A simple dialog for creating a new chart.

Super-class: wGanttDlgSuper

wGanttDlgPrint

Provides an interface for selecting printing options.

Super-class: wGanttDlgSuper

wGanttDlgSetRelation

A simple dialog for selecting relation options when creating a relationship between two phases.

Super-class: wGanttDlgSuper

wGanttDlgSuper

The super-class for all wGanttDlg... windows. It implements code to handle construction as a sub-window or stand-alone window. It also manages the oGanttUI instance for messaging and translation.

Super-class: none

wGanttUI

This is the main interface for editing a chart. It manages the chart layer object oGanttLayerObject, the message object oGanttUI and directly uses tbGanttUI and wGanttDlgEditPhase. Most other windows are utilised by this class.



Object Classes

There are a small number of objects that manage the loading and saving of charts and the messaging between the various interface classes.

oGanttDataObject

Implements all database access for loading and saving charts.

Super-class: none

oGanttLayerObject

This class is derived from the external NV object `.Gantt.GanttObject` and provides additional methods and properties to that of the external object. It also communicates with `oGanttDataObject` for loading and saving to the database. It also implements an instance of `oGanttUI` for receiving and sending interface messages.

Super-class: .Gantt.GanttObject

oGanttMessage

Implements the hub of the OGantt interface message handling. Any class that adds this object to its instance variables during runtime, will receive messages that are sent via this object. It also provides functionality for creating a parameter row variable for variable number of parameters as required by different messages.

Super-class: none

oGanttUI

Mainly implements the handling of the OGantt string table for multi-lingual support. It derives from the message object `oGanttMessage` and any class that includes this object in its instance variables will gain message handling and translation of the class.

Super-class: oGanttMessage

File Classes

The file classes store the GANTT data in an Omnis data file. The entire database access is managed by the object class oGanttDataObject. The examples are capable of storing a GANTT chart as a single record or using multiple records for GANTT details with relational links to the main record. As all DB access is centralised inside an object class, it should be fairly easy to adopt the DB access code to SQL and your own data structures.

fGantt

The main GANTT chart record. When storing a chart as a single record, all GANTT data is stored as a record in this file.

fGanttCalHolidays

Stores the holiday records. All GANTT charts share this data, although it is possible to assign specific holidays to individual resources.

fGanttCalWorkingHours

Stores the working hours. All GANTT charts share this data, although it is possible to assign specific working hours to individual resources.

fGanttMarkers

Stores a chart's special dates when storing a chart using multiple records. The relational link GMAR_GANTT_ID links to the main record stored in fGantt.

fGanttPhases

Stores the phase details when storing a chart using multiple records. The relational link GPHA_GANTT_ID links to the main record stored in fGantt.

fGanttRelations

Stores the relation details when storing a chart using multiple records. The relational link GREL_GANTT_ID links to the main record stored in fGantt.

fGanttResources

Stores the resource details when storing a chart using multiple records. The relational link GRES_GANTT_ID links to the main record stored in fGantt.

fGanttSuspensions

Stores the suspension period details when storing a chart using multiple records. The relational link GSUS_GANTT_ID links to the main record stored in fGantt.



Toolbar Classes

tbGanttUI

The main toolbar for the window wGanttUI.

Super-class: none



Report Classes

rGanttUI


A simple report class that demonstrates how to print a GANTT chart.


Super-class: none

External Component Reference

Introduction

This chapter lists all the OGantt constants, properties, methods and events. The OGantt component library supplies a number of external objects

 Window Object “GanttUI” for editing and viewing GANTT charts in an Omnis window.

 Report object “GanttPrint” for printing GANTT charts in an Omnis report.

 Non-visual object “GanttObject” for building and editing GANTT charts.

Contents

Constants - OGantt provides a number of constants that are used with OGantt properties or methods. The constants are organised into functional groups and can be accessed from the Omnis Catalogue.

GanttUI/GanttPrint Properties - Properties of the GanttUI window control.

Note: Properties shown in red are read-only and cannot be assigned.

GanttUI Methods - Methods of the GanttUI window control.

GanttUI Events - Events of the GanttUI window control.

GanttObject Properties - Properties of the OGantt NV layer object.

Note: Properties shown in red are read-only and cannot be assigned.

GanttObject Methods - Methods of the GanttUI NV layer object.

Constants

kGantt...FirstLoad

Constants for specifying if this build is a first load or additional load. Used with \$tlbuild() and \$gbuild().

Name	Value	Description
kGanttIsFirstLoad	1	This is the first load.
kGanttNoFirstLoad	2	It is not the first load.

kGantt(days)

Constants for use in the holidays list.

Name	Value	Description
kGanttSunday	0	Sunday
kGanttMonday	1	Monday
kGanttTuesday	2	Tuesday
kGanttWednesday	3	Wednesday
kGanttThursday	4	Thursday
kGanttFriday	5	Friday
kGanttSaturday	6	Saturday

kGanttBar...

Constants for specifying the bar style. Assigned to \$gbarstyle.

Name	Value	Description
kGanttBarNormal	0	Traditional patterned style.
kGanttBarWash4	1	3D wash style with little light.
kGanttBarWash8	2	3D wash style with medium light.
kGanttBarWash12	3	3D wash style with strong light.
kGanttBarWash16	4	3D wash style with very strong light.

kGanttCalc...

Constants for specifying the calculation type for totals in the totals section. Used with \$grsetfuncforfield(), \$ggettotals() and \$grsetfuncforfield().

Name	Value	Description
------	-------	-------------

kGanttCalcNone	0	No calculation.
kGanttCalcMax	1	Show the maximum.
kGanttCalcTot	2	Show the total.
kGanttCalcCount	3	Show the number of elements.
kGanttCalcDifferent	4	Show the number of different elements.
kGanttCalcEqual	5	Show the number of equal elements with one specified.

kGanttCanDrop...

Constants for specifying where drops can occur. Used with \$lddcandrop and \$gddcandrop. (New for version 4.0)

Name	Value	Description
kGanttCanDropNowhere	0	No dropping is allowed.
kGanttCanDropBetweenRows	1	Can drop between rows.
kGanttCanDropOnRow	2	Can drop on top of rows.
kGanttCanDropOnPhase	4	Can drop on top of phases (\$gddcandrop only)

kGanttDrop...

