VARCHART
XGantt

ActiveX Edition 5.2
User's and
Reference Guide

NETRONIC

VARCHART XGantt ActiveX Edition

Version 5.2

User's Guide

NETRONIC Software GmbH
Pascalstrasse 15
52076 Aachen
Germany
Phone +49 (0) 2408 141-0
Fax +49 (0) 2408 141-33
Email sales@netronic.com
www.netronic.com

Last Revision: 27 April 2020

Table of Contents

1 Introduction

1.1 VARCHART XGantt at a Glance

Gantt charts allow to display and plan the chronological sequence of tasks and the capacity of resources. Due to their graphical visualization, interrelations and changes become obvious at a glance. Besides being employed in the project management, Gantt diagrams have been established above all in control panels of the manufacturing and in systems of resource management and disposition.

VARCHART XGantt is an interactive graphic component which can easily be integrated into your own applications within short time because there is no time-consuming programming of graphical charts. Due to the great variety of layout options, VARCHART XGantt meets individual graphical demands. The print-out is of first-class quality.

> **The functionalities of VARCHART XGantt are:**

- Creating, deleting or shifting of nodes
- Creating and deleting of links
- Visualization of date fields by bars or symbols
- Data driven allocation of graphical attributes
- Sorting and grouping according to various criteria
- Collapsing or expanding of groups of activities
- Variable structure of the time scale
- Flexible design of the table area
- Adding of date lines and line grids
- Continuous zooming of diagrams
- Zooming of diagram sections to full screen size
- Integrated page preview and print-out with paging
- Exchange of the application data via files or the programming interface
- Various design options for histograms
- Easy customization of properties via the property pages
- Customization of default interactions via events

- Powerful programming interface
- The Resource Scheduling module of VARCHART XGantt supports the conception of interactive decision making. It unifies both, generating schedules automatically after pre-defined strategies and taking individual constraints into account.

Note: All source code samples of this documentation are written in Microsoft Visual Basic 6.0.

1.2 Technical Requirements

To develop an application using the VARCHART ActiveX control you will need

- operating system, Server 2003, Vista, Windows 7 or Windows 8.
- a development environment that supports the integration of ActiveX controls such as Visual C++, Visual Basic, Visual Fox Pro, Delphi, Centura, Oracle Forms, Progress, HTML (Visual Basic Script)
- about 50 MB hard disk space.

1.3 Installation

Start the **Setup** program and follow the instructions.

During the installation procedure, a reference of the VARCHART ActiveX component is registered in the Windows registry. You can run the registration yourself using the Windows system file *regsvr32.exe*:

- c:\windows\system32\regsvr32 "c:\program files\varchart\xgantt\-vcgantt.ocx"

The specified paths certainly depend on the settings of your computer.

The installation procedure is logged to the file *install.log* allowing for tracing where files were copied.

The same file will be used for uninstalling. You can start the uninstalling procedure by selecting **Start** -> **Settings** -> **Control Panel** and then **Add/Remove Programs**.

You can remove the registration entry yourself by using the command

- c:\windows\system32\regsvr32 -u "c:\program files\varchart\xgantt\-vcgantt.ocx"

Alternatively, you can make an unattended installation of VARCHART XGantt. For this, please enter:

start/wait (NameOfTheSetupFile).exe /L1033 /s /V"/qn ADDLOCAL=ALL"

By this call, the installation will run without user interaction and without status information displayed on the screen. Please note:

1. The invoking procedure, such as a DOS box, needs to be run with administrator privileges; otherwise a UAC message may appear that requests a user entry.
2. Language parameters: /L1033: installation in English; /L1031: installation in German; L2052: installation in Chinese
3. Progress information: /qb: progress information will be displayed; /qn: no progress information will appear; you won't see anything on the screen.
4. Start/wait you should use in case the installation is run by a batch file; if you don't use 'wait', the batch file will run parallel to the installation.

1.4 Licensing

For licensing the VARCHART XGantt control please click on the icon  and draw the control onto the form.

Open the **Property Pages** by a right mouse click on the control.

On the **General** tab, please open the licensing dialog by clicking on the **Licensing...** button.

By clicking on the button **Request license information from NETRONIC...** a dialog to fill in the user data will open.

Four items are needed for the licensing:

- the hardware identification
- the license number
- the name of the staff member
- the name of the company

Please fill in the information needed. You will find the license number "BXnnnn" on the delivery note of your order.

If you click on **Send email to NETRONIC...**, an email will be generated that only needs to be dispatched. Alternatively, you can write an email manually that contains the required information. Please send all enquiries concerning the licensing to license@netronic.com

After sending the mail, you will immediately receive a license file. To finish the licensing procedure, please copy the file to the installation directory (directory that contains the file **vcgantt.ocx**).

1.5 Delivery

When delivering your application, please check if the below files are present in your customer´s Windows directory. If they are not present, you need to include them in your shipment:

VARCHART XGantt files:

- *vcgantt.ocx* (version 5.0)
- *vcpane32u.dll* (version 5.5)
- *vcprct32u.dll* (version 5.5)
- *vcwin32u.dll* (version 5.5)
- *vxcsv32u.dll* (version 1.320)
- *opsaps.dll* (version 7.3)

Microsoft libraries:

- *gdiplus.dll*
- *mfc100u.dll*
- *msvcp100.dll*
- *msvcr100.dll*

The file *vcgantt.ocx* needs to be registered by using the command line *regsvr32 vcgantt.ocx*.

In order to install the libraries *mfc100u.dll, msvcp100.dll, mfcm100u.dll* and *msvcr100.dll* you can either copy them directly to the Windows system directory or you can use the setup file *vcredist_vs2010_x86.exe*. These files are located in the installation folder of XGantt in the subfolder **redist**.

The below files **must not** be shipped to the end user:

- *vcgantt.lic* (contains your developer license)
- *vcgantt.chm* (online help file for developers)

1.6 VARCHART ActiveX in Visual Studio 6.0 or 7.0 with Visual C++/MFC

To insert a VARCHART ActiveX control in your MFC project, please proceed as follows:

Visual Studio 6.0:

In the **Project** menu select the item **Add To Project...** and then the subitem **Components and Controls**. In the dialog box which appears then select the NETRONIC VARCHART ActiveX from the registered controls and click on the **Insert** button. After a control question a dialog box appears. In the listbox deselect all MFC wrappers created by the wizard except the first class (this is not possible). Click on the **OK** button. Then click on the **Close** button to close the dialog box.

Visual Studio 7.0:

In the context menu of a dialog resource select the item **Insert ActiveX Control...** and transfer the selected ActiveX control to the dialog. Then create an instance variable and a DDX_CONTROL entry in the DoDataExchange method either manually or with the help of the wizard via the context menu (menu item **Insert Variable...**). In the latter case also a MFC wrapper will be created automatically. Alternatively you can create MFC wrappers in the ClassView (inclusive the ones for the subobjects), but then the Enum definitions will be missing.

Thus both development environments offer the automatical creation of MFC wrappers. With the help of these wrappers you can use the methods and properties of the ActiveX control in the same way as for normal MFC objects. Without wrappers you would have to study more intensively the OLE conventions. But the created wrappers are not really satisfactory:

- The automatically generated files do not contain Enum definitions (only Visual Studio 6.0).
- All subclasses are stored in separate files. That makes it impossible to use different VARCHART ActiveX controls at the same time (Visual Studio 6.0). In Visual Studio 7.0 subclasses are not generated; thus they cannot be used at all.
- For API updates of the controls the update of the wrappers would be possible only indirectly. Furthermore, Visual Studio 7.0 uses different name conventions than older versions. This would make changes in older projects necessary (new name prefixes: **get_** and **set_** for properties instead of **Get** and **Set**).

- If you want to use several VARCHART ActiveX controls in one project, name conflicts with the subobjects will occur.

Therefore NETRONIC Software GmbH offers an own pair of MFC wrapper files: *xgantt.h* and *xgantt.cpp*. This file is stored in the subdirectory MFC of the installation directory of the VARCHART ActiveX control. It contains all wrappers and the helpful Enum definitions.

All definitions have been put into a namespace so that you can use several VARCHART ActiveX controls in one project without name conflicts in case of subobjects that appear several times.

Remove the automatically created wrappers from your project, add the cpp file to your project, and import the header file into the dialog class.

If you use only one control in a class, the below code lines will be sufficient:

Example Code

```
#include "xgantt.h"
using namespace XGantt;
```

If you use several VARCHART ActiveX controls in one class, you have to place the namespace in front of each subobject that appears in at least two controls (e.g. CVcNode or CVcTitle) in addition. The following example demonstrates the declaration of a variable for a title object:

Example Code

```
XGantt::CVcTitle title = VcGantt1.GetTitle();
```

In the event procedures instead of objects only the LPDISPATCH pointers are passed. These pointers can be connected to the object via the corresponding **Attach** method of the object. Then you should not forget to enter **Detach()** at the end of the usage of the object.

If you have started projects with the generated files, a change should not be difficult, since NETRONIC uses the files generated by Visual Studio 6.0 as basis so that they should be compatible. The only difference is the usage of namespaces in order to make the names of subobjects clear.

1.7 VARCHART ActiveX in HTML Pages

In this chapter it is shown how to get VARCHART ActiveX controls working in a HTML page and how to control them by script. Two different ways of embedding exist: direct embedding and embedding an ActiveX contol which contains a VARCHART ActiveX control. The former is suitable for small web applications, whereas for larger web applications, you should develop your own ActiveX control, which most development environments allow for.

1.7.1 Restrictions

Compared to other applications, there are some restrictions:

- The client used needs to be run by the Windows operating system, since it is the only system that runs ActiveX controls. This is not required of the server.
- If you embed the ActiveX control directly, Javascript/JScript (ECMAScript) is not suitable as a script language because it does not offer by-reference parameters, which makes it impossible to return values other than the return value itself, for example the methods **IdentifyObjectAt** and most of the events, e.g. **OnNodeCreate**. VBScript however, offered only by the Microsoft Internet Explorer, is suitable.
- Mozilla browsers (including Firefox and Netscape) and Opera are only appropriate for direct embedding, if an ActiveX plug-in is used. There is the solution of Mozilla ActiveX Project and the plug-in MeadCo Neptune, which works independently of browsers. By the way, Mozilla Active X Projext does not offer a "silent" installation by a CAB file, which is the default with the Internet Explorer.

Please consider that direct embedding and the cosecutive management of the VARCHART ActiveX control by a script cannot replace a real application. Scripts are only suitable for small applications. If you plan a larger application, you should develop your own ActiveX control, e.g. by using Visual Basic 6.0, containing one or several VARCHART ActiveX controls. For example a script cannot access the mass storage of the target computer, whereas an ActiveX control is able to do this (even if it is not supposed to).

1.7.2 Implementation Including Direct Embedding

The below section describes how to directly implement VARCHART ActiveX controls into HTML pages in the Microsoft Internet Explorer by using the script language VBScript.

The ActiveX control is embedded into the HTML page by an OBJECT tag:

Example Code

```
<OBJECT ID="VcGantt1" WIDTH=700 HEIGHT=350
 CLASSID="CLSID:A4E79A20-C9E1-11CF-BDD7-02608C4302A9"
 CODEBASE="vcgantt.cab#version=4,000,0,0">
</OBJECT>
```

The command specifies the size and the Class ID of the VARCHART ActiveX control. Each VARCHART ActiveX control has got a unique Class ID by which it is identified if it was recorded in the registry before. If an ActiveX control is to be displayed without an explicite installation, the code base parameter will be used. It specifies where the associated installation file is located on the server. The CAB file to be specified there is delivered by NETRONIC Software GmbH. In addition, the version number has to be specified to make sure that the control is loaded and installed whenever there is no or just an old version on the target computer.

The CAB file was signed by NETRONIC Software GmbH, so that the user in the Internet Explorer will receive a message on the certification when the browser starts to install the control. The VARCHART ActiveX control on purpose was not signed as safe ("Safe for Scripting") for the use in script languages, since writing to the file system of the computer is possible by the export of charts and the **SaveAs** method. If you develop your own ActiveX contol, you should sign it as safe for the installation and for the use in script languages (for example by the **Package and Deployment Wizard** of Visual Basic 6.0), to ensure a use free of problems on the Internet.

After embedding the VARCHART ActiveX control in the HTML page, you now need to provide your own configuration file to make the VARCHART ActiveX contol show the desired appearance. For this, you need a script in which the property **ConfigurationName** of the VARCHART ActiveX control points to a URL (needs to start by **http:**//), which preferably describes a file located in the same directory on the server as the other files.

Example Code

```
VcGantt1.ConfigurationName =
"http://www.netronic_test.com/xgantt_sample.ini"
```

Please note that not only the INI file of the VARCHART ActiveX control but also an IFD file with the same name are read. Both have to be located on the server. The files can be generated in the following way: Drag the

VARCHART ActiveX control into a development environment and configure it by its property pages. Then save the configuration files by the property page **General**. By doing so, your licence will also be stored to the configuration file, which is vital to using the ActiveX control.

A little web application is delivered amongst the programming samples.

If the URL of the INI file is known while the HTML page is written (i. e. if it does not have to be determined by script), you can assign the configuration file by the <PARAM> tag within the <OBJECT> tag. The advantage is that the ActiveX control initially shows the valid settings such as colors, proportions etc., but abstains from temporarily showing the default settings.

Example Code

```
<OBJECT CLASSID=...>
<PARAM NAME="ConfigurationName"
       VALUE="http://www.netronic.de/mysample.ini">
</OBJECT>
```

Note: Former releases of the VARCHART ActiveX controls were marked by "Licensed", so that in the HTML page the License Manager had to be addressed. This has been eliminated now; nevertheless the former code will comply with present and future releases.

1.7.3 Implementation Including Indirect Embedding

If you develop your own ActiveX control which contains a VARCHART control, in terms of the embedding you can proceed in a similar way as described above.

Beside, for the "silent" automatic installation in the Internet Explorer you need to generate a CAB file of your own. This is possible for example by the **Package and Deployment Wizard** of Visual Basic 6.0, which was mentioned earlier, and by the free command line tool **cabarc** of the Microsoft Cabinet SDK. The CAB file should contain the same files that are present in the CAB file delivered with the VARCHART ActiveX controls. For this, you can extract the contents of the CAB file by commercial ZIP tools or by **cabarc**. The installation is controlled by an INF file, that you can adapt yourself or that can be generated by the **Package and Deployment Wizard**. Alternatively, for generating a CAB file, you can use the tool **IExpress** which is delivered with later Windows versions and originates from the IEAK (Internet Explorer Administration Kit).

In addition, you need to sign your own controls and CAB files, since only then they can be used in the Internet Explorer (this may be modified for certain zones in the **Internet options** menu, but often it is not desired).

Signing is possible by acquiring a code signature from a certification authority (lists see below) and by signing your DLL, OCX and finally your CAB files. This requires to use the free command line tool **signcode** from the Microsoft platform SDK or **signtool** from the Microsoft .NET Framework SDKs.

1.7.4 Trouble-Shooting

If problems occur when executing ActiveX controls in the Internet Explorer, the free tool **Code Download Log Viewer** of Microsoft has proved to be helpful. It allows to trace the parts that did not work during the download. Also the Script debuggers can be recommended, such as the free **Microsoft Script Debugger**.

When downloading INI and IFD files from an IIS web server, please note that these file types have to be made known to the web server by invoking the dialog **file types** properties of the web sites in the tree view of the Internet Information Service on the tab **HTTP Header** and by allocating INI and IFD file types to the MIME type **text/plain**.

It should not be ignored, that often scripts on the server need to be debugged, which is possible by using development environments of web applications (for example using Microsoft FrontPage for ASP). Scripts on the server side imply the problem not to allow for simple things such as message boxes and log files to mark bugs in the script.