Constants specifying where a drop occurred. Used with \$ddwhere which may be set to a combination of the constants below. (New for version 4.0)

Name	Value	Description
kGanttDropNowhere	0	No dropping occurred.
kGanttDropBefore	1	Drop occurred before the row specified by \$ddresourceid.
kGanttDropAfter	2	Drop occurred after the row specified by \$ddresourceid.
kGanttDropTreeList	4	Drop occurred over the tree list.
kGanttDropGantt	8	Drop occurred over the GANTT chart.
kGanttDropPhase	16	Drop occurred over a phase specified by \$ddphaseid.

kGanttGraphType...

Constants for specifying the type of graph to display. Used with \$grsetgraphtype().

Name	Value	Description
kGanttGraphTypeNone	1	No graph.

kGanttGraphTypeCurve	2	Standard line graph.
kGanttGraphTypeResource	3	Advanced resources graph.
kGanttHol...		
Constants for use in the holidays list.		
Name	Value	Description
kGanttHolFixedDay	1	A holiday that has a fixed date.
kGanttHolWeekDayRep	2	A holiday that is repeated weekly, such as Sunday.
kGanttHolYearDayRep	3	A holiday that is repeated yearly.
kGanttList...		
Constants for use with the \$tlsetlist() and \$gsetlist() methods.		
Name	Value	Description
kGanttListTree	1	The list contains the data of the tree, use with \$tlsetlist()
kGanttListColumn	2	The list contains the data of one column of the tree, used with \$tlsetlist()
kGanttListResource	1	The list contains the data of the resources, used with \$gsetlist().
kGanttListPhase	2	The list contains the data of the phases, used with \$gsetlist().
kGanttListSuspension	3	The list contains the data of the suspension periods, used with \$gsetlist().
kGanttListRelation	4	The list contains the data of the relation-ships between phases, used with \$gsetlist().
kGanttModReason...		
Constants for testing the reason of a modification. Used with lists returned by \$ggetmodlistphase(), \$ggetmodlistresource(), \$ggetmodlistrel(), \$ggetmodlistsusp().		
Name	Value	Description
kGanttModReasonAdd	1	A new phase, resource, relation or suspension was added.
kGanttModReasonMod	2	A phase, resource, relation or suspension was modified.
kGanttModReasonDel	3	A phase, resource, relation or suspension was deleted.

kGanttPrintPos...

Constants for positioning the chart during printing. Use with the property \$printposmode. The constants specify which of the report objects coordinates are to be ignored in favour of the margin properties \$marginup, \$margindown, \$marginleft and \$marginright. When using any of the constants other than kGanttPrintPosIgnoreNone, each record that contains a chart or portion of a chart must be printed on a new page.

Name	Value	Description
kGanttPrintPosLegacy	0	Legacy positioning (pre v3.7.2)
kGanttPrintPosIgnoreNone	1	All the report object's current coordinates are used and as much of the chart is fitted within the bounds of the object. This is the only mode that allows more than one chart or portion of a chart on the same page.
kGanttPrintPosIgnoreAll	2	All the report object's coordinates are ignored and the chart is placed on the paper according to the four margin properties.
kGanttPrintPosIgnoreLeftRightBottom	3	The report object's left, right and bottom coordinates are ignored in favour of the left, right and bottom page margins. The top of the chart is placed at the object's current top coordinate, allowing for other content above the chart.
kGanttPrintPosIgnoreLeftRight	4	The report object's left and right coordinates are ignored in favour of the left and right page margins. The top and bottom of the chart is placed at the object's current top and bottom coordinate, allowing for other content above and below the chart.
kGanttPrintPosIgnoreBottom	5	The report object's bottom coordinate is ignored in favour of the bottom page margin. The bottom of the chart is placed at the object's current bottom coordinate, allowing for other content below the chart.

kGanttPrintResult...





Status codes returned into an report instance variable when printing a chart. The instance variable name must be specified via the property \$resultvar.


Name	Value	Description
kGanttPrintResultFinish	0	The chart has finished printing.
kGanttPrintResultContinue	1	Not all of the chart has been printed.
kGanttPrintResultError	-1	An error occurred during printing.




kGanttRel...		
Constants for setting the relationship between two phases. Used with the relations list.		
Name	Value	Description
kGanttRelFinishStart	1	Finish to start relationship.
kGanttRelStartFinish	2	Start to finish relationship.
kGanttRelStartStart	3	Start to start relationship.
kGanttRelFinishFinish	4	Finish to finish relationship.
kGanttRelFinishStartBlock	5	Finish to start relationship without any delay.
kGanttRelStartFinishBlock	6	Start to finish relationship without any delay.
kGanttRelStartStartBlock	7	Finish to finish relationship without any delay.
kGanttRelFinishFinishBlock	8	Finish to finish relationship without any delay.
kGanttRelAdd...		
Result codes when adding relationships. Used with \$gaddrelation().		
Name	Value	Description
kGanttRelAddOk	1	Relationship was added.
kGanttRelAddErrParam	2	Parameter error.
kGanttRelAddErrRelated	3	Phases are already related.
kGanttRelAddErrCycle	4	Adding this relationship would create a cycle/loop.
kGanttScale...		
Constants for setting the display scale. Used with \$gsetscale().		
Name	Value	Description
kGanttScaleMinute	1	Minute scale.
kGanttScale15Minute (v4.0)	2	Fifteen minutes scale
kGanttScaleHour	3	Hour Scale.
kGanttScale2Hour (v4.0)	4	Two hourly scale
kGanttScale4Hour	5	Four hourly scale.
kGanttScale6Hour (v4.0)	6	Six hourly scale.
kGanttScaleDay	7	Day scale.










kGanttScaleWeek	8	Week scale.
kGanttScaleMonth	9	Month scale.
kGanttScaleYear	10	Year scale.
kGanttTot...		
Constants for adding total fields to the total section. Use with the list assigned to \$totcalclist.		
Name	Value	Description
kGanttTotNone	0	No totals.
kGanttTotTotal	1	Show totals.
kGanttTotProgress	2	Show progressive totals.
kGanttTotFixedText	3	Show fixed text.
kGanttTypeDur...		
Constants for use in the phase list.		
Name	Value	Description
kGanttTypeDurFix	1	The duration is a fixed duration (including suspension periods).
kGanttTypeDurEff	2	The duration is a effective duration(excluding suspension periods).