> References for solving problems and for further technical information:

OBJECT Tag which specifies component FileVersion and #Version

http://support.microsoft.com/kb/167597

How To Implement IObjectSafety in Visual Basic 6.0 Controls

http://support.microsoft.com/kb/182598

Mozilla ActiveX Project

http://www.adamlock.com/mozilla/

MeadCo Neptune

www.meadroid.com/neptune

Microsoft Cabinet SDK

http://support.microsoft.com/kb/310618

Microsoft IExpress

www.microsoft.com/technet/prodtechnol/ie/ieak/techinfo/deploy/60/en/iexpress.mspx?mfr=true

Code Download Log Viewer (CDLLOGVW)

http://msdn.microsoft.com/archive/default.asp?url=/archive/en-us/samples/internet/browsertools/cdllogvw/default.asp

Microsoft Script Debugger

www.microsoft.com/downloads/details.aspx?FamilyID=2f465be0-94fd-4569-b3c4-dffdf19ccd99&DisplayLang=en

Code signing

http://msdn.microsoft.com/library/default.asp?url=/workshop/security/authcode/intro_authenticode.asp

Certification authorities

VeriSign: www.verisign.com/developer

Thawte: www.thawte.com

GeoTrust: www.geotrust.com

GlobalSign: www.globalsign.net

Signcode tool

http://msdn.microsoft.com/library/default.asp?url=/workshop/security/authcode/signing.asp

Signtool tool

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/seccrypto/security/signtool.asp

1.8 Support and Advice

Are you wondering whether VARCHART XGantt is going to meet the special requirements of your Gantt chart?

Are you trying to make a plan of how much effort it could be to program a special feature of your Gantt chart?

Have you just started testing VARCHART XGantt and are you wondering how to get to a special feature of your Gantt chart?

We would be glad to assist you with any queries you may have. Please contact

NETRONIC Software GmbH

Pascalstr. 15

52076 Aachen

Germany

Phone +49-2408-141-0

Fax +49-2408-141-33

Email support@netronic.com

www.netronic.com

...by the way: you may order our support and maintenance service which goes beyond the 30 days of free support during the initial testing phase. The service includes:

- A support hotline
- Detailed expert advice to questions of application
- Quick fixing of possible bugs in the software
- Upgrades to new VARCHART XGantt releases for development and runtime versions.

We also offer training classes and workshops (at your or at our place).

2 Tutorial

2.1 Overview

In this chapter, we will get you aquainted with the basic features of VARCHART XGantt which are essential for integrating the bar chart into your own application.

Step by step, we will explain to you the important aspects of VARCHART XGantt for the application development and go into the particulars of the wide range of designing options. We recommend to read this tutorial chapter by chapter, while the other parts of the user guide rather serve for consulting on specific situations.

- **Property pages and dialogs**

  In the quoted chapter you will find comprehensive information on the property pages and dialogs which allow to configure VARCHART XGantt at design time without having to write code.

- **Elements of the user interface**

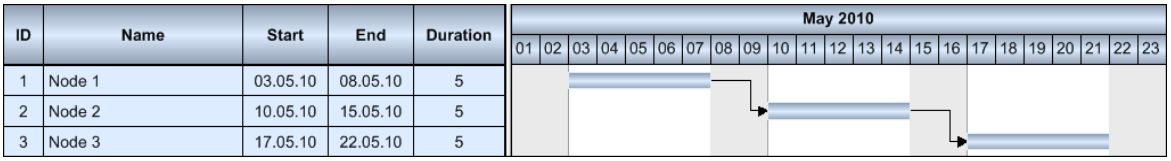
  In the chapter quoted above the interactions which are available in the diagram are described. Details of the user interface can be fitted or changed individually.

- **API Reference**

  In the above chapter you will find detailed information on all objects, properties, methods and events of VARCHART XGantt.

We use Visual Basic 6.0 as developing environment for the samples.

Our first progam sample will show the below result:



ID	Name	Start	End	Duration
1	Node 1	03.05.10	08.05.10	5
2	Node 2	10.05.10	15.05.10	5
3	Node 3	17.05.10	22.05.10	5

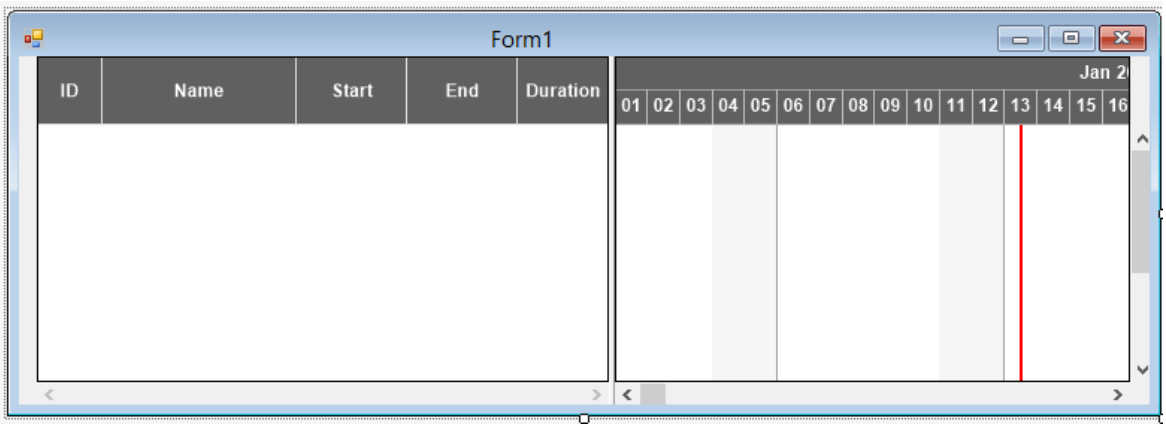
You will find the starter sample in the folder **Programs\UserGuideSamples-\XGantt_Tutorial01_App**

With this application, you will get to know better the inbuilt interactions of VARCHART XGantt.

2.2 Placing the Control on a Form

To place the VARCHART XGantt control on a form, please click on its icon in the toolbox and, using the mouse, draw a frame at the desired position in the form. In order to include the below sample in the developing environment, please load the configuration file "ActiveX-Sample.ini". In that file the same settings of names, colors and measures are used as in the below paragraphs. How to import a configuration file is described at the end of this tutorial in the chapter "Saving and loading the configuration".

If you run the program now, the result should correspond to the illustration.



If you wish the bottom and right-hand side of the VARCHART Windows Forms control to be adjusted to the full size of the window during runtime, the Load and Resize event of the form must contain the following code:

Example Code

```
Private Sub Form_Load()
    VcGantt1.Width = ScaleWidth - VcGantt1.Left
    VcGantt1.Height = ScaleHeight - VcGantt1.Top
End Sub

Private Sub Form_Resize()
    VcGantt1.Width = ScaleWidth - VcGantt1.Left
    VcGantt1.Height = ScaleHeight - VcGantt1.Top
End Sub
```

Note: The VARCHART XGantt control inserted is called **VcGantt1** in this example and in the ones following. This name automatically is assigned by the developing environment but can be modified if desired.

2.3 Supplying Data

In order to display activities and links VARCHART XGantt needs to be supplied with data. By default, the communication required is realized by two tables:

1. Maindata
2. Relations

By loading the data file **samples.ini** the tables are filled by the below data:

Fields of the Maindata table:

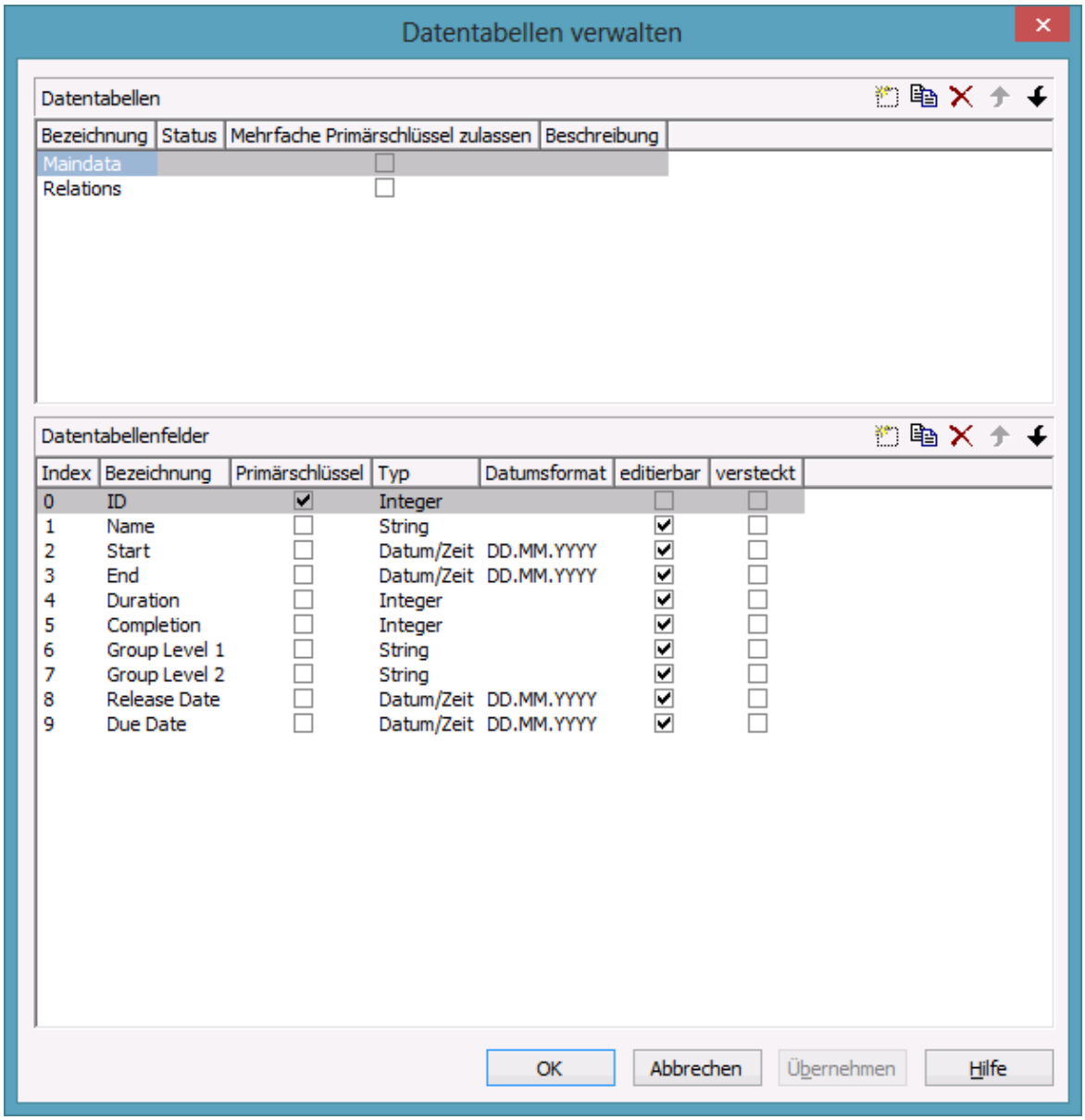
Index	Name	Primary key	Type	Date format
0	ID	True	Integer	
1	Name	False	Alphanumeric	
2	Start	False	Date/Time	DD.MM.YYYY
3	Ende	False	Date/Time	DD.MM.YYYY
4	Duration	False	Integer	

Fields of the Relations:

Index	Name	Primary key	Type	Date format
0	Link ID	True	Alphanumeric	
1	Predecessor Node ID	Alphanumeric		
2	Successor Node ID	Alphanumeric		

Additionally required fields have to be defined manually. You can do this at design time via the dialog **Edit data table** or at run time via the method **Add(...)** of the object **VcDataTableFieldCollection**.

If you need more tables than the two defined by default you can create them on the property page **Administrate data tables** after having clicked **Extended data tables enabled** on the property page **General**. The fields needed for the new tables you can create (or edit) in the dialog **Edit data table**.



The method **DataRecordByID()** of **VcDataRecordCollection** permits to quickly find objects by means of the primary key.

In order to make activities and links visible in our starter sample, you need to enter some records into the data table first.

This you can do by using the method **Add(...)** of the object type **VcDataRecordCollection**. The method **EndLoading** completes the data input for the corresponding chart be composed. For this, please enter the below code lines in the **Load** event of the form.

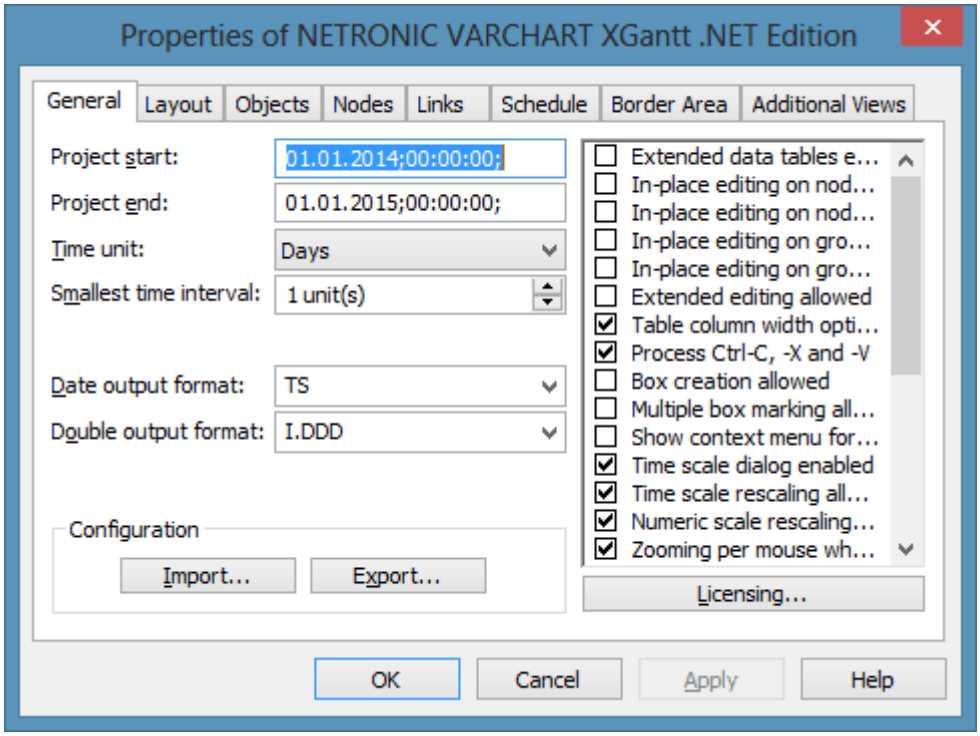
Example Code

```
Set dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
Set dataRecCltn = dataTable.DataRecordCollection
dataRecCltn.Add "1;Node 1;07.05.2007;;5"
dataRecCltn.Add "2;Node 2;14.05.2007;;5"
dataRecCltn.Add "3;Node 3;21.05.2007;;5"
```

```
Set dataTable =
VcGantt1.DataTableCollection.DataTableByName("Relations")
Set dataRecCltn = dataTable.DataRecordCollection
dataRecCltn.Add "1;1;2"
dataRecCltn.Add "2;2;3"

VcGantt1.EndLoading
```

The values in a record are separated by semicolons. The order of the fields has to correspond to the order of the fields in the data definition. New records have to have an unambiguous identification which is not empty. The date in the record has to correspond to the DateFormat definition in the data definition table. The interpretation of the duration depends on the **Time unit** and is pre-set to **days** on the **General** property page.

The **Date output format** is defined consistently for the table and every dialog on the **General** property page.



> Loading data from a CSV file

Alternatively, you may also load the data from what is called a CSV file. The structure of the file has to correspond to the below scheme:

Example Code

```
1;Node 1;07.05.2007;;5
2;Node 2;14.05.2007;;5
3;Node 3;21.05.2007;;5
****
1;1;2
2;2;3
```

Every record has its own line. The contents of the lines correspond to the delivery parameters of the method **Add(...)** of the object type **VcDataRecordCollection**.

The records of the Maindata are listed first, afterwards the records of the Relations. Use **** Table name **** in order to mark the beginning of each record group.

If you have saved such a file under **intro.csv** e.g., you may import the data as follows:

Example Code

```
VcGantt1.Open("c:\intro.csv")
```

> Specifying the period of time which is represented

Up to this point, activities remain invisible, since the time scale has not yet been adapted to the period in which the nodes were positioned. The range of the time scale to be displayed can either be defined by the properties **TimeScaleStart** and **TimeScaleEnd** or they can be determined from the data by the method **OptimizeTimeScaleStartEnd(...)** of the object **VcGantt**.

Example Code

```
 VcGantt1.TimeScaleEnd = DateSerial(2008, 1, 1)
 VcGantt1.TimeScaleStart = DateSerial(2007, 5, 4)
```

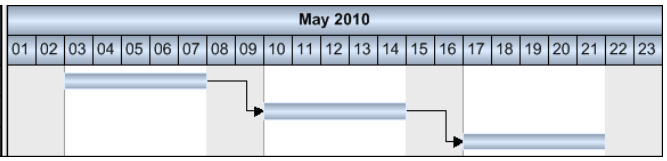
Below the code lines are listed that needed for our starter sample.

Example Code

```
Private Sub Form_Load()
    VcGantt1.Width = ScaleWidth - VcGantt1.Left
    VcGantt1.Height = ScaleHeight - VcGantt1.Top

Set dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
Set dataRecCltn = dataTable.DataRecordCollection
dataRecCltn.Add "1;Node 1;07.05.2007;;5
dataRecCltn.Add "2;Node 2;14.05.2007;;5"
dataRecCltn.Add "3;Node 3;21.05.2007;;5"

Set dataTable =
VcGantt1.DataTableCollection.DataTableByName("Relations")
Set dataRecCltn = dataTable.DataRecordCollection
dataRecCltn.Add "1;1;2"
dataRecCltn.Add "2;2;3"

    VcGantt1.EndLoading

    VcGantt1.OptimizeTimeScaleStartEnd (3)
End Sub

Private Sub Form_Resize()
    VcGantt1.Width = ScaleWidth - VcGantt1.Left
    VcGantt1.Height = ScaleHeight - VcGantt1.Top
End Sub
```

If you run the program now, the result should correspond to the illustration.

ID	Name	Start	End	Duration
1	Node 1	03.05.10		5
2	Node 2	10.05.10		5
3	Node 3	17.05.10		5



2.4 Calculating End Dates

The table column with the end dates is still empty. The end of an activity can be calculated from the fields **Start** and **Duration** with the help of the calendar which is included in VARCHART XGantt .

In the default calendar the weekdays (monday to friday) are defined as active times and the weekends (saturday and sunday) are defined as non active times.

In the diagram, you can recognize the non active times by the gray background. The calendar may be switched off by deactivating the option **Assign calendar to nodes** on the **Nodes** property page.

Please note the difference in calculating with or without calendar:

An activity which starts on friday and lasts 3 days will end on tuesday if the calendar is activated. Without calendar, the activity will end on sunday already.

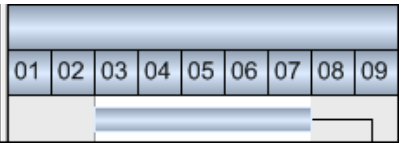
The end date is calculated via the method **AddDuration(...)** of the object **VcCalendar**. For this reason, **start** and **duration** of each activity are needed. They can be retrieved from the corresponding data fields via the index. After having set the end date via the method **DataField(...)**, the method **UpdateNode** of **VcNode** has to be called so that the alteration of the data becomes visible.

Example Code

```
Dim tmpCal As VcCalendar
Dim tmpDate As Date
Set tmpCal = VcGantt1.CalendarCollection.Active
tmpDate = tmpCal.AddDuration(node.DataField(2), node.DataField(4))
node.DataField(3) = tmpDate
node.UpdateNode
```

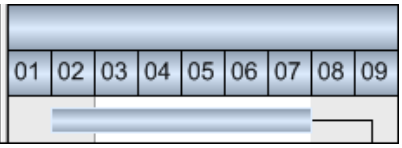
Start and end dates of activities that were created or modified by mouse interactions are automatically placed in active times.

ID	Name	Start	End	Duration
1	Node 1	03.05.10	08.05.10	5



In contrast, dates that were set by the API or by editing dialogs can be placed in non-working times.

ID	Name	Start	End	Duration
1	Node 1	03.05.10	08.05.10	5



Dates that were generated by calculation are always situated in working times. In order to ensure dates set by the API to be placed in working times, the start date needs to be calculated from the end date and from the duration of the activity.

Example Code

```
tmpDate = tmpCal.AddDuration(node.DataField(3), _
                        (-1) * node.DataField(4))
node.DataField(2) = tmpDate
```

For keeping the data consistent, missing or negative durations should be treated as improper and be reset to 0. If the start date is missing, the end date cannot be calculated. The required code was summarized to a separate method named **SetNodeEndDate(...)**.

Example Code

```
Private Sub SetNodeEndDate(ByVal node As VcNode)
   'Avoid empty or negative duration
   If node.DataField(4) = "" Or node.DataField(4) < 0 Then
       node.DataField(4) = "0"
   End If
   'Start date empty then end date should also be empty
   If node.DataField(2) = "31.12.1899 00:00:00" Then
       node.DataField(3) = ""
   Else
     'Precondition is property page nodes
     '"Assign calendar to nodes" must be true
     Dim tmpCal As VcCalendar
     Dim tmpDate As Date
     Set tmpCal = VcGantt1.CalendarCollection.Active
      tmpDate = tmpCal.AddDuration(node.DataField(2), _
          node.DataField(4))
     node.DataField(3) = tmpDate
     'Start date only in active times
     tmpDate = tmpCal.AddDuration(node.DataField(3), _
                           (-1) * node.DataField(4))
     node.DataField(2) = tmpDate
     node.UpdateNode
   End If
End Sub
```

The calculation of dates is required:

1. After activities were loaded
2. After dates or durations were modified by a data editing dialog or by an in-place editor
3. After activity values were modified by the API

After modifications by mouse interactions however, a calculation does not have to be initiated, since then an internal calculation will be carried out automatically.

A computation loop which includes all nodes can be set up by the property **NodeCollection** of the **VcGantt** object. Its code will be added to the end of the event **Form1_Load(...)**.

Example Code

```
'Calculate end date for all nodes
Dim node As VcNode
For Each node In VcGantt1.NodeCollection
   SetNodeEndDate node
Next
```

Alterations of data caused by the user can be caught via the event **OnNodeModifyComplete**. The method call carries out the calculation of the end date.

Example Code

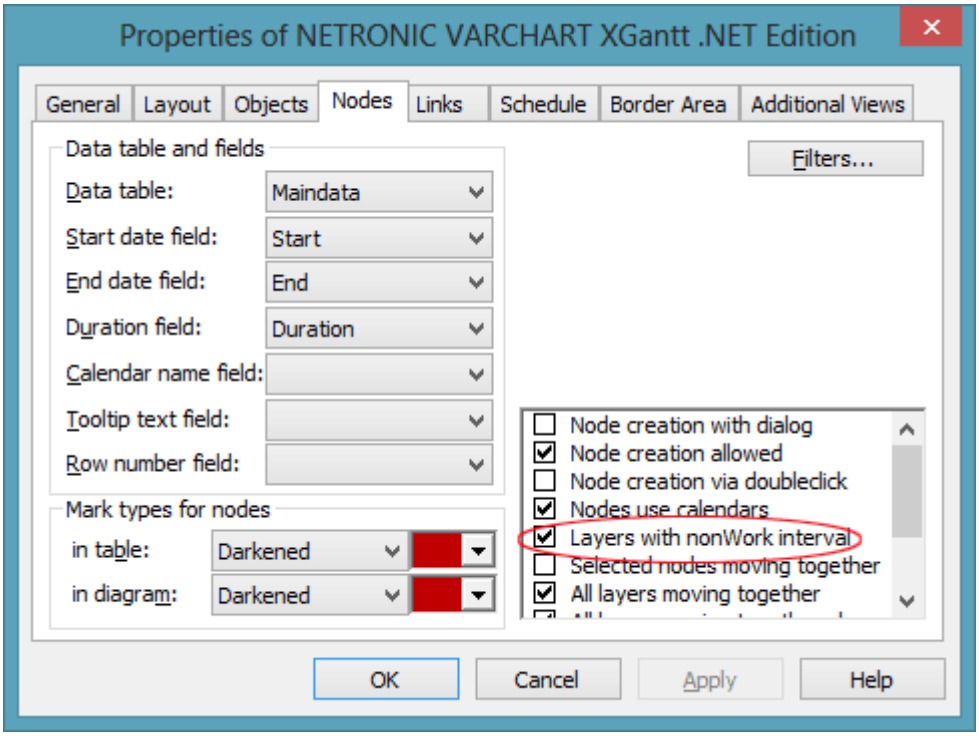
```
Private Sub VcGantt1_OnNodeModifyComplete _
    (ByVal node As VcGanttLib.VcNode, _
    ByVal isLastNodeInSeries As Boolean)
    SetNodeEndDate node
End Sub
```

If data have been altered via API, a **SetNodeEndDate(...)** has to be called.

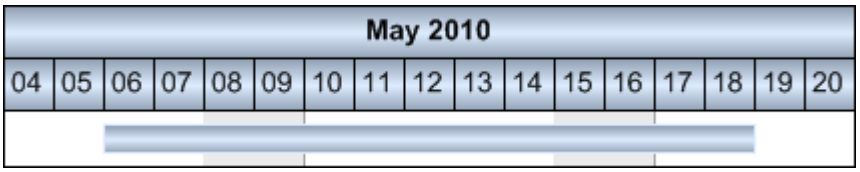
2.5 Marking non Working Intervals in Activities

Replacements of sections of activities by non working intervals can be visiualized by the option **Show non work interval**. The option is only effective if the activities depend on a calendar. This can be achieved by ticking **Assign calendar to nodes**.

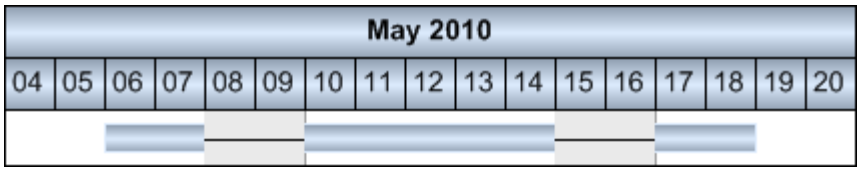
The settings can be made at runtime or at design time. You can find the options on the **Nodes** property page at the bottom right.



At runtime the setting can be made by the property **ShowNonWorkInterval** of the object VcGantt.



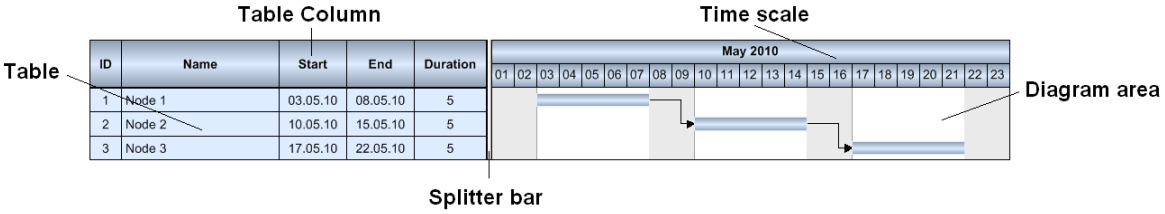
ShowNonWorkInterval = false



ShowNonWorkInterval = true

2.6 Interactions in the Table and Diagram Area

This subchapter and the one following will give you a general idea of interactions in the Gantt diagram. For more detailed information see chapter 5, **User Interface**.



> Modifying the left table/diagram width ratio

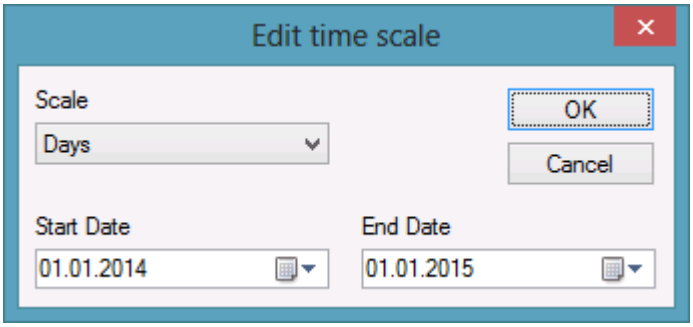
You can modify the sizes of the table and the diagram section of a Gantt chart by moving the gray vertical separating line (sash or splitter bar) towards the left or right. On the **Layout** property page the ratio can be pre-set in the field **Table/diagram width ratio**.

> Modifying the table column

By dragging the vertical separation line on the right of a table caption you can change the width of a table column. You can automatically adjust the column width to the length of its contents by double-clicking on the separation line. The automatical adjustment can be switched on or off on the **General** property page by ticking the option **Allow table column width optimization** in the list box on the right.

> Defining the start and end date of the time scale

By a double-click on the time scale you can pop up the **Edit Timescale** dialog box. It lets you edit the start and end dates of the time scale. This option may be activated or blocked on the **General** property page by the option **Show time scale dialog** in the list box on the right.



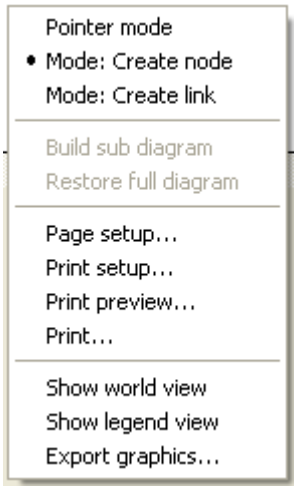
> Scaling the Time Scale

By dragging to the left or to the right in the time scale section you may enlarge or reduce the width of the unit of the time scale. This feature can be activated or or deactivated on the **General** property page by the option **Allow Time scale re-scaling** in the list box on the right.

2.7 Interactions with Activities

> Create new activity

Before creating a new activity you need to change to the mode **Create node**, which you can do by using the context menu of the diagram area (right mouse button).



After having selected the menu item the mouse pointer will take on the shape of small crosshairs. Now an activity can be drawn with the left mouse button pressed in the desired place of the diagram area. After finishing it is useful to return to the **Pointer mode** of the context menu.

In the **Create Node** mode the program may intervene by the event **OnNode-CreateCompleteEx()**. This is useful if you wish to pre-set data values.

Example Code

```
Private Sub VcGantt1_OnNodeCreateCompleteEx _
    (ByVal node As VcGanttLib.VcNode,_
    ByVal creationType As VcGanttLib.CreationTypeEnum,_
    ByVal isLastNodeInSeries As Boolean)

    node.DataField(1) = "Node " + node.DataField(0)
    node.UpdateNode
End Sub
```

The code displayed above will modify the contents of the data field **Name**; it will attach the contents of the field **ID** to the term **Node**.

> Modifying the duration of an activity

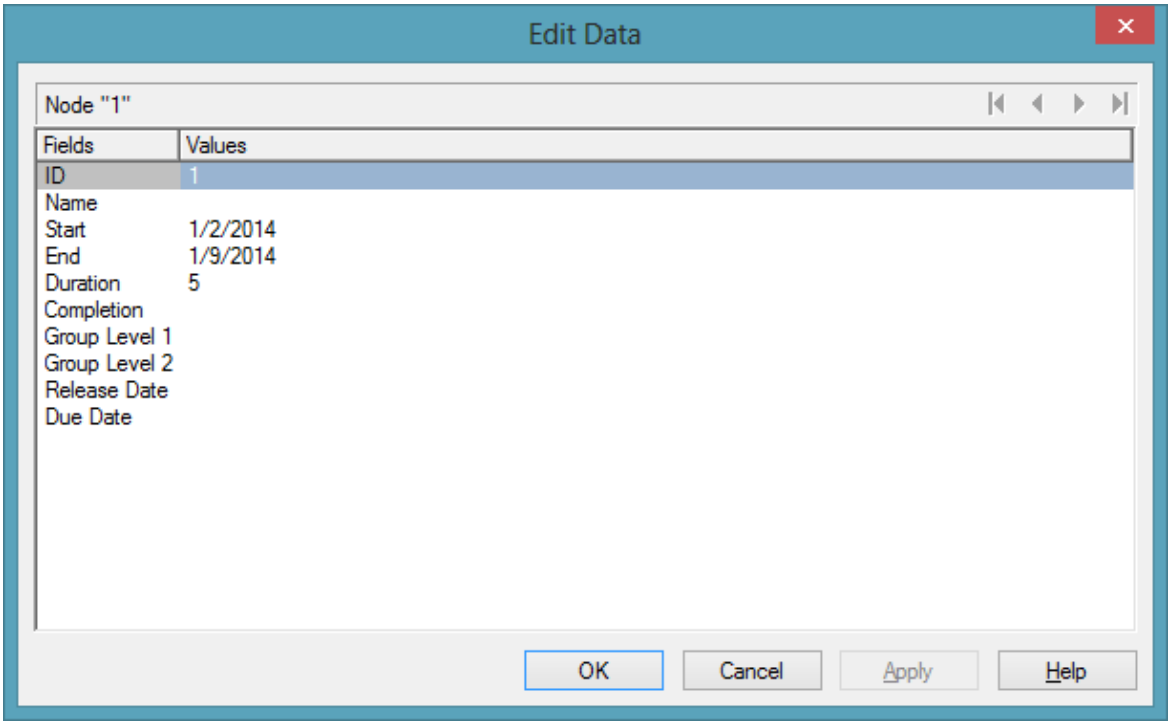
Return to the **Pointer Mode** if necessary and move the mouse pointer close to the inner right or left delimiter of the activity. The pointer will transform to a vertical line and horizontal arrow. By dragging the delimiter to the left or to the right, you can extend or reduce the size of the activity.