GanttUI/GanttPrint Properties

Name	Type	Description
\$adaptdimension 	Boolean	If true adopt the dimensions of the gantt to the page.
\$scalborder (v3.7.0)	Integer	Border style of the chart header. One of the standard Omnis kBorder... constants.
\$scalbordercolor (v3.7.0)	Integer	Colour used for painting the border of the chart header. Specify a median colour that is used to produce light and dark shades for the various border styles. It mostly works best when both the fill and border colours are set the same, but the border colours can be set independently to give a different effect.
\$scalfillcolor (v3.7.0)	Integer	Background fill colour for the chart header.
\$scalfontname	Integer	Font for the chart header.
\$scalfontsize	Integer	Font size for the chart header.
\$scalfontstyle	Integer	Font style for the chart header.
\$scalheight	Integer	Height in pixels of the chart header .
\$colstoprint 	Integer	Specifies how many tree list columns to print.
\$ddphaseid (v4.0.0)	Integer	Phase ID of the phase where the drop occurred when \$ddwhere is kGanttDropPhase. May be zero. (read-only) See also kGanttCanDrop..., \$gddcandrop, \$tlddcandrop and \$ddresourceid.
\$ddresourceid (v4.0.0)	Integer	Resource ID of the line where the drop occurred in relation to \$ddwhere (read-only) See also kGanttCanDrop..., \$gddcandrop, \$tlddcandrop and \$ddphaseid.
\$ddwhere (v4.0.0)	Integer	Flags specifying where the drop occurred, one of the kGanttDrop... constants (read-only). See also kGanttCanDrop..., \$gddcandrop, \$tlddcandrop, \$ddresourceid and \$ddphaseid.
\$diagmaxdim 	Number	Maximum dimensions of the diagram.
\$diagmindim 	Number	Minimum dimensions of the diagram.

\$enddate 	Date	Specifies end date of the print job. Empty or NULL prints all.
\$gbarstyle	Integer	Style of the bars in the gantt chart.
\$gdaymonthname	List	Sets the day and month names for the chart header.
\$gddcandrop (v4.0.0)	Integer	Flags specifying where we can drop in the GANTT chart, one of the kGanttCanDrop... constants. If set to kGanttCanDropNowhere, dropping on the GANTT chart is disabled. See also \$lddcandrop, \$ddwhere, \$ddresourceid and \$ddphaseid.
\$gfontname	Integer	Chart font name
\$gfontsize	Integer	Chart font size
\$gfontstyle	Integer	Chart font style
\$gphasecandrop	Boolean	If set to kTrue, phases can be dragged and dropped between different resources. See also evPhaseCanDrop and evGMove.
\$graphtype	Integer	Type of diagram to be displayed.
\$grcurvedrawbox	Integer	If kTrue the line graph shows square points.
\$grcurvesum	Boolean	If kTrue the lines in the line graph show the sum.
\$gridhlinecolor (v3.8.0)	RGB colour	Specifies the line colour for the horizontal grid lines.
\$gridhlinestyle (v3.8.0)	Line Style	Specifies the line style for the horizontal grid lines. (hair line not supported)
\$gridvlinecolor (v3.8.0)	RGB colour	Specifies the line colour for the vertical grid lines.
\$gridvlinestyle (v3.8.0)	Line Style	Specifies the line style for the vertical grid lines. (hair line not supported)
\$grellinestyle (v3.8.0)	Line Style	Specifies the line style for the relationship lines. (hair line not supported)
\$growheight	Integer	Height of the chart row in pixels.
\$grscalecolor (v3.8.0)	RGB Colour	Specifies the colour for the graph scale
\$grscalefontname (v3.8.0)	Font name	Specifies the font for the graph scale

\$grscalefontsize (v3.8.0)	Integer	Specifies the font size for the graph scale
\$grscalefontstyle (v3.8.0)	Integer	Specifies the font style for the graph scale
\$gshowhlines	Boolean	Shows or hides the horizontal lines of the Gantt.
\$gshowvlines	Boolean	Shows or hides the vertical lines of the chart.
\$headborder (v3.7.0)	Integer	Border style of the tree list header. One of the standard Omnis kBorder... constants.
\$headbordercolor (v3.7.0)	Integer	Colour used for painting the border of the tree list header. Specify a median colour that is used to produce light and dark shades for the various border styles. It mostly works best when both the fill and border colours are set the same, but the border colours can be set independently to give a different effect.
\$headfillcolor (v3.7.0)	Integer	Background fill colour for the tree list header.
\$headfontname	Integer	Font of the Tree list header.
\$headfontsize	Integer	Font size of the header in the Tree list.
\$headfontstyle	Integer	Font style of the header in the Tree list.
\$headheight	Integer	Height in pixels of the header in the Tree list.
\$hourmovement <input type="checkbox"/>	Boolean	If kTrue, sets the movement of phases at hour intervals.
\$hscroll (v4.0.0)	Integer	As there is no real horizontal scroll range, this property always returns zero. One can offset the current horizontal position by a negative or positive value. The value represents increments of the minor horizontal scale. So when the minor scale is days, assigning 10 to \$hscroll will offset the horizontal position by +10 days. See also \$vscroll.
\$ignorerelations (v4.0.0)	Boolean	If true, while dragging a phase, the setting of relationships between phases is disabled.
\$layergantt 	Integer	Specifies which layer must be displayed in the chart.
\$layerstringlist 	Integer	Layer string list to print, separated by commas.
\$layertree 	Integer	Specifies which layer must be displayed in the tree list.

\$legendstringlist 	List	List of the strings to view in the legend.
\$majorscale	Integer	Major scale of the chart.
\$majorscaleshowsyear	Boolean	If kTrue, show the year in the chart header.
\$margindown 	Number	Margin below the chart when printing. See \$printposmode for more details.
\$marginleft 	Number	Margin to the left of the chart when printing. See \$printposmode for more details.
\$marginright 	Number	Margin to the right of the chart when printing. See \$printposmode for more details.
\$marginup 	Number	Margin above the chart when printing. See \$printposmode for more details.
\$minorscale	Integer	Minor scale of the chart.
\$numdecimals	Integer	Number of decimals to display.
\$printposmode (v3.7.2)	Integer	Specifies how the chart is positioned on the page in conjunction with the objects coordinates and the margin properties. It takes one of the kGanttPrintPos... constants.
\$printsequence 	Boolean	If kTrue, prints a progressive number in the Gantt calendar.
\$printtreeallpages 	Boolean	If kTrue, prints the tree on all pages.
\$resultvar 	Integer	This property takes a report instance var for returning a printing status code, one of the kGanttPrintResult... constants.
\$showdiagram	Boolean	Shows or hides the diagram/graph.
\$showselline	Boolean	If ktrue, shows the selected row in the Tree list.
\$showtotsection	Boolean	Shows or hides the total section.
\$splitterhpos	Integer	Vertical position of the splitter in pixels.
\$splittervpos	Integer	Horizontal position of the splitter in pixels.
\$startdate 	Date	Specifies start date of the print job. Empty or NULL prints all.