> Moving an activity

Return to **Pointer mode** if necessary and move the mouse pointer onto the center region of the activity. The pointer will take on the shape of a small square and four arrows. You can now move the activity to the desired position by dragging the mouse. If you want to move more than one activity simultaneously please activate the option **Move all selected nodes** on the property page **Nodes** in the list box on the right.

> Editing the data of an activity

By double clicking on an activity or on the corresponding table line, the dialog box **Edit data** will pop up.



> Deleting an activity

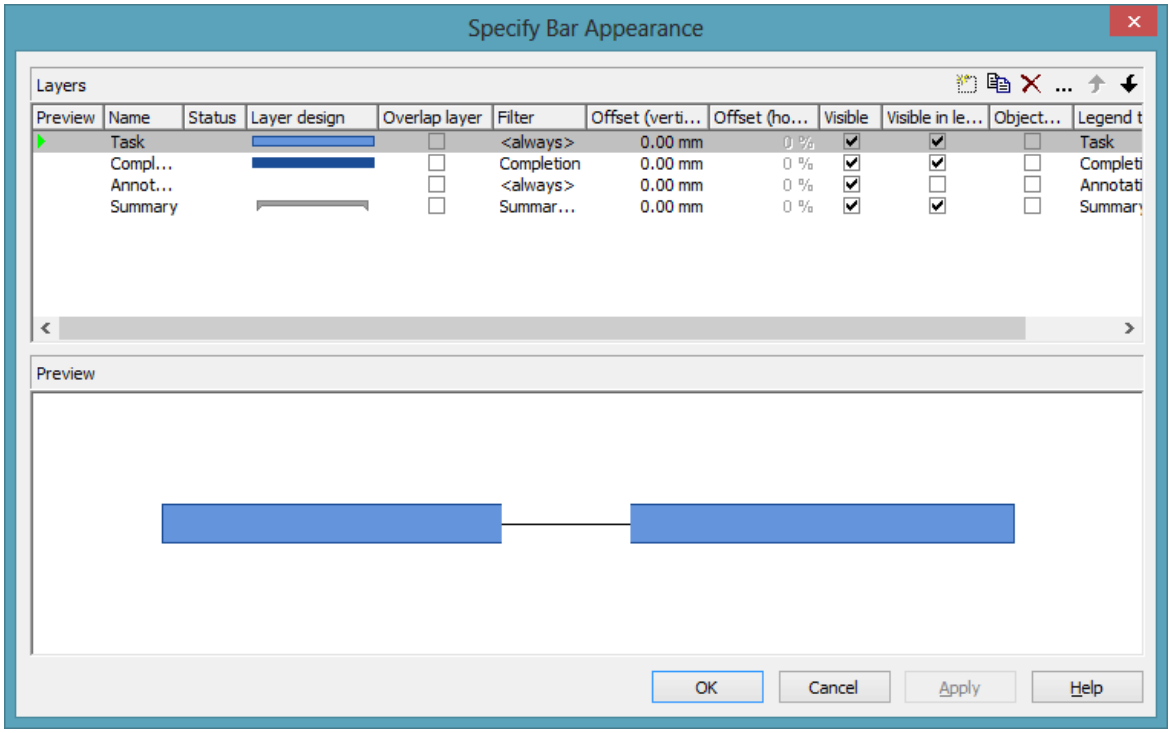
Marked activities can be deleted by pressing the **Del** key or by selecting **Delete nodes** in the context menu of the activity. You can mark an activity by clicking on it or on the corresponding table row. By pressing the **Ctrl** key you can mark more than one activity.

2.8 Using Layers

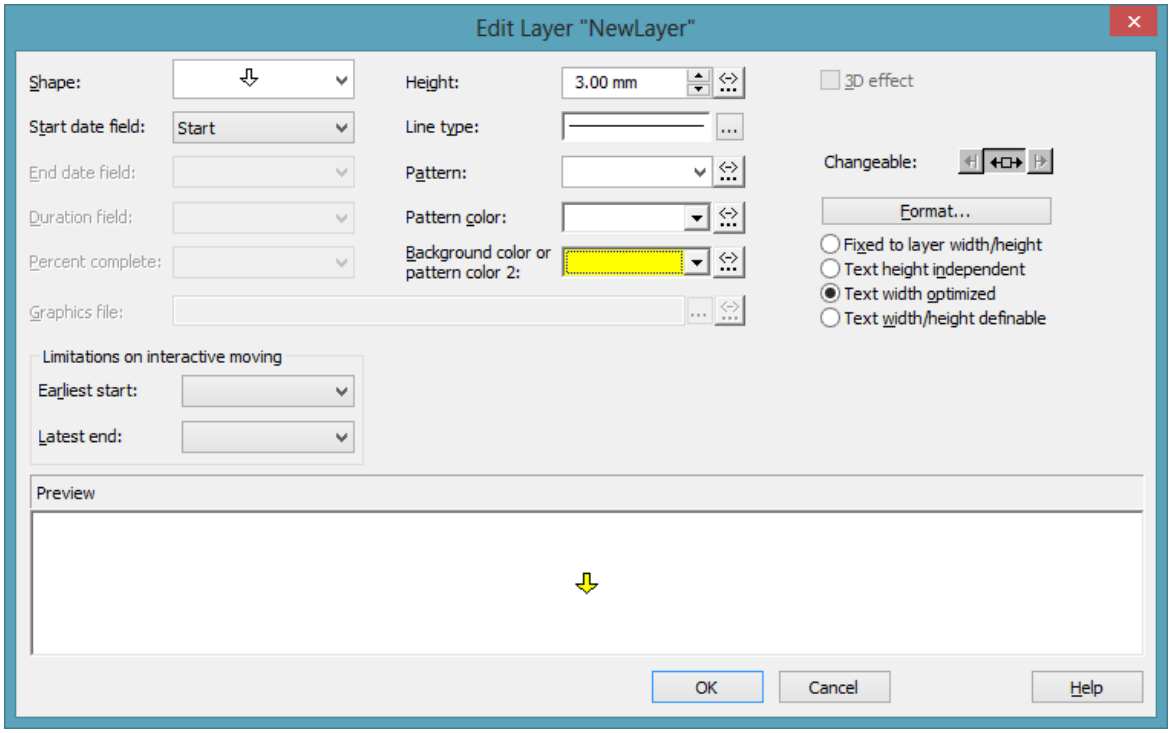
A layer is the graphical representation of a pair of dates. The same pair of dates can be displayed by different layers. They can be superimposed graphically.

For our sample, we are now going to create a second, different looking layer.

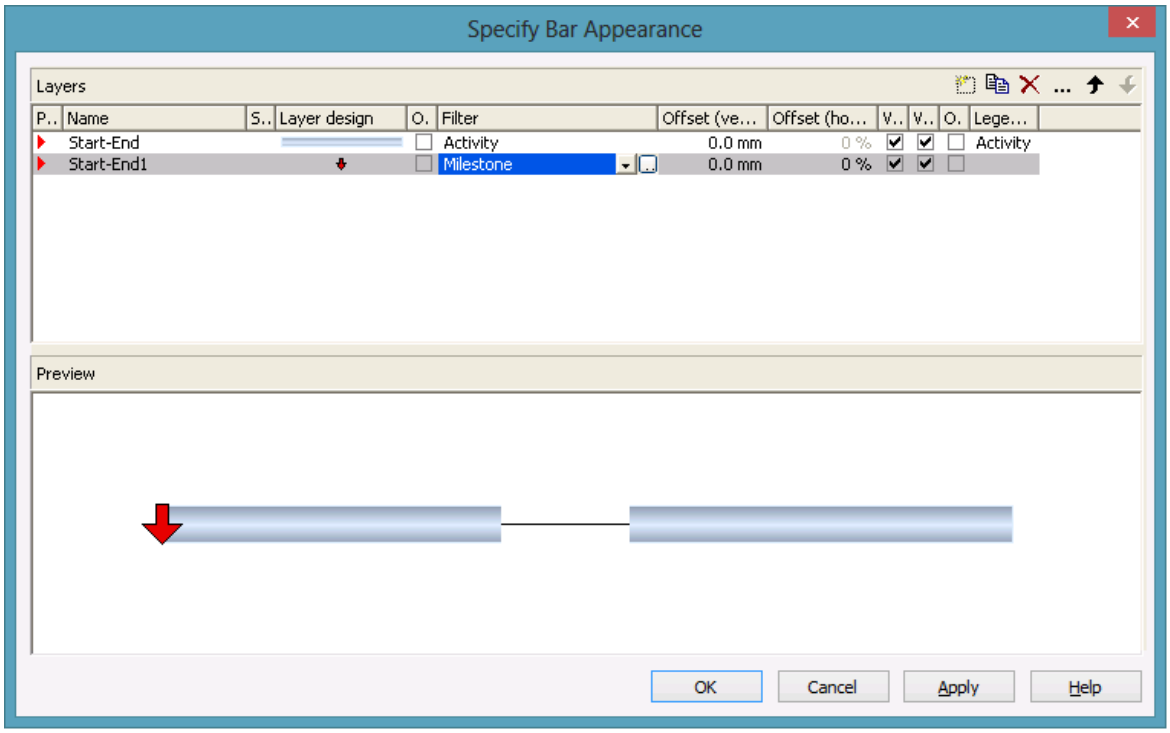
1. On the **Objects** property page select **Layers...**. The dialog **Specify Bar Appearance** will pop up. The layer named **Task** was already defined.



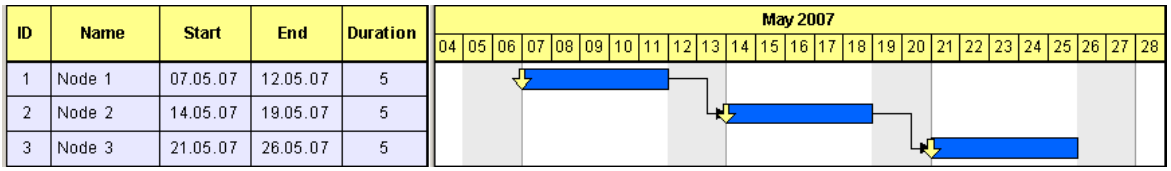
2. Copy the definition of the layer **Task** by clicking on the **Copy layer** button .
3. Change the name of **NewLayer** to **Start-End** and open the **Edit Layer** dialog by clicking on ... .
4. Modify the **Background color** to yellow and the **Shape** to arrowhead downward.



5. By clicking on OK, you will return to the dialog **Specify Bar Appearance**.
6. The superimposition of the layers can be made visible in the preview if you click in the lines of the column **Preview** where a red triangle indicates the preview of the corresponding layer in the lower half of the window.



7. In our programming sample, the modification of the definition shows the below result:



ID	Name	Start	End	Duration
1	Node 1	07.05.07	12.05.07	5
2	Node 2	14.05.07	19.05.07	5
3	Node 3	21.05.07	26.05.07	5

2.9 Using Filters

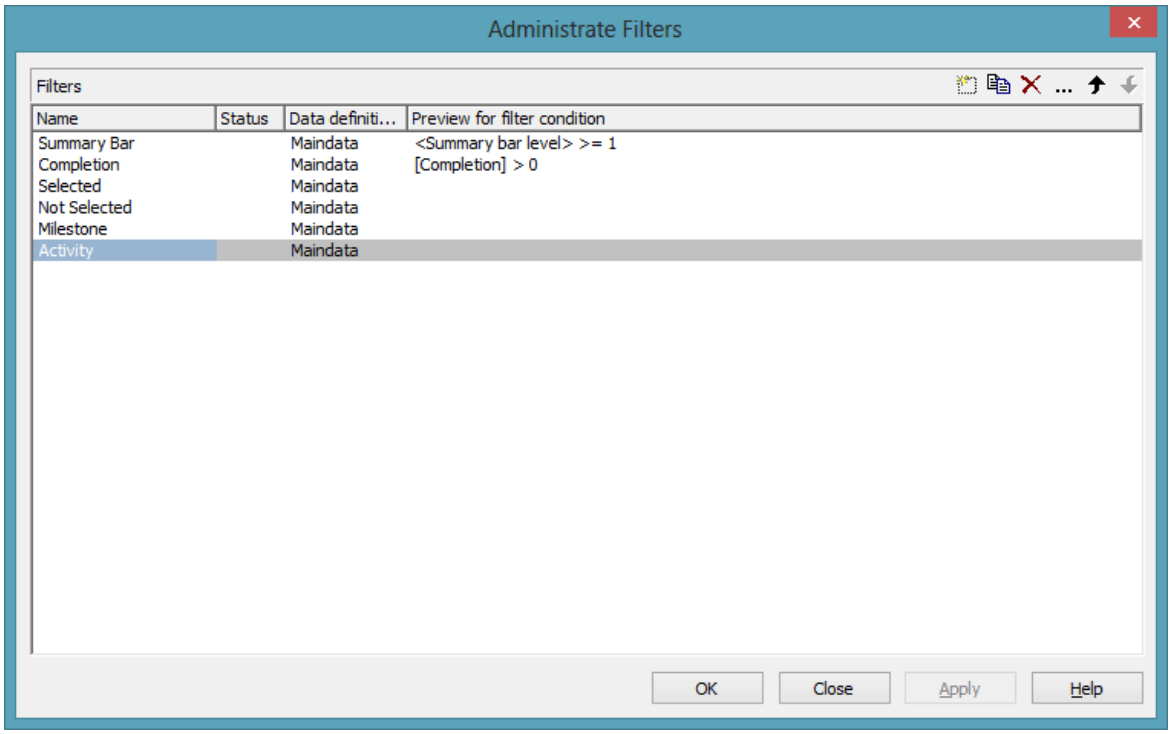
Next, we would like to have the yellow arrow appear only in case of a milestone, i.e. if the duration of an activity equals 0.

This problem can be solved easily by using filters. A filter consists of a series of linked conditions which result in a logical **Yes** or **No** statement.

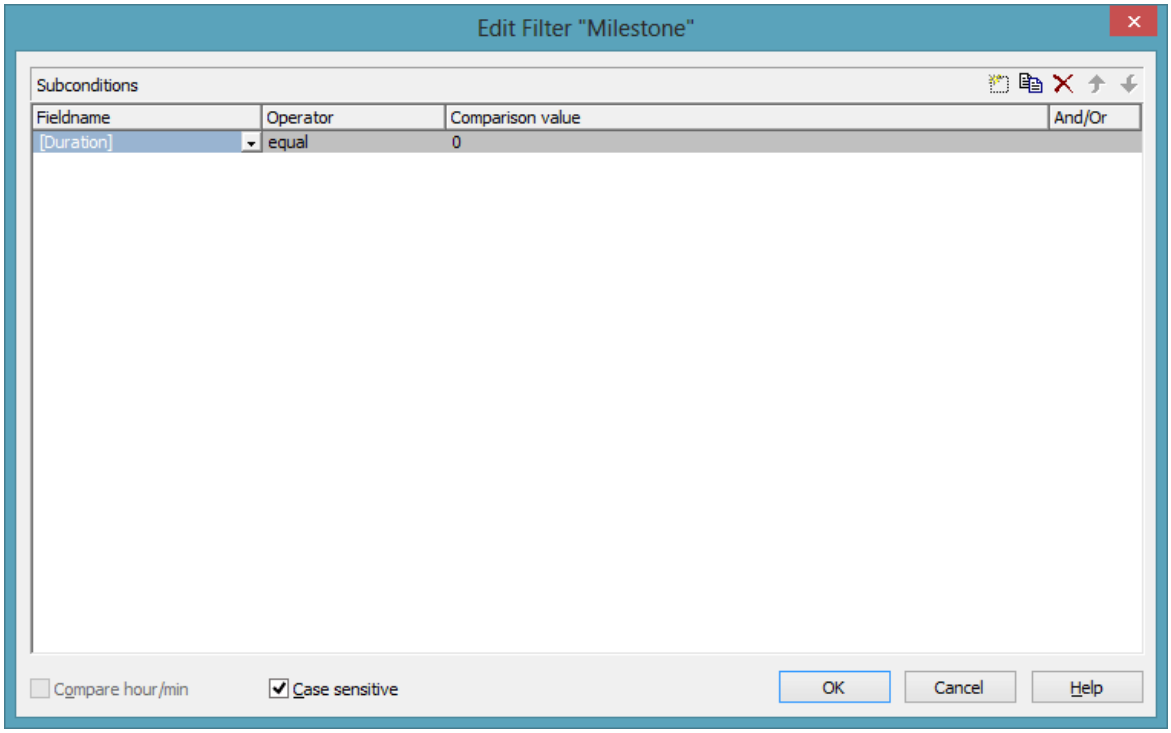
Layers are always linked to filters. A layer only becomes visible if the evaluation of the filter conditions is positive. The built-in filter <always>, which by default is assigned to a layer, always produces a positive answer. For our sample, we need two filters with one condition each:

- The yellow arrow shall appear if the duration of the activity equals 0
- The blue bar shall appear if the duration is larger than 0

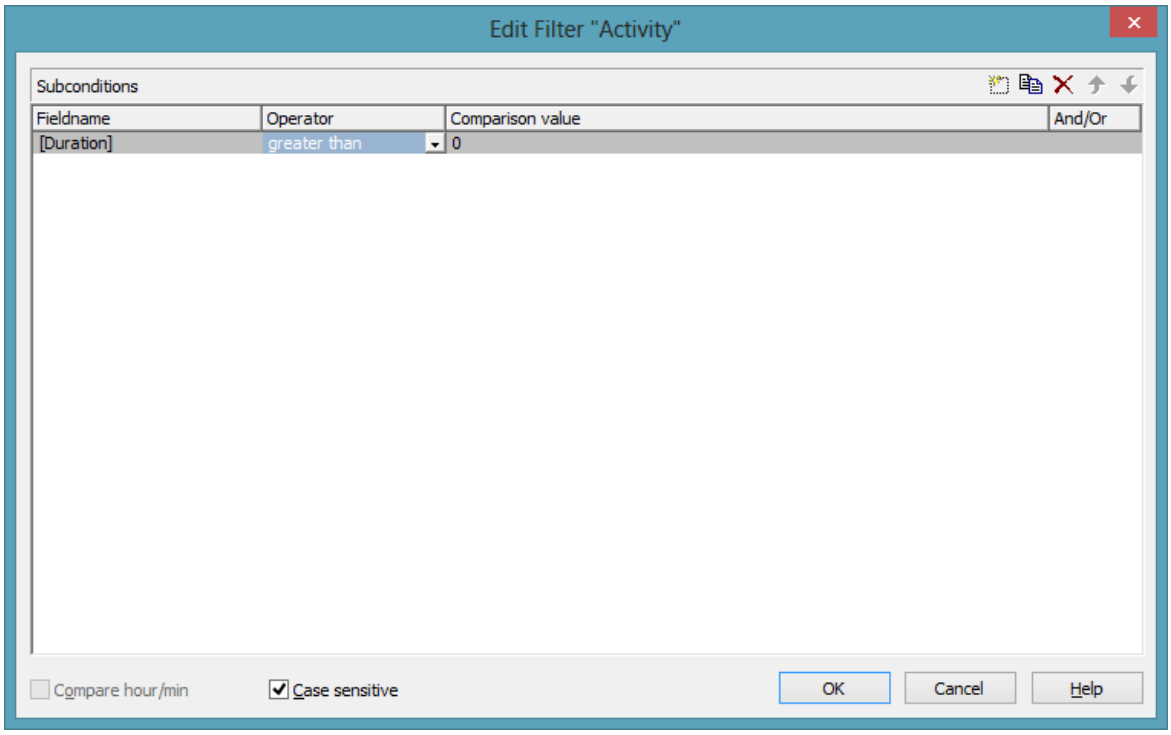
1. On the property pages **Objects** please click on the button **Filters...**. The dialog **Administer Filters** will pop up.
2. Create two new filters by clicking on the button .



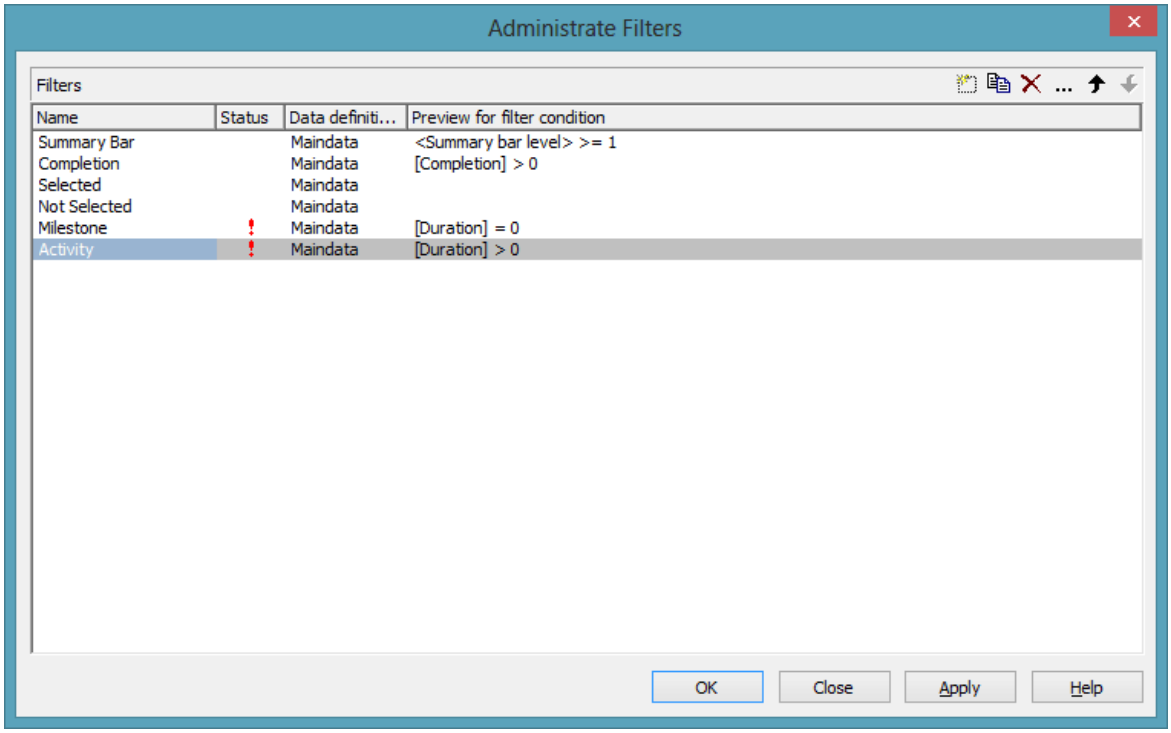
3. Select the filter "Milestone" and open the dialog **Edit Filter** by clicking on ... .
4. Select "Duration" as **Fieldname**, as **Operator** "equal" and as **Comparison value** 0.



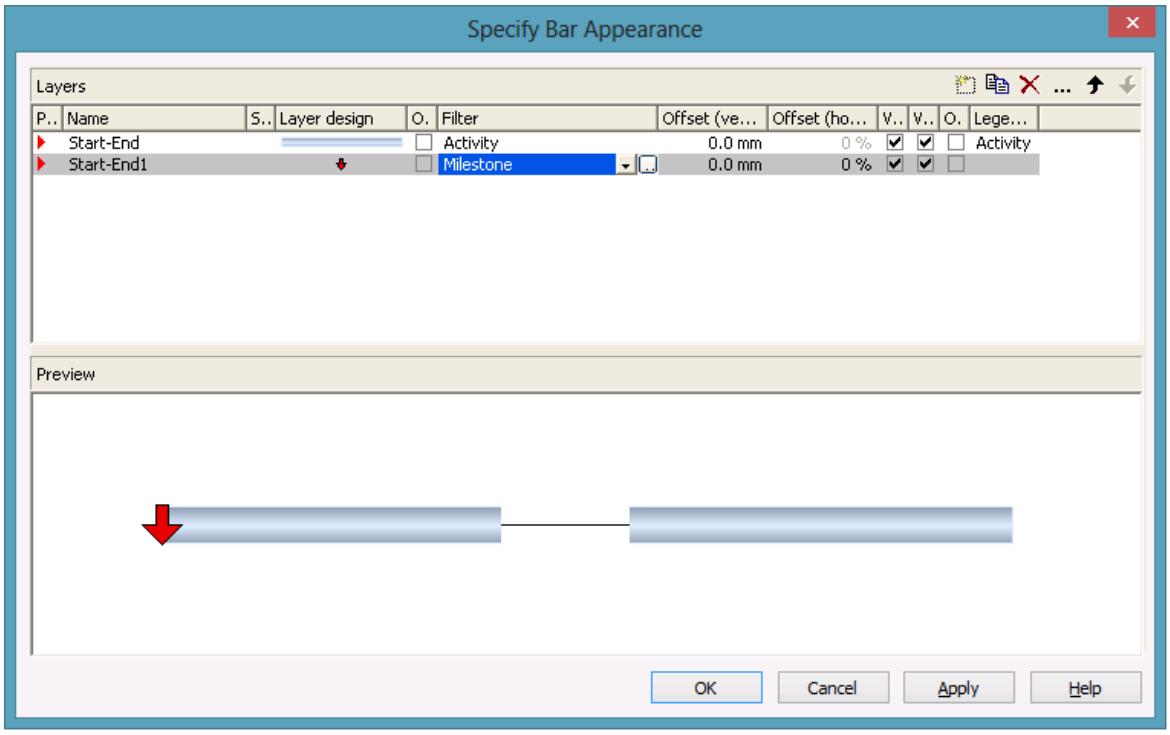
5. Leave the dialog by clicking on **OK**.
6. Select "Activity" and by clicking ··· open the **Edit Filter** dialog again.
7. Select "Duration" as **Fieldname**, for the **Operator** "greater than" and for the **Comparison value** 0.



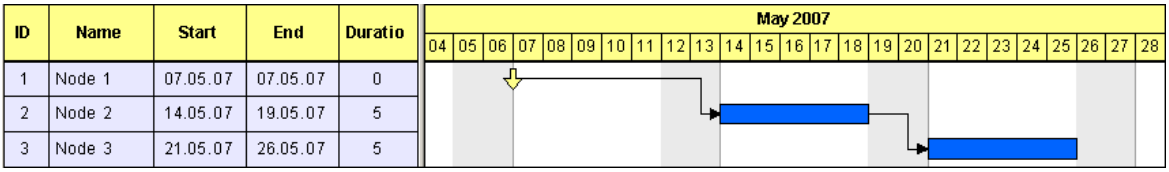
8. Leave the dialog by clicking on **OK**.



9. Click on **OK** again to return to the property pages.
10. To put the filters into operation they need to be assigned to the layers. For this, please click on the button **Layers** to open the dialog **Specify Bar Appearance**.



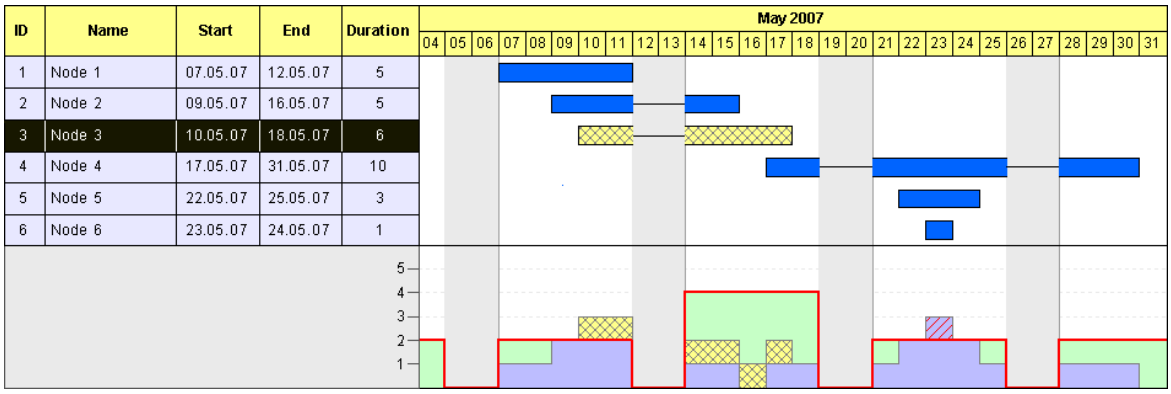
11. If you run the program now and if the duration of the first activity is set to 0, the below result will be produced:



The starter sample can be found in the directory UserGuideSamples\VB.NET\XGantt_Tutorial01_App.

2.10 Creating Histograms

In this sample you will get acquainted with histograms. We will demonstrate how to define an availability curve, how a capacity curve can be created from activities and how to visualize the section of marked activites within in the work load.



In the above example, an activity occupies a resource by the quantity of a single unit. Where activities overlap, occupation units are added up to the total capacity occupied.

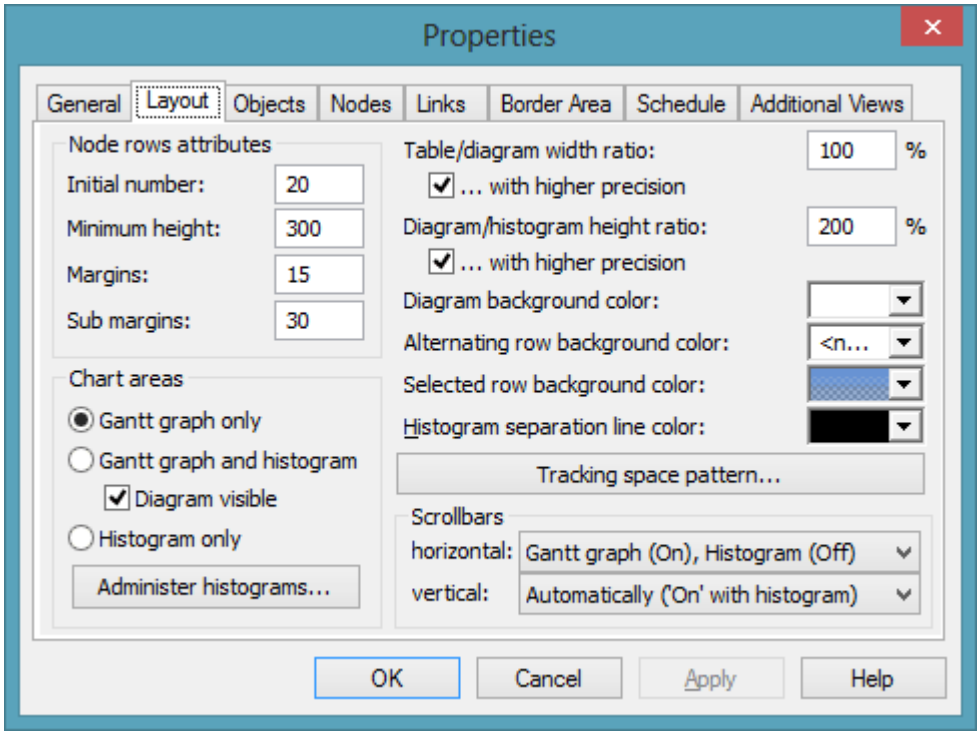
In the steps following, we will complete our previous sample by the features mentioned. To better illustrate the functions of the histogram, we have used different records and ommited links. The Form1_Load sample was modified as shown below:

Example Code

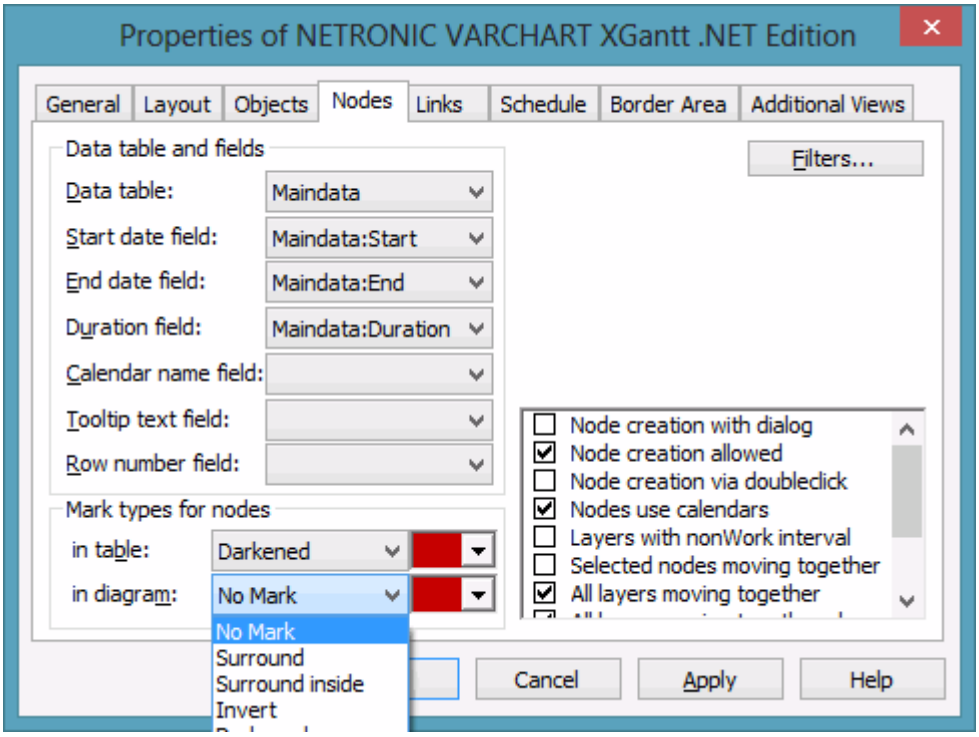
```
Private Sub Form_Load()
    VcGantt1.Width = ScaleWidth - VcGantt1.Left
    VcGantt1.Height = ScaleHeight - VcGantt1.Top

    VcGantt1.InsertNodeRecord ("1;Node 1;07.05.07;;5")
    VcGantt1.InsertNodeRecord ("2;Node 2;09.05.07;;5")
    VcGantt1.InsertNodeRecord ("3;Node 3;10.05.07;;6")
    VcGantt1.InsertNodeRecord ("4;Node 4;17.05.07;;10")
    VcGantt1.InsertNodeRecord ("5;Node 5;22.05.07;;3")
    VcGantt1.InsertNodeRecord ("6;Node 6;23.05.07;;1")
    VcGantt1.EndLoading

    VcGantt1.OptimizeTimeScaleStartEnd (3)

    'Calculating the end date of all nodes
    Dim node As VcNode
    VcGantt.SuspendUpdate True
    For Each node In VcGantt1.NodeCollection
        SetNodeEndDate node
    Next
    VcGantt.SuspendUpdate False
End Sub
```

Beside, we removed the filters ("Milestone", "Activity") and the additional layer ("Milestone") from the above sample. The complete program can be found in the directory **UserGuideSamples\VB6\XGantt_Tutorial02**.