\$swapclientarea (v3.7.0)	Boolean	In version 3.7.0 the new behaviour is that the chart area above the splitter bar resizes when the total or graph section is visible and the window is resized. If this is not the desired behaviour this property has been provided to revert to previous behaviour.
\$tlddcandrop (v4.0.0)	Integer	Flags specifying where we can drop in the tree list, one of the kGanttCanDrop... constants. If set to kGanttCanDropNowhere, dropping on the tree list is disabled. See also \$gddcandrop, \$ddwhere, \$ddresourceid and \$ddphaseid.
\$tlexpandicon	Integer	Icon for the expansion of the Tree List.
\$tlfontname	Integer	Font of the Tree list.
\$tlfontsize	Integer	Font size of the Tree list.
\$tlfontstyle	Integer	Font style in the Tree list.
\$tlrealrowheight	Integer	Real row height in the Tree list.
\$tlrowheight	Integer	Height of a row in the Tree list. If = 0 it will be automatically adjusted to the font size.
\$tlshowhlines	Boolean	Shows or hides the horizontal lines of the Tree list.
\$tlshowvlines	Boolean	Shows or hides the vertical lines of the Tree list.
\$stotalclist	List	List of data to be calculated in the total section.
\$vscroll (v4.0.0)	Integer	Returns the first visible line starting from 1. Assigning \$vscroll will change the first visible line by scrolling the chart appropriately. See also \$hscroll.

GanttUI Methods

\$addlayer()

Syntax: *OGanttUIRef*.\$addlayer(iLayerId)

Adds a layer to the control.

Parameter	Description
iLayerId	The id of the layer object returned by <i>OGanttObject</i> .\$layerid.

\$gbeginsplit()

Syntax: *OGanttUIRef*.\$gbeginsplit()

Puts control into split phase mode.

Parameter	Description
none	

\$genablelayer()

Syntax: *OGanttUIRef*.\$genablelayer(iLayerId,bEnable)

Enables or disables the specified layer.

Parameter	Description
iLayerId	The id of the layer object returned by <i>OGanttObject</i> .\$layerid.
bEnable	kTrue or kFalse

\$gendsplit()

Syntax: *OGanttUIRef*.\$gendsplit()

Cancels the split phase mode.

Parameter	Description
none	

\$getbitmap()

Syntax: *OGanttUIRef*.\$getbitmap(dStartDate,dEndDate)

Creates and returns a bitmap of the GANTT chart for the specified date range.

Parameter	Description
dStartDate	Start date of date range for which to return the bitmap. (New for v4.0.0)
dEndDate	End date of date range for which to return the bitmap. (New for v4.0.0)

returns	Bitmap as a picture
\$getmarkerlist()	
Syntax: <i>OGanttUIRef</i> .\$getmarkerlist(&lMarkerList,&bMarkersBehind)	
Gets the list of markers and their drawing position for the chart. Markers are vertical lines displayed in the chart area.	
See also \$setmarkerlist.	
Parameter	Description
lMarkerList	List will be returned with three columns (Date,Colour,Width).
bMarkersBehind	Returns true if markers are painted behind phases and text.
\$ggetselectphase()	
Syntax: <i>OGanttUIRef</i> .\$ggetselectphase()	
Returns the ID of the selected phase	
Parameter	Description
returns	ID of the selected phase
\$ggetstartdate()	
Syntax: <i>OGanttUIRef</i> .\$ggetstartdate()	
Returns the first visible date (left most date column). (new for version 4.0.0)	
Parameter	Description
returns	Date / Time
\$gredraw()	
Syntax: <i>OGanttUIRef</i> .\$gredraw()	
Redraws the chart area of the control.	
Parameter	Description
none	
\$grredraw()	
Syntax: <i>OGanttUIRef</i> .\$grredraw()	
Redraws the graph/diagram area of the control.	
Parameter	Description
none	

\$grsetgraphtype()

Syntax: *OGanttUIRef*.\$grsetgraphtype(iGraphType)

Sets the type of the graph to be displayed

Parameter	Description
iGraphType	Specifies the graph type. One of the kGanttGraphType... constants.

\$gsetdaymonthname()

Syntax: *OGanttUIRef*.\$gsetdaymonthname(IDayMonthList)

Sets the day and month names for the chart header.

Parameter	Description
IDayMonthList	Single column list of day and month names.

\$gsetscale()

Syntax: *OGanttUIRef*.\$gsetscale(iMajorScale,iMinorScale)

Sets the major and minor scales of the chart area.

Note: When specifying date formatting strings for iMajorScaleFormat or iMinorScaleFormat, OGantt uses the standard Omnis jst() function to evaluate the format string for each date column. Unfortunately, the jst() function cannot return ISO week numbers. We have therefore added support for the tilde character '~'. If you add the tilde character to the date format strings when calling \$gsetscale, OGantt will insert the ISO week number in its place. (new for version 4.0.0)

Parameter	Description
iMajorScale	The major scale. One of the kGanttScale... constants.
iMinorScale	The minor scale. One of the kGanttScale... constants.
cMajorScaleFormat (v3.7.0)	Date formatting string for the major scale. The formatting string must have the same syntax as date formatting strings specified in the jst() function excluding the preceding 'D:' or 'T:'.
cMinorScaleFormat (v3.7.0)	Date formatting string for the minor scale. The formatting string must have the same syntax as date formatting strings specified in the jst() function excluding the preceding 'D:' or 'T:'.
iMinorScaleWidth (v3.7.0)	Specifies the width of the minor scale in pixels.
iSnapScale (v3.8.0)	Specifies the scale in minutes for snapping when moving or sizing phases.