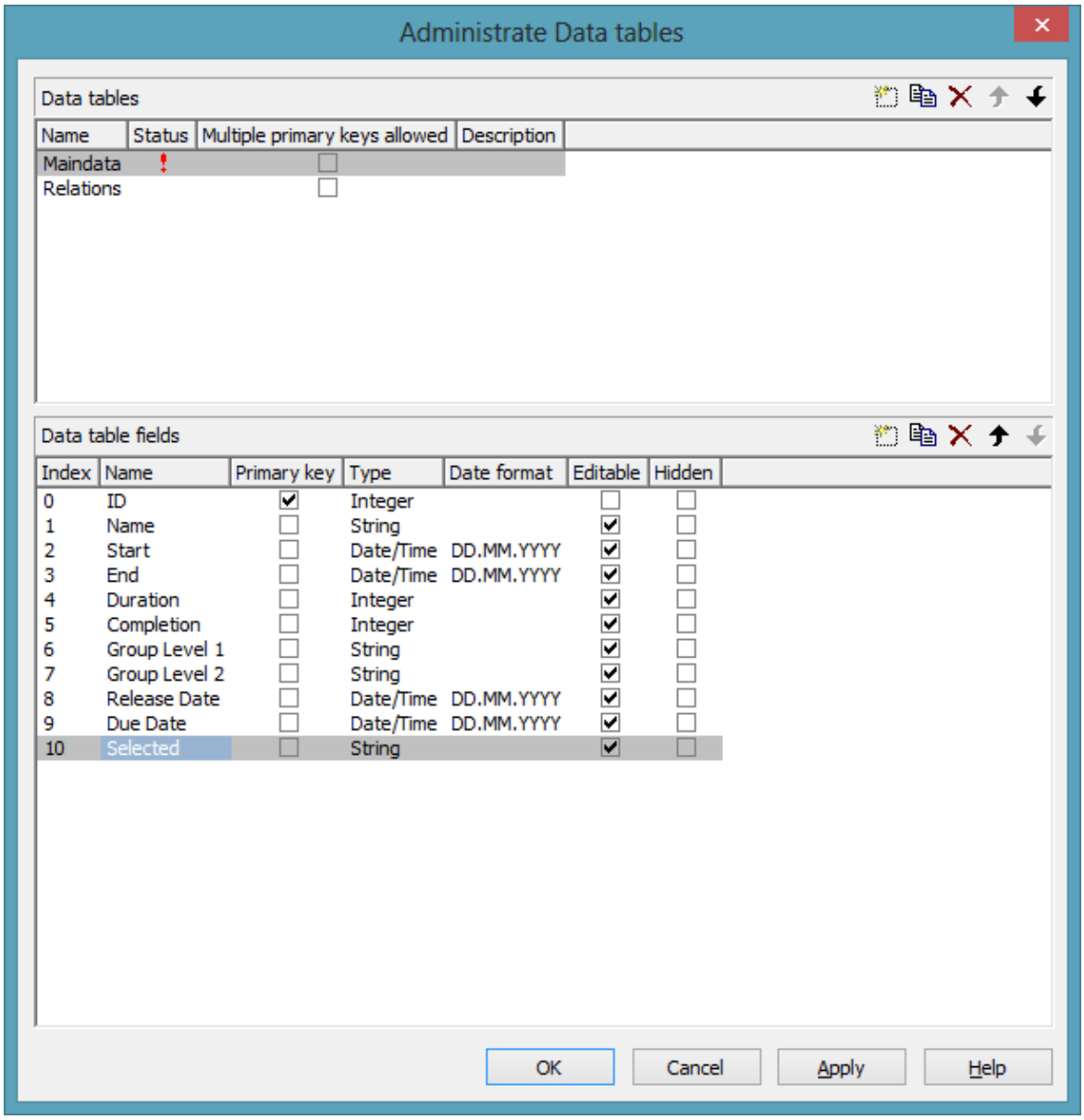
First, displaying histograms needs to be enabled on the property page **Layout** in the section **Chart areas** by selecting the radio button **Gantt graph and histogram**.



Marked nodes shall display a cross hatch pattern. Therefore, on the propert page **Nodes** in the section **Mark type for nodes** set the field **in diagram:** to **No Mark**.



One more data field will be needed later on in our sample, which we are going to create now. In the dialog box **Edit data table…** please create a field of the type **Integer** and name it **Selected**. The field will make the display of the activity depend on ist marking state.



The value of the field **Selected** needs to be updated each time the event **OnNodesMarkComplete** is triggered.

Example Code

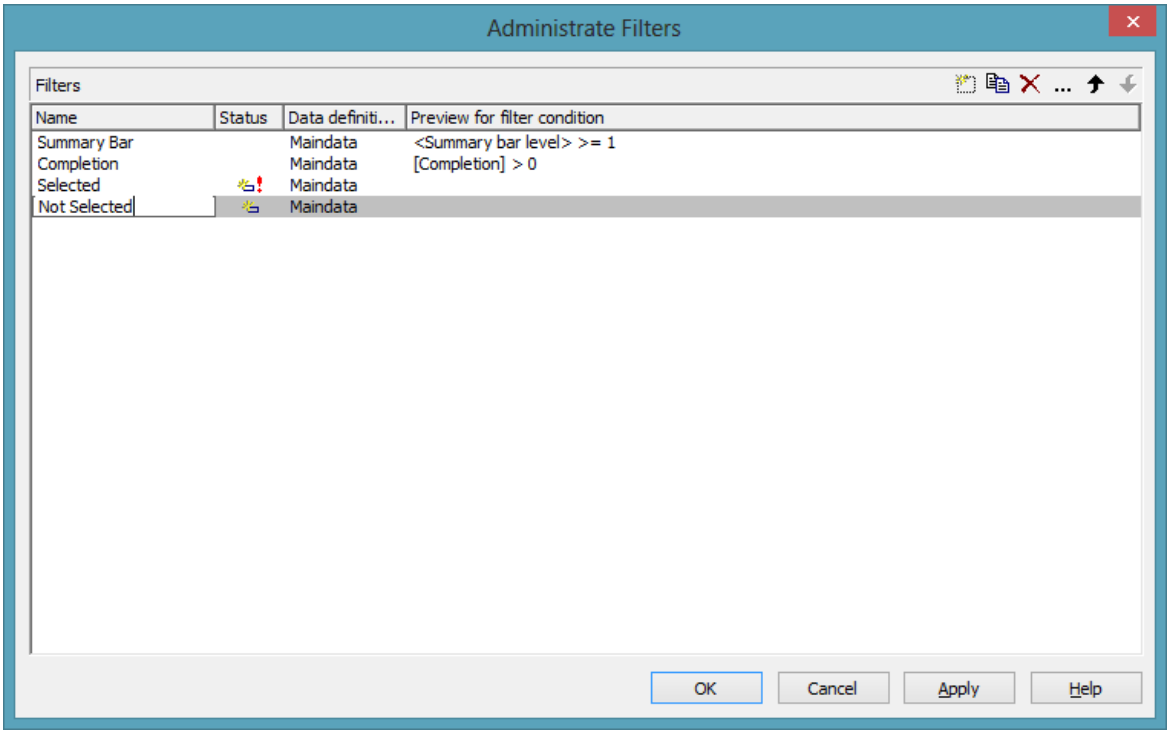
```
Private Sub VcGantt1_OnNodesMarkComplete()
    Dim node As VcNode
    For Each node In VcGantt1.NodeCollection
        If node.MarkNode = True Then
            node.DataField(5) = 1
        Else
            node.DataField(5) = 0
        End If
        node.UpdateNode
    Next

End Sub
```

In the event **OnNodeCreateCompleteEx** the below code will prevent that a newly created node is marked when appearing. Since all nodes previously selected will be unmarked when a new node is created, the contents of the field **Selected** has to be updated.

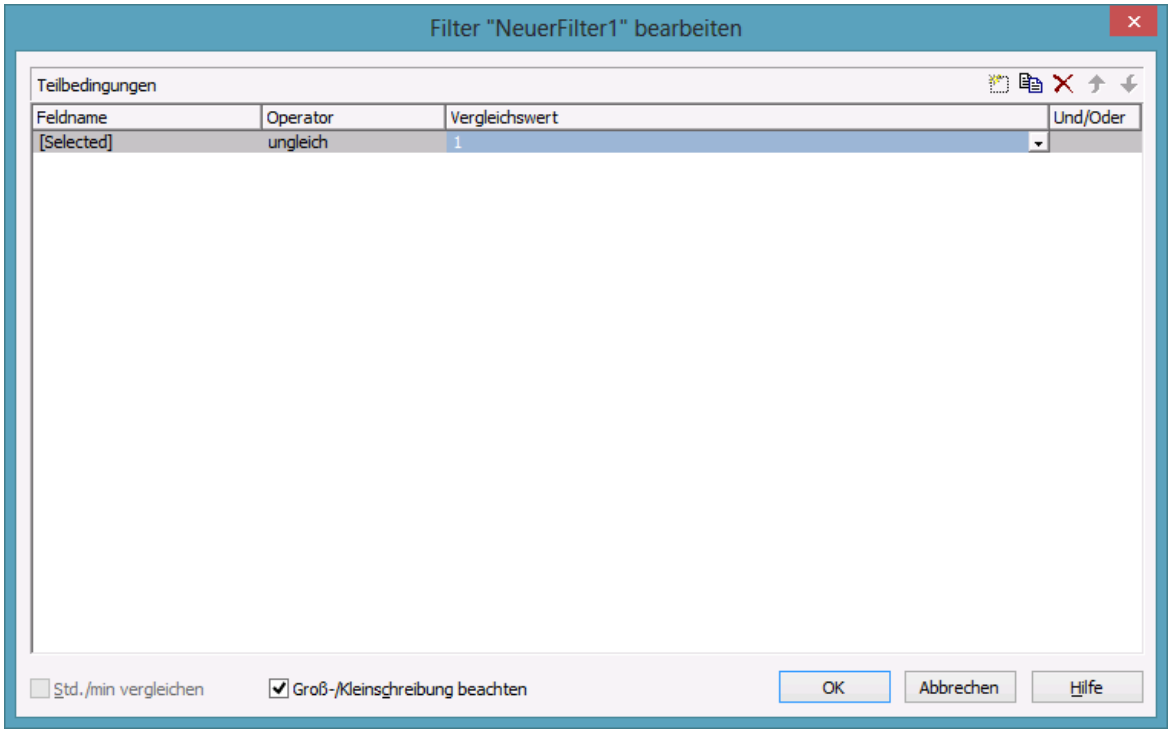
Example Code

```
Private Sub VcGantt1_OnNodeCreateCompleteEx _
    (ByVal node As VcGanttLib.VcNode,_
    ByVal creationType As VcGanttLib.CreationTypeEnum,_
    ByVal isLastNodeInSeries As Boolean)

    node.DataField(1) = "Node " + node.DataField(0)
    node.MarkNode = False
    node.UpdateNode

    Dim node As VcNode
    For Each node In VcGantt1.NodeCollection
        node.DataField(5) = 0
        node.UpdateNode
    Next
End Sub
```

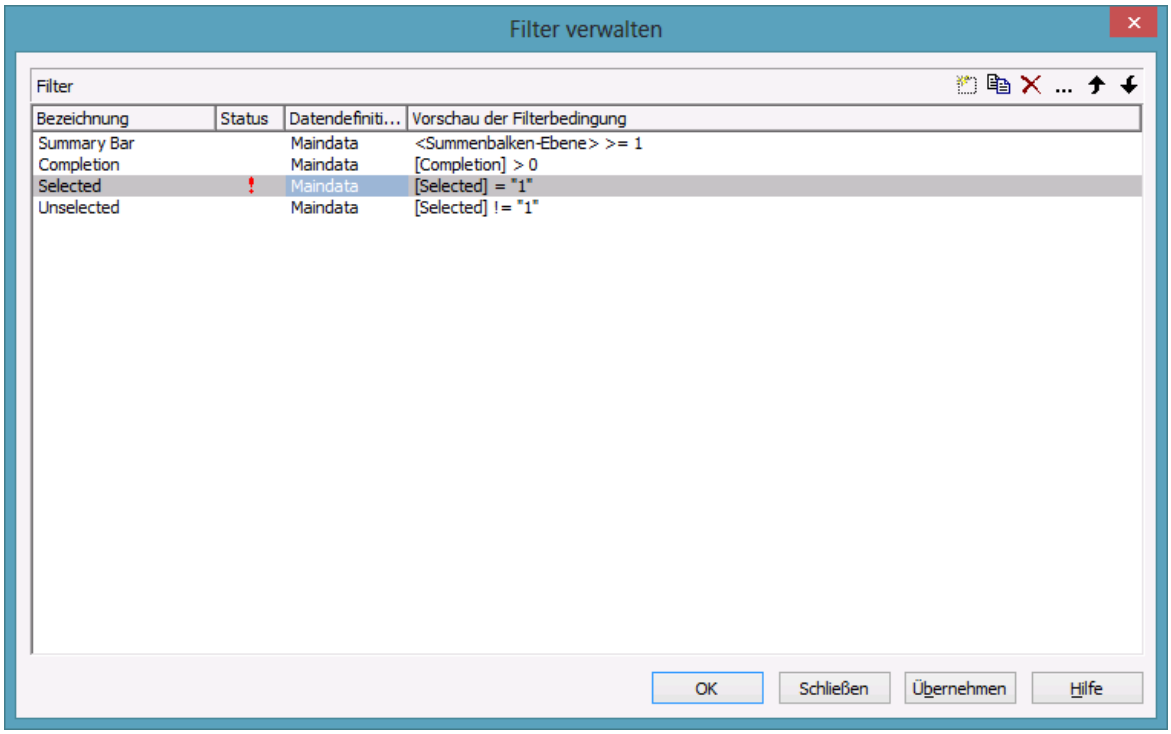
Next, we will define filters which differentiate between marked and unmarked activities. Please click on the button **Filter...** on the property page **Objects** to get to the dialog **Administer Filters**. Please create two new filters by clicking on  and name one of them **NotSelected** and the other one **Marked**.



To the filter **NotSelected**, please set the condition **Selected not equal 1**.

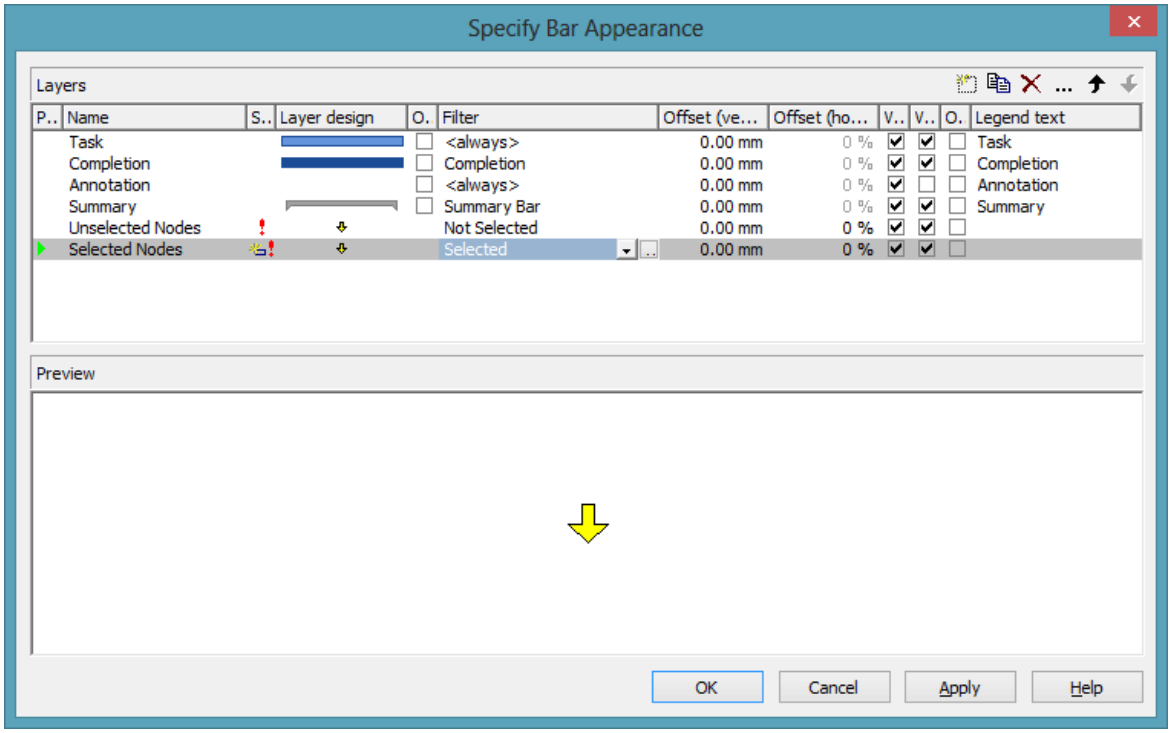


To the filter **Selected**, please set the condition **Selected equal 1**:



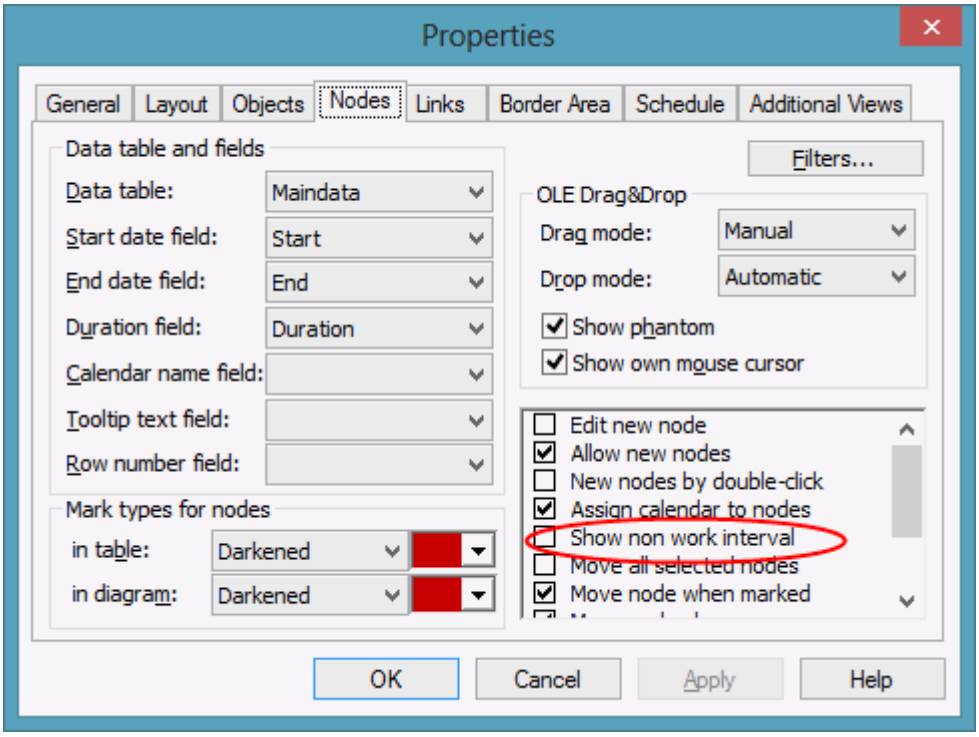
Now, the appearance of the activities shall to be linked to the filters. Please go to the dialog **Specify Bar Appearance** by clicking on the button **Layers**

on the property page **Objects**. Rename the layer **Start-End** into **Unmarked Nodes** and assign the filter **NotSelected** to it. Copy the layer by clicking on and name the copy **Marked Nodes**. Assign the filter **Selected** to the layer.

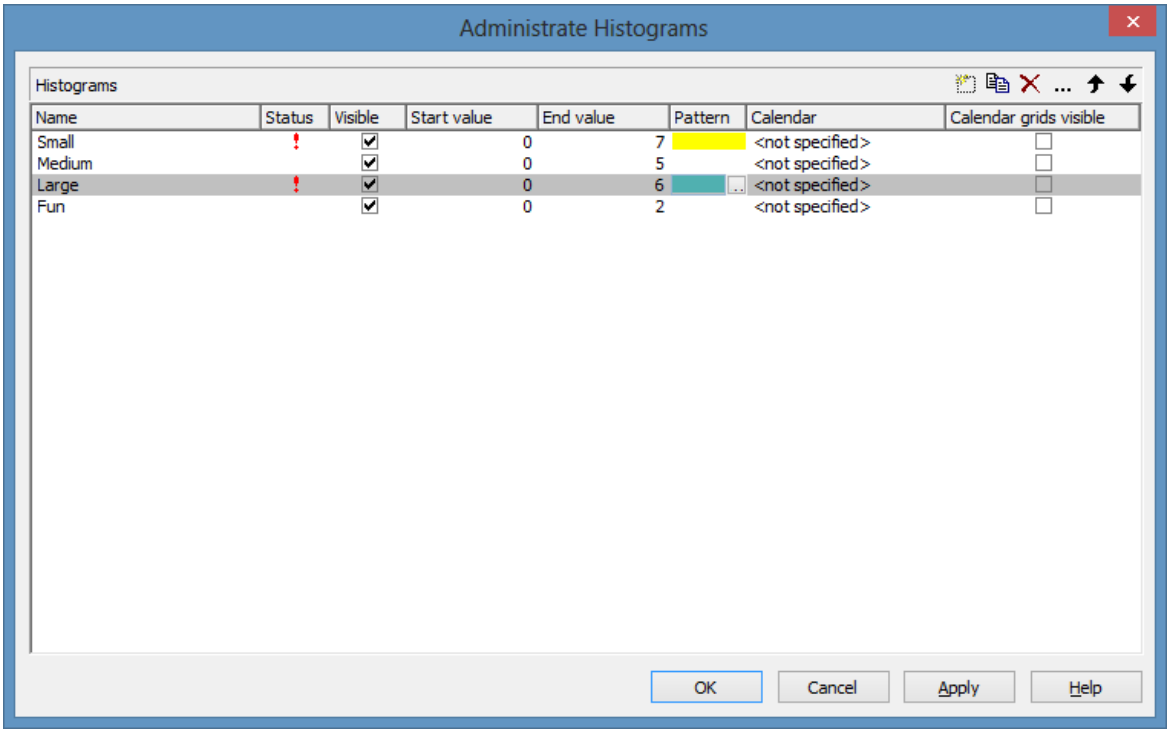


Both layers still look alike. Modify the design of the layer **SelectedNodes** in the dialog **Edit Layer** by selecting the pattern **cross hatch** and the background color **yellow**.

Note: On the property page **Nodes** the option **Show non-work interval** should be ticked to ensure that in non-work intervals (e.g. on weekends) a line instead of a bar will be displayed.



Next, we will define the curves in the histogram. You can get to the dialog **Administer histograms** on the property page **Layout** by clicking on the button **Administer histograms**



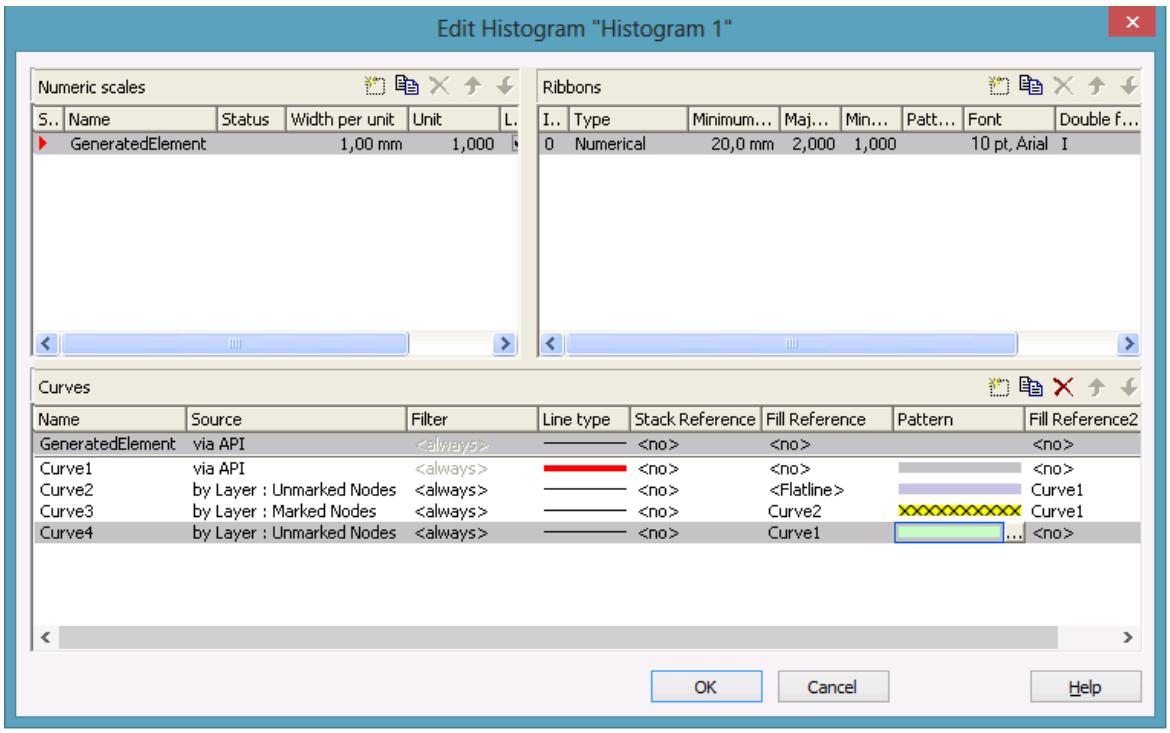
Several histograms may be present in a Gantt chart at the same time. Each of the histograms has a numeric scale of ist own and contains ist own curves.

We are now going to define a start and an end value to the numeric scale of the histogram. For this, in **Histogram 1** please set the end value to 6.

Click on the button **Edit histograms** ... in order to modify the pre-defined histogram.

Curve 1 is the "availability" curve that indicates the available capacity. It is marked by a red line. Curve 2 adds up the work load of marked nodes. Curve 3 adds up the work load of unmarked nodes. Curve 4 provides the green background complementary to the availability curve.

When opening the dialog, the first curve already exists. Please create three more curves and define their properties according to the illustration.

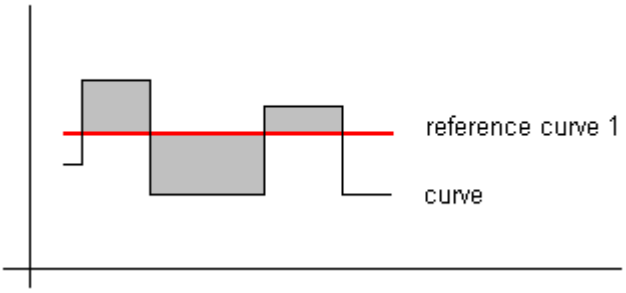


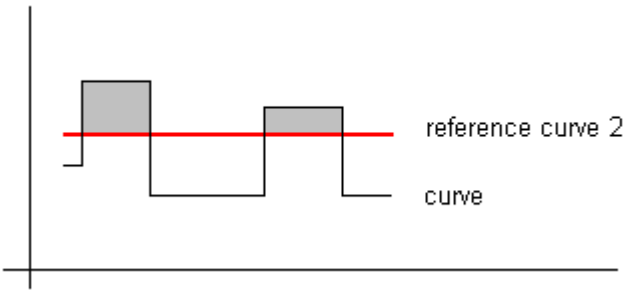
Curves can be stacked on one another. Stacking serves to add up the values of curves. A curve to be added needs a curve to which it is added, a **reference curve**. You can select the reference curve by the field **Stack reference**.

Curves in a histogram can form an area that may have a color and a pattern, for example a solid gray area or a green one hatched by red lines. If you wish to form an area and fill it with a color or a pattern, you need to set a **fill reference** to a curve.

Two different types of fill areas exist:

1. Areas that form above and below a curve
2. Areas that only form above a curve and therefore indicate a transgression





In the former case, the fill reference curve is to be specified by the field **Fill reference**; in the latter case, please use the field **Fill Reference2**. If you wish the x axis to limit the area, please select **Flatline**.

Finally, the programming code in the **Load** event needs to be modified to provide the values of the activities and of the capacity curve.

Example Code

```
Private Sub Form_Load()
    VcGantt1.Width = ScaleWidth - VcGantt1.Left
    VcGantt1.Height = ScaleHeight - VcGantt1.Top

    VcGantt1.InsertNodeRecord ("1;Node 1;07.05.09;;5")
    VcGantt1.InsertNodeRecord ("2;Node 2;09.05.09;;5")
    VcGantt1.InsertNodeRecord ("3;Node 3;10.05.09;;6")
    VcGantt1.InsertNodeRecord ("4;Node 4;17.05.09;;10")
    VcGantt1.InsertNodeRecord ("5;Node 5;22.05.09;;3")
    VcGantt1.InsertNodeRecord ("6;Node 6;23.05.09;;1")
    VcGantt1.EndLoading

    VcGantt1.OptimizeTimeScaleStartEnd (3)

    'Calculating the end dates of all nodes
    Dim node As VcNode
    For Each node In VcGantt1.NodeCollection
        SetNodeEndDate node
    Next

    Dim histogram As VcHistogram
    Dim curve As VcCurve

    Set histogram = VcGantt1.HistogramCollection.FirstHistogram
    Set curve = histogram.CurveCollection.CurveByName("Curve1")
    curve.PointsEquidistant = False
    curve.SetValues "01.05.09", "2"
```

```
    curve.SetValues "05.05.09", "0"
    curve.SetValues "07.05.09", "2"
    curve.SetValues "12.05.09", "0"
    curve.SetValues "14.05.09", "4"
    curve.SetValues "19.05.09", "0"
    curve.SetValues "21.05.09", "2"
    curve.SetValues "26.05.09", "0"
    curve.SetValues "28.05.09", "2"
End Sub
```

Run the program and click on an activity. In the histogram, you can recognize immediately by the hatching pattern on a yellow background, what section the acitivity occupies in the total resource occupation.