iWeekStart	<p>Specifies the week start day (one of the Omnis day constants). The week start is used when displaying weekly columns in the chart header.</p> <p>Example: <code>\$gsetscale(kGanttScaleWeek, kGanttScaleDay, "D m 'Y (~)", "v", 30, #NULL, kMonday)</code></p>
<p>\$gsetselectphase() Syntax: <i>OGanttUIRef</i>.\$gsetselectphase(iPhaseId) Selects the specified phase.</p>	
Parameter	Description
iPhaseId	The ID of the phase to be selected. Pass zero to deselect the phase.
<p>\$gsetstartdate() Syntax: <i>OGanttUIRef</i>.\$gsetstartdate(dNewStartDate) Sets the start display date for the chart.</p>	
Parameter	Description
dNewStartDate	The date at which to position the chart.
<p>\$gsettooltiptext() Syntax: <i>OGanttUIRef</i>.\$gsettooltiptext(ITooltipText) Sets the tool tip text that is used when sizing or moving phases.</p>	
Parameter	Description
ITooltipText	<p>List of text for the various components of the tool tip.</p> <p>Line 1 = "Completion" Line 2 = "Start" Line 3 = "Finish" Line 4 = "Duration" Line 5 = date format i.e. "D M y H:N" (optional)</p>
<p>\$guselayer() Syntax: <i>OGanttUIRef</i>.\$guselayer(iLayerId) Makes the specified layer the current layer to be used in the chart area.</p>	
Parameter	Description
iLayerId	The id of the layer object returned by <i>OGanttObject</i> .\$layerid.

www.brainydata.com

\$lsetstringlist()	
Syntax: <i>OGanttUIRef</i> .\$lsetstringlist(IStringList)	
Specifies the list of text for the graph and totals legend.	
Parameter	Description
IStringList	List with two columns (Text,Colour).
\$removelaye()	
Syntax: <i>OGanttUIRef</i> .\$removelaye(iLayerId)	
Removes the specified layer from the control.	
Parameter	Description
iLayerId	The id of the layer object returned by <i>OGanttObject</i> .\$layerid.
\$setmarkerlist()	
Syntax: <i>OGanttUIRef</i> .\$setmarkerlist(IMarkerList,bMarkersBehind)	
Sets the list of markers for the chart. Markers are vertical lines displayed in the chart area. See also \$getmarkerlist.	
Parameter	Description
IMarkerList	List with three columns (Date,Colour,Width).
bMarkersBehind	Specifies if markers are to be drawn behind phases and their text.
\$tlgetselectresource()	
Syntax: <i>OGanttUIRef</i> .\$tlgetselectresource()	
Returns the ID of the selected resource in the tree.	
Parameter	Description
returns	ID of the resource.
\$tlredraw()	
Syntax: <i>OGanttUIRef</i> .\$tlredraw()	
Redraws the tree list of the control.	
Parameter	Description
none	

\$tlselectresource()	
Syntax: <i>OGanttUIRef.\$tlselectresource(iResourceId)</i>	
Selects the specified resource in the tree.	
Parameter	Description
iResourceId	ID of the resource to select. Pass zero to deselect the resource.
\$tluselayer()	
Syntax: <i>OGanttUIRef.\$tluselayer(iLayerId)</i>	
Makes the specified layer the current layer to be used in the tree area.	
Parameter	Description
iLayerId	The id of the layer object returned by <i>OGanttObject.\$layerid</i> .
\$totrecalc()	
Syntax: <i>OGanttUIRef.\$totrecalc()</i>	
Forces the control to recalculate all totals.	
Parameter	Description
none	
\$totredraw()	
Syntax: <i>OGanttUIRef.\$totredraw()</i>	
Redraws the totals section.	
Parameter	Description
none	

GanttUI Events

evClick

This event is generated when a click occurs in the chart

no event parameters

evCollapse

This event is generated when the user collapses a category in the tree list

Event Parameter	Description
pResourceId	The ID of the resource that was collapsed.

evDragDrop

This event is generated when the drags and drops resources in the tree list.

Event Parameter	Description
pResourceId	The ID of the resource that was dropped in a new location.
pParentId	The ID of the new parent
pOrder	The new order value of the resource
pOldParentId	The parent ID of the previous parent.

evExpand

This event is generated when the user expands a category in the tree list

Event Parameter	Description
pResourceId	The ID of the resource that was expanded.

evGAddRel

This event is generated when the user asks to create a new relation by dragging between two phases.

Event Parameter	Description
pPhaseFromId	The ID of the first phase.
pPhaseToId	The ID of the second phase.

evGAddSusp

This event is generated when the user asks to create a new suspension by clicking a phase when in split mode.

Event Parameter	Description
-----------------	-------------

pPhaseId	The ID of the phase.
pSplitDate	The date and time at which the split is to occur.
evGChangeProgress	
This event is generated when the user changed the progress of a phase by dragging inside a phase.	
Event Parameter	Description
pPhaseId	The ID of the phase.
pNewProgress	The value of the new progress in seconds.
evGCreate	
This event is generated when the user asks to create a new phase by dragging in the chart area.	
Event Parameter	Description
pResourceId	The ID of the resource for which the phase is to be created. If zero, a new resource must be created as well.
pNewStartDate	The start date and time of the phase
pNewEndDate	The end date and time of the phase
evGdblClick	
This event is generated when the user double clicks in the chart area.	
Event Parameter	Description
pResourceId	If the ID is non-zero a resource or phase was clicked. (New for version 4.0.0)
pPhaseId	If the ID is non-zero a phase was clicked.
pRelationId	If the ID is non-zero a relationship line was clicked. (New for version 4.0.0)
pMarkerId	If non-zero a marker line was clicked. The ID is the line number of the marker list as it was supplied when loading the chart. (New for version 4.0.0)
evGMove	
This event is generated when the user moves a phase by clicking and dragging.	
Event Parameter	Description
pResourceId	If the ID is non-zero it specifies the resource the phase was dropped onto. See also evPhaseCanDrop and \$gphasecandrop. (New for version 4.0.0)

pPhaseId	If the ID is non-zero a phase was moved.
pNewStartDate	The new start date and time.
pNewEndDate	The new end date and time
evGRClick	
<p>This event is generated when the user right clicks in the chart area. If pPhaseId is zero and pResourceId is non-zero, a resource line was clicked. If pResourceId is also zero, the space below the resources was clicked.</p>	
Event Parameter	Description
pResourceId	If the ID is non-zero a resource or phase was clicked. (New for version 4.0.0)
pPhaseId	If the ID is non-zero a phase was clicked.
pRelationId	If the ID is non-zero a relationship line was clicked (New for version 4.0.0)
pMarkerId	If non-zero a marker line was clicked. The ID is the line number of the marker list as it was supplied when loading the chart. (New for version 4.0.0)
evGResize	
<p>This event is generated when the user resizes a phase by clicking and dragging the right edge of the phase.</p>	
Event Parameter	Description
pPhaseId	If the ID is non-zero a phase was resized.
pNewStartDate	The new start date and time.
pNewEndDate	The new end date and time
evGSelected	
<p>This event is generated when the user selects a phase or background by clicking a phase. If the background of the chart was clicked, pPhaseId will be zero but pResourceId will be set if a line belonging to a resource was clicked. If pResourceId is zero, the space below the resources was clicked. (updated for v4.0.0)</p>	
Event Parameter	Description
pPhaseId	The ID of the phase.
pResourceId	The ID of the resource the phase belongs to.

evHeadLClick	
This event is generated when the user clicks the tree header	
Event Parameter	Description
pColumnNumber	The number of the column that was clicked.
evHeadRClick	
This event is generated when the user right clicks the tree header	
Event Parameter	Description
pColumnNumber	The number of the column that was clicked.
evPhaseCanDrop	
The event evPhaseCanDrop allows you to control which resources a phase being dragged can be dropped on. To prevent the drop, simply discard the event. See also evGMove and \$gphasecandrop. (new for version 4.0.0)	
Event Parameter	Description
<i>none</i>	
evTLDBlClick	
This event is generated when the user double clicks in the tree list.	
Event Parameter	Description
pResourceId	The ID of the resource that was clicked.
pColumnNumber	Number of the column in the tree list that was clicked (v3.8.0)
evTLRClick	
This event is generated when the user right clicks in the tree list. If pResourceId is zero, the space below the resources was clicked.	
Event Parameter	Description
pResourceId	The ID of the resource that was clicked.
pColumnNumber	Number of the column in the tree list that was clicked (v3.8.0)
evTLSelected	
This event is generated when the user selects a resource by clicking in the tree. If pResourceId is zero, the space below the last resource was clicked.	
Event Parameter	Description
pResourceId	The ID of the resource that was selected.

pColumnNumber	Number of the column in the tree list that was clicked (v3.8.0)
evTLTextChanged	
<p>This event is generated when the user changed the text in a cell in the tree. The tree list data is not directly changed. This has to be done by the developer in response to this event and the tree list has to be updated.</p>	
Event Parameter	Description
pResourceId	The ID of the resource.
pColumnNumber	The column number
pNewText	The new text for the cell.
evTLTextEditEnd	
<p>This event is generated if the cursor is about to leave the cell and text has not been altered. Discarding this event will force the cursor to remain in the current cell.</p>	
Event Parameter	Description
pResourceId	The ID of the resource.
pColumnNumber	The column number
pNewText	The text in the cell.
evTLTextEditStart	
<p>This event is generated when a cell is about to be edited. This message cannot be discarded.</p>	
Event Parameter	Description
pResourceId	The ID of the resource.
pColumnNumber	The column number
pNewText	The current text in the cell.

GanttObject Properties		
Name	Type	Description
\$layerid	Integer	The id of the layer. It is used in many of the method calls to the OGanttUI window control and OGanttPrint report control.

GanttObject Methods

\$scaldate()

Syntax: *OGanttObjectRef*.\$scaldate(dStartDate,iOffset,iResourceId)

Calculate date from given date plus positive or negative offset. It will take into account the resources working hours and holidays.

Parameter	Description
dStartDate	The date from which to calculate the new date
iOffset	The offset in seconds
iResourceId	The ID of the resource

\$scalduration()

Syntax: *OGanttObjectRef*.\$scalduration(dStartDate,dEndDate,iResourceId,iPhaseId)

Calculate the duration of a phase based on the given date range. It will take into account the resources working hours and holidays.

Parameter	Description
dStartDate	The start date from which to calculate the duration
dEndDate	The end date from which to calculate the duration
iResourceId	The ID of the resource
iPhaseId	The ID of the phase

\$scalenddate()

Syntax: *OGanttObjectRef*.\$scalenddate(dStartDate,iDuration,iResourceId,iPhaseId)

Calculate the end date of a phase based on the given date and duration. It will take into account the resources working hours and holidays.

Parameter	Description
dStartDate	The start date
iDuration	The duration
iResourceId	The ID of the resource
iPhaseId	The ID of the phase

\$gaddrelation()

Syntax: *OGanttObjectRef*.\$gaddrelation(iRelationId,iPhaseFromId,iPhaseToId,iRelationType,iRelationLag)

Add a relationship between two phases

Parameter	Description
iRelationId	The ID for the new relation
iPhaseFromId	ID of phase from which the relation starts
iPhaseToId	ID of phase to which the relation connects
iRelationType	The type of the relation. One of the kGanttRel... constants.
iRelationLag	The delay of the relationship expressed in seconds.
returns	One of the kGanttRelAdd... constants.
\$gaddsuspension()	
Syntax: <i>OGanttObjectRef</i> .\$gaddsuspension(iPhaseId,iSuspensionId,dSplitDate,iDuration)	
Add a suspension period to the specified phase.	
Parameter	Description
iPhaseId	The ID for the phase.
iSuspensionId	The ID for the new suspension.
dSplitDate	The start date of the period.
iDuration	The duration of the period.
\$gbuild()	
Syntax: <i>OGanttObjectRef</i> .\$gbuild(bFirstLoad)	
Rebuild the chart. This call is made after setting the various chart lists by calls to \$gsetlist()	
Parameter	Description
bFirstLoad	Specify kGanttIsFirstLoad, if this is the first build of the chart. For subsequent rebuilds for adding or updating chart objects, this must specify kGanttNoFirstLoad.
\$gclear()	
Syntax: <i>OGanttObjectRef</i> .\$gclear()	
Clears the chart. Removes all resources, phases, relations and suspensions from the chart.	
Parameter	Description
<i>none</i>	
\$gcollapse()	
Syntax: <i>OGanttObjectRef</i> .\$gcollapse(iResourceId)	
Collapses the specified resource in the chart area.	

Parameter	Description
iResourceId	ID of the resource.
\$gdelphase()	
Syntax: <i>OGanttObjectRef.\$gdelphase(iPhaseId)</i>	
Deletes the specified phase.	
Parameter	Description
iPhaseId	ID of the phase.
\$gdelresource()	
Syntax: <i>OGanttObjectRef.\$gdelresource(iResourceId)</i>	
Deletes the specified resource.	
Parameter	Description
iResourceId	ID of the resource.
\$gdelrelation()	
Syntax: <i>OGanttObjectRef.\$gdelrelation(iRelationId)</i>	
Deletes the specified relation.	
Parameter	Description
iRelationId	ID of the relation.
\$gdelsuspension()	
Syntax: <i>OGanttObjectRef.\$gdelsuspension(iPhaseId,iSuspensionId)</i>	
Deletes the specified suspension.	
Parameter	Description
iPhaseId	ID of the phase the suspension belongs to.
iSuspensionId	ID of the suspension.
\$gexpand()	
Syntax: <i>OGanttObjectRef.\$gexpand(iResourceId)</i>	
Expands the specified resource in the chart area.	
Parameter	Description
iResourceId	ID of the resource.

\$gfreemodlistresource()

Syntax: *OGanttObjectRef*.\$gfreemodlistresource()

Clears the resource modification list. After calling \$ggetmodlistresource() and dealing with changes, you should call this method to clear the list.

Parameter	Description
-----------	-------------

none

\$gfreemodlistphase()

Syntax: *OGanttObjectRef*.\$gfreemodlistphase()

Clears the phase modification list. After calling \$ggetmodlistphase() and dealing with changes, you should call this method to clear the list.

Parameter	Description
-----------	-------------

none

\$gfreemodlistrel()

Syntax: *OGanttObjectRef*.\$gfreemodlistrel()

Clears the relation modification list. After calling \$ggetmodlistrel() and dealing with changes, you should call this method to clear the list.

Parameter	Description
-----------	-------------

none

\$gfreemodlistsusp()

Syntax: *OGanttObjectRef*.\$gfreemodlistsusp()

Clears the suspension modification list. After calling \$ggetmodlistsusp() and dealing with changes, you should call this method to clear the list.

Parameter	Description
-----------	-------------

none

\$ggetdatalist()

Syntax: *OGanttObjectRef*.\$ggetdatalist(iListType)

Get the current list of resources, phases, relations or suspensions from the chart.

Parameter	Description
-----------	-------------

iListType	Which list to return. One of the kGanttList... constants. Can be kGanttListResource, kGanttListPhase, kGanttListRelation or kGanttListSuspension.
-----------	---

\$ggetmodlistphase()

Syntax: *OGanttObjectRef*.\$ggetmodlistphase()

Returns the list of modifications made to phases. After dealing with the changes you should call \$greemodlistphase().

Parameter	Description
-----------	-------------

none

\$ggetmodlistresource()

Syntax: *OGanttObjectRef*.\$ggetmodlistresource()

Returns the list of modifications made to resources. After dealing with the changes you should call \$greemodlistresource().

Parameter	Description
-----------	-------------

none

\$ggetmodlistrel()

Syntax: *OGanttObjectRef*.\$ggetmodlistrel()

Returns the list of modifications made to relations. After dealing with the changes you should call \$greemodlistrel().

Parameter	Description
-----------	-------------

none

\$ggetmodlistsusp()

Syntax: *OGanttObjectRef*.\$ggetmodlistsusp()

Returns the list of modifications made to suspensions. After dealing with the changes you should call \$greemodlistsusp().

Parameter	Description
-----------	-------------

none

\$ggettotals()

Syntax: *OGanttObjectRef*.\$ggettotals(iCalculationType,iFieldNumber,bToGroupValues)

Get a list of total values from the chart. The total values are calculated based on specified calculation type and the field values in the phases.

Parameter	Description
-----------	-------------

iCalculationType	How to calculate the totals. One of the kGanttCalc... constants.
------------------	--

iFieldNumber	The number of the field.
--------------	--------------------------

bToGroupValues	If kTrue, totals are grouped according to the major scale.
returns	List of totals
\$gindent()	
Syntax: <i>OGanttObjectRef</i> .\$gindent(iResourceId)	
Indent the specified resource.	
Parameter	Description
iResourceId	ID of the resource.
\$gmakecategory()	
Syntax: <i>OGanttObjectRef</i> .\$gmakecategory(iResourceId,IPhaseData)	
Convert the phases of a resource into a category	
Parameter	Description
iResourceId	ID of the resource.
IPhaseData	List of phase data
\$gmakephase()	
Syntax: <i>OGanttObjectRef</i> .\$gmakephase(iResourceId)	
Convert a category into a phase.	
Parameter	Description
iResourceId	ID of the resource.
\$gmovephase()	
Syntax: <i>OGanttObjectRef</i> .\$gmovephase(iPhaseId,dNewStartDate)	
Move the specifies phase to the new date and time.	
Parameter	Description
iPhaseId	ID of the phase.
dNewStartDate	The new start date and time.
\$grsetfuncforfield()	
Syntax: <i>OGanttObjectRef</i> .\$grsetfuncforfield(iFieldNumber,iCalculationType,iParameter)	
Set the function for calculating the specified field.	
Parameter	Description
iFieldNumber	Number of the field
iCalculationType	One of the kGanttCalc... constants.

iParameter	The parameter for the calculation.
\$grsetfieldtoalc()	
Syntax: <i>OGanttObjectRef</i> .\$grsetfieldtoalc(iField1,iField2,iField3, iColor1,iColor2,iColor3)	
Specifies on which fields to carry out the calculations.	
Parameter	Description
iField1..iField3	Numbers of the fields
iColor1..iColor3	Colour for the display
\$gsetcalendar()	
Syntax: <i>OGanttObjectRef</i> .\$gsetcalendar(lDefaultDay,lHolidayList,lExceptionList)	
Sets the working hours, holidays and working hours exceptions.	
Parameter	Description
lDefaultDay	List of default working hours
lHolidayList	List of non-working days, i.e. holidays and weekend days.
lExceptionList	List of dates and specific working hours for these dates.
\$gsetlist()	
Syntax: <i>OGanttObjectRef</i> .\$gsetlist(lList,lListType)	
Set a list for the chart. After setting all the lists you must call \$gbuild() to update the chart.	
Parameter	Description
lList	List of resources, phases, relations or suspensions.
lListType	One of the kGanttList... constants. Can be kGanttListResource, kGanttListPhase, kGanttListRelation or kGanttListSuspension.
\$gunindent()	
Syntax: <i>OGanttObjectRef</i> .\$gunindent(iResourceId)	
Un-indent the specified resource.	
Parameter	Description
iResourceId	ID of the resource.
\$gupdateondd()	
Syntax: <i>OGanttObjectRef</i> .\$gupdateondd(iResourceId,iParentId,nOrder)	
Tells the external to do the work in response to a drag&drop message. Must also call \$tlupdateondd().	

Parameter	Description
iResourceId	ID of the resource.
iParentId	New parent ID
nOrder	New order.
\$gupdorder()	
Syntax: <i>OGanttObjectRef</i> .\$gupdorder(iResourceId,nNewOrder)	
Update the order of a resource in the chart. Must also call \$tlupdorder().	
Parameter	Description
iResourceId	ID of the resource.
nNewOrder	The new order.
\$gupdphase()	
Syntax: <i>OGanttObjectRef</i> .\$gupdphase(iPhaseId,lNewData)	
Update the specified phase with the new data.	
Parameter	Description
iPhaseId	ID of the phase.
lNewData	List with new data for the phase.
\$gupdrelation()	
Syntax: <i>OGanttObjectRef</i> .\$gupdrelation(iRelationId,iPhaseFromId,iPhaseToId,iRelationType,iRelationLag)	
Update the specified relation.	
Parameter	Description
iRelationId	ID of the relation.
iPhaseFromId	ID of phase from which the relation starts
iPhaseToId	ID of phase to which the relation connects
iRelationType	The type of the relation. One of the kGanttRel... constants.
iRelationLag	The delay of the relationship expressed in seconds.
returns	One of the kGanttRelAdd... constants.

\$gupdsuspension()

Syntax: *OGanttObjectRef*.\$gupdsuspension(*iPhaseId*,*iSuspensionId*,*dNewStartDate*,*dNewEndDate*)

Update the specified suspension.

Parameter	Description
<i>iPhaseId</i>	ID of the phase.
<i>iSuspensionId</i>	ID of the suspension.
<i>dNewStartDate</i>	New start date.
<i>dNewEndDate</i>	New end date.

\$tlbuild()

Syntax: *OGanttObjectRef*.\$tlbuild(*bFirstLoad*)

Rebuild the tree. This call is made after setting the various chart lists by calls to \$tlsetlist()

Parameter	Description
<i>bFirstLoad</i>	Specify <i>kGanttIsFirstLoad</i> , if this is the first build of the chart. For subsequent rebuilds for adding or updating chart objects, this must specify <i>kGanttNoFirstLoad</i> .

\$tlbuild()

Syntax: *OGanttObjectRef*.\$tlbuild(*bFirstLoad*)

Update the specified suspension.

Parameter	Description
<i>iPhaseId</i>	ID of the phase.

\$tlclear()

Syntax: *OGanttObjectRef*.\$tlclear()

Clears the tree. Removes all cells from the tree list.

Parameter	Description
<i>none</i>	

\$tlcollapse()

Syntax: *OGanttObjectRef*.\$tlcollapse(*iResourceId*)

Collapse the specified resource in the tree.

Parameter	Description
<i>iResourceId</i>	ID of the resource.

www.brainydata.com

\$tldelnode()

Syntax: *OGanttObjectRef*.\$tldelnode(*iResourceId*)

Deletes a row in the tree.

Parameter	Description
<i>iResourceId</i>	ID of the resource.

\$tlexpand()

Syntax: *OGanttObjectRef*.\$tlexpand(*iResourceId*)

Expand the specified resource in the tree.

Parameter	Description
<i>iResourceId</i>	ID of the resource.

\$tlgetcelldata()

Syntax: *OGanttObjectRef*.\$tlgetcelldata(*iResourceId*,*iColumnNumber*)

Return the data for the specified cell from the tree

Parameter	Description
<i>iResourceId</i>	ID of the resource.
<i>iColumnNumber</i>	Number of the column.
returns	The cell's data

\$tlgetcolwidth()

Syntax: *OGanttObjectRef*.\$tlgetcolwidth()

Returns a comma separated string of the tree list column widths

Parameter	Description
returns	String of column widths

\$tlgetprevsibling()

Syntax: *OGanttObjectRef*.\$tlgetprevsibling(*iResourceId*)

Return the ID of the previous sibling of the specified resource.

Parameter	Description
<i>iResourceId</i>	ID of the resource.
returns	ID of previous sibling

\$tlgetdatalist()

Syntax: *OGanttObjectRef*.\$tlgetdatalist(iColumnNumber)

Returns the tree data list for the specified column.

Parameter	Description
iColumnNumber	The column number
returns	The data list

\$tlindent()

Syntax: *OGanttObjectRef*.\$tlindent(iResourceId)

Indent the specified resource or category in the tree.

Parameter	Description
iResourceId	ID of the resource.

\$tlisexpanded()

Syntax: *OGanttObjectRef*.\$tlisexpanded(iResourceId)

Return the expanded state of the specified resource.

Parameter	Description
iResourceId	ID of the resource.
returns	kTrue or kFalse

\$tlmakecategory()

Syntax: *OGanttObjectRef*.\$tlmakecategory(iResourceId)

Converts the specified resource to a category.

Parameter	Description
iResourceId	ID of the resource.

\$tlmakesresource()

Syntax: *OGanttObjectRef*.\$tlmakesresource(iResourceId)

Converts the specified category to a resource.

Parameter	Description
iResourceId	ID of the resource.

\$tlsetheader()

Syntax: *OGanttObjectRef*.\$tlsetheader(lHeaderList)

Set the attributes for the column headers of the tree list.

Parameter	Description
IHeaderList	List of attributes, one row per column.
\$tlsetlist()	
Syntax: <i>OGanttObjectRef.\$tlsetlist(IList,iListType)</i>	
Set the data for a column in the tree list	
Parameter	Description
IList	The data. Content depends on list type
iListType	One of the kGanttList... constants. Can be kGanttListTree or kGanttListColumn.
\$tlunindent()	
Syntax: <i>OGanttObjectRef.\$tlunindent(iResourceId)</i>	
Un-indent the specified resource or category in the tree.	
Parameter	Description
iResourceId	ID of the resource.
\$tlupdateondd()	
Syntax: <i>OGanttObjectRef.\$tlupdateondd(iResourceId,iParentId,nOrder)</i>	
Tells the external to do the work in response to a drag&drop message. Must also call \$gupdateondd().	
Parameter	Description
iResourceId	ID of the resource.
iParentId	New parent ID
nOrder	New order.
\$tlupdatecell()	
Syntax: <i>OGanttObjectRef.\$tlupdatecell(iResourceId,iColumnNumber,I newData)</i>	
Update a cell in the tree list with new data.	
Parameter	Description
iResourceId	ID of the resource.
iColumnNumber	The column number.
I newData	The new data.

\$tluporder()

Syntax: *OGanttObjectRef*.\$tluporder(iResourceId,nNewOrder)

Update the order of a resource or category in the tree list. Must also call \$gupdorder().

Parameter	Description
iResourceId	ID of the resource.
nNewOrder	The new order.