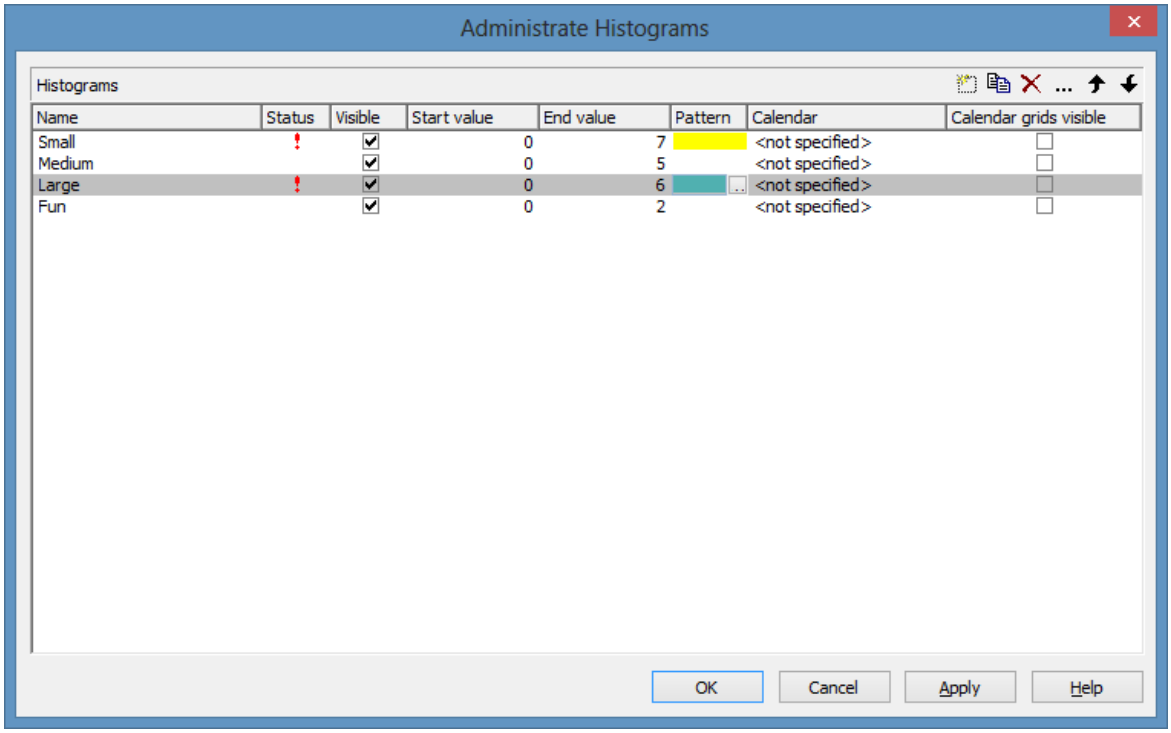
When moving activities, the degree of utilization will change and you will recognize capacity overloads and shortfalls deriving from your interaction.

Calendar Grids in Histograms

You can assign one ore more calendar grids to a histogram, so that different calendar grids in the Gantt graph can also become visible in the histogram.

To have an own calendar grid assigned to a histogram, three conditions have to be fulfilled:

1. A calendar has to be assigned to the histogram
2. The calendar grid has to be switched on
3. An appearance has to be defined that enables the display of the calendar grid



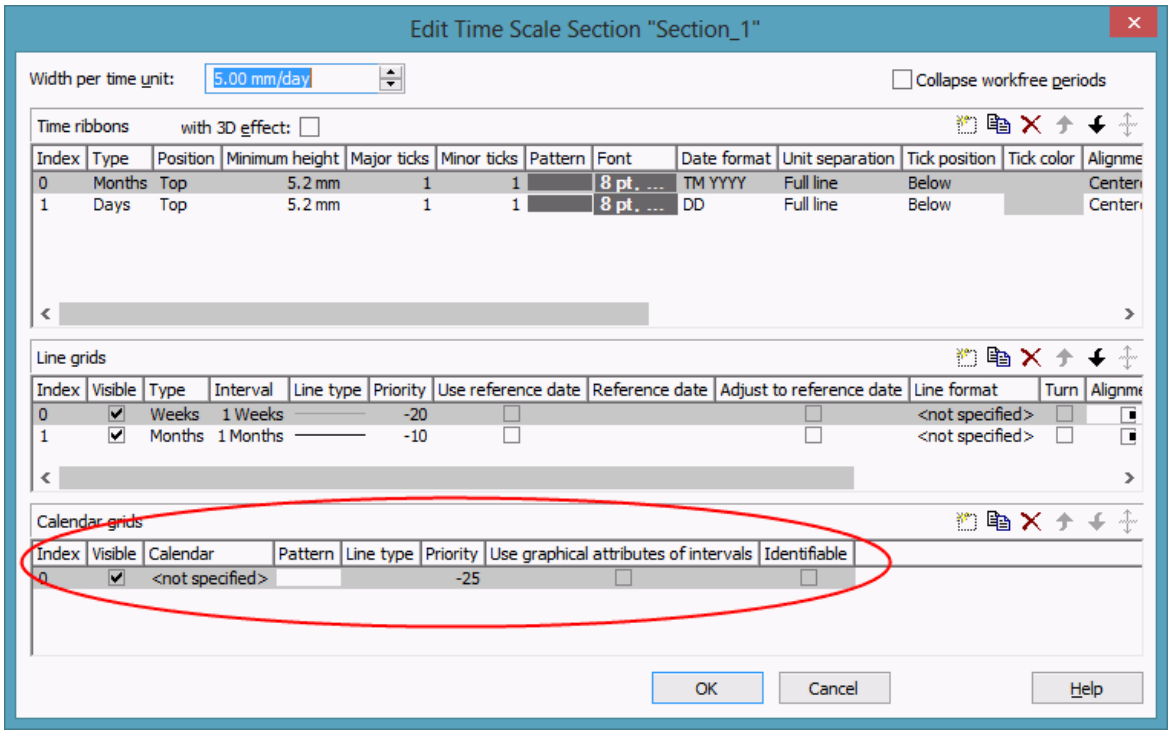
Calendar assigned, calendar grid switched on

The corresponding API calls are:

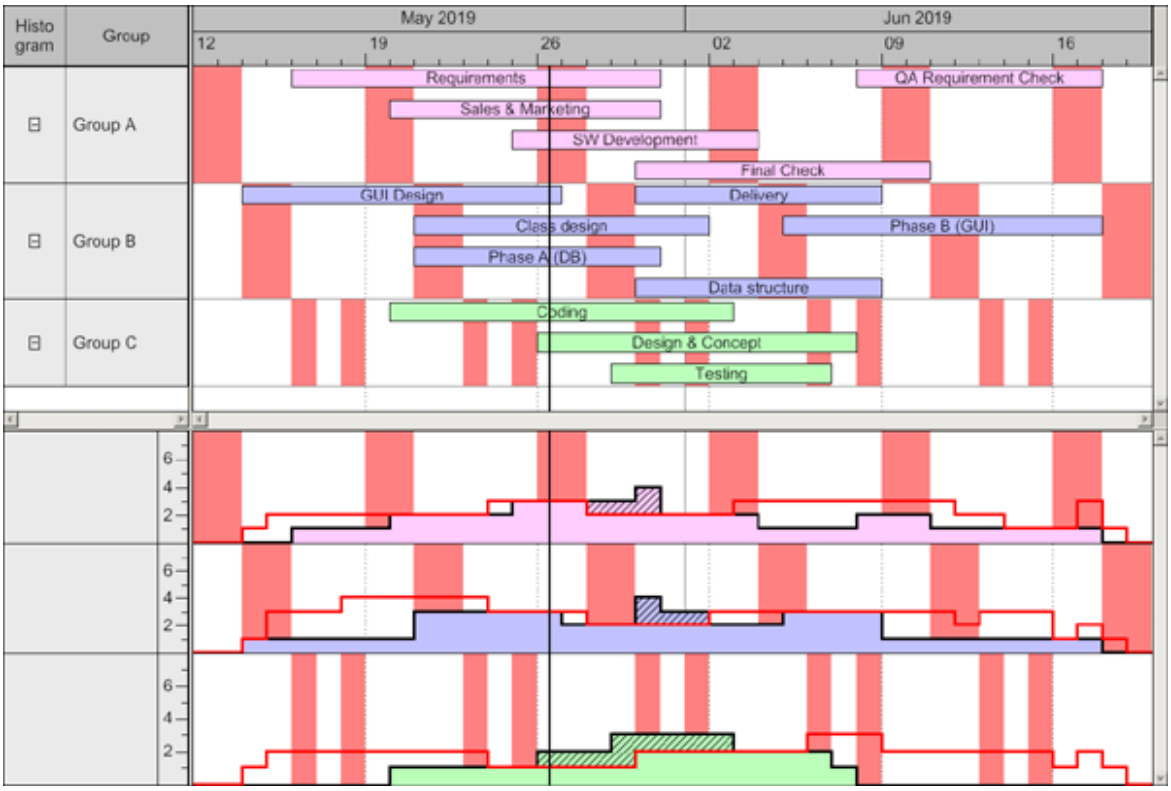
Example Code

```
// assigning the calendar to the histogram (by the calendar name)
histogram.calendarName = group.DataField(14)
// switching the calendar grid on
histogram.ShowCalendarGrids = True
// setting the histogram visible
histogram.Visible = True
```

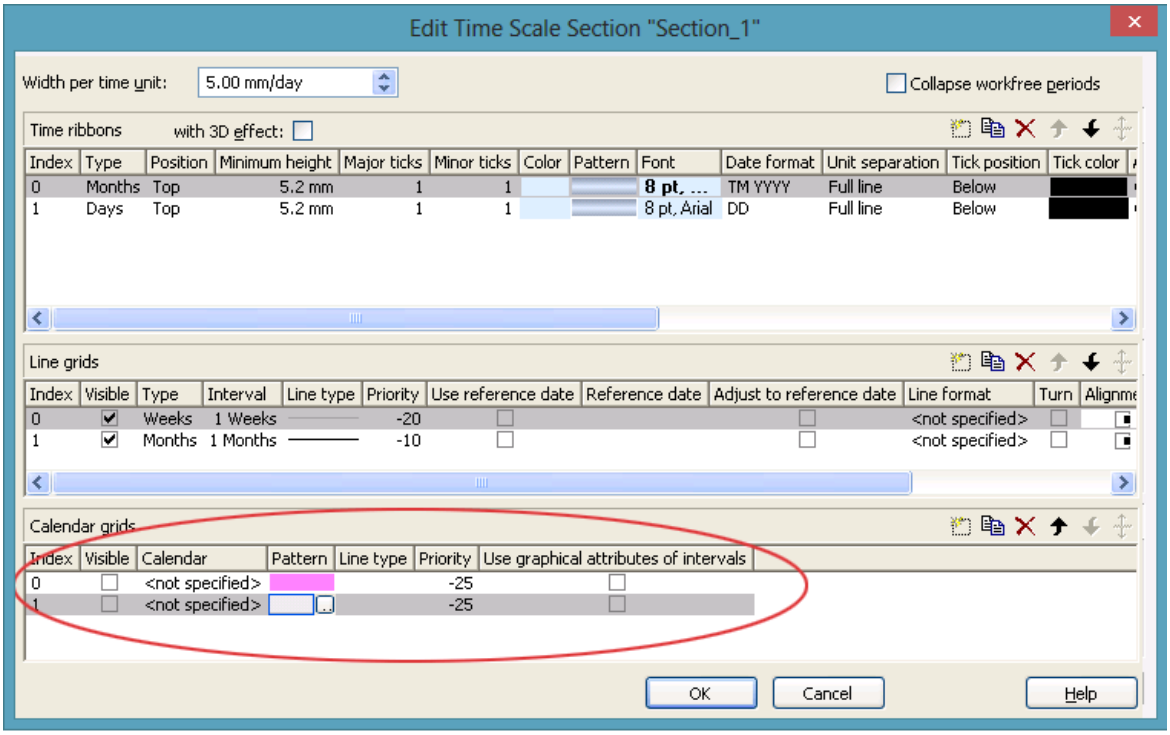
As a calendar grid for the histogram VARCHART XGantt takes the first invisible calendar grid in the first section of the time scale, if there is no other one present. This is the same calendar grid that is used groupwise in the Gantt graph:



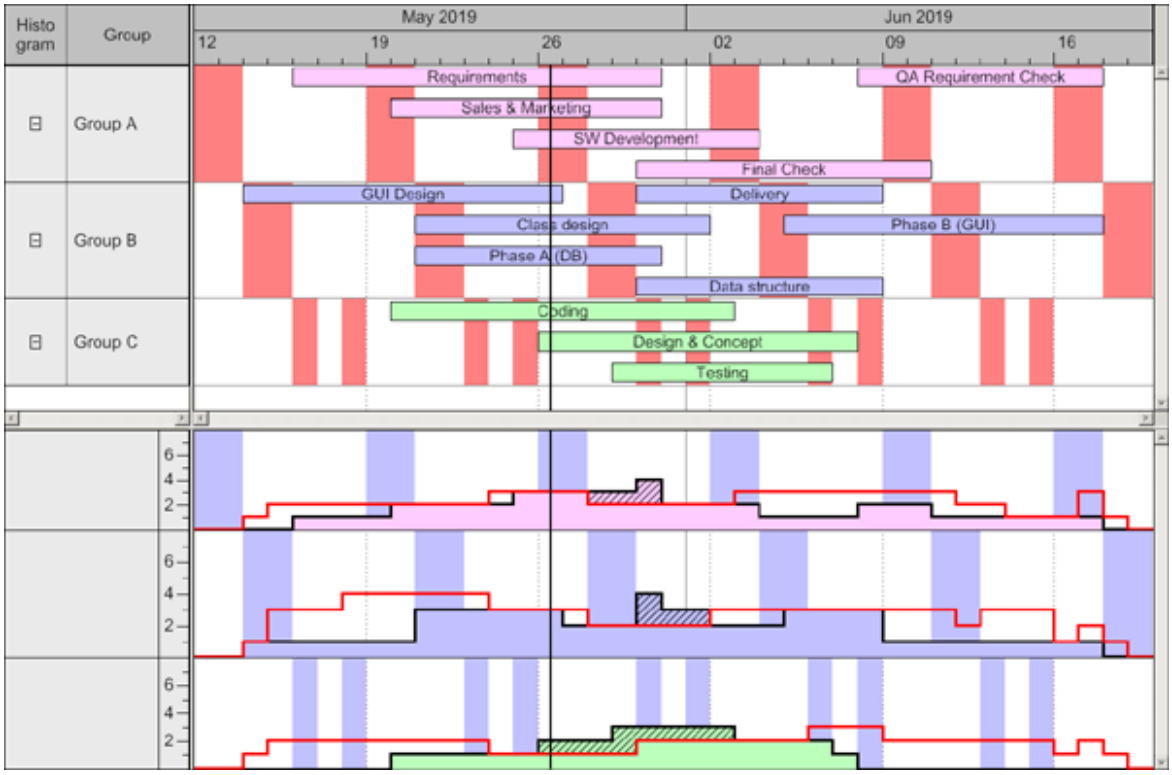
Thus the calendar grid will display the same appearance in the Gantt graph as in the histogram. In the example below it is a calendar grid that shows a different pattern for each group (groupwise calendar grid):



If you set another calendar grid to the time scale section, VARCHART XGantt will use this one for its histograms:



By using the second calendar grid, you can assign a different appearance compared to the calendar grid in the Gantt Graph. In our case, it shows a different color:


Histo gram
Group
May 2019
Jun 2019
12
19
26
02
09
16
Group A
Requirements
QA Requirement Check
Sales & Marketing
SW Development
Final Check
Group B
GUI Design
Delivery
Class design
Phase B (GUI)
Phase A (DB)
Data structure
Group C
Coding
Design & Concept
Testing
6
4
2

2.11 Printing the Diagram

If you have finished designing your diagram, you can finally print it. In runtime mode, select **Print** from the context menu (right mouse click in the empty diagram). This will take you to the Windows **Printing** dialog.

You also can use the method **PrintIt** of the object VcGantt to trigger the printing of the diagram.

If you want to edit the printer settings in runtime mode, you can select the menu item **Print setup...** from the context menu and pop up the corresponding Windows dialog.

The method **PrintDirect** of the object Vc Gantt lets you print the diagram directly. A dialog box will not be displayed.

If you want to edit the page settings at runtime, you can select **Page setup...** from the context menu or select **Print Preview** in the context menu and there click on the **Page Setup...** button.

You can also use the method **PageLayout** of the object VcGantt to open the corresponding dialog.

In the **Page Setup** dialog you can specify e.g. the scaling, whether the pages shall be numbered, the margins, the alignment etc. For further information see chapter 5.23 "Setting up Pages".

2.12 Exporting a Diagram

Your diagram can be exported as a graphics file:

- Select the menu item **Export graphics** from the default context menu. From there you will get to the Windows dialog **Save as**, where you can save the diagram as a graphics file.
- Use the API method **ShowExportGraphicsDialog** or **ExportGraphics-ToFile**.

Please find detailed information on graphics formats in the chapter: **Important Concepts: Graphics Formats**.

2.13 Saving the Configuration

You can store the settings of the property pages to an configuration external to your project at any time and re-load them when required. This is useful if you want to re-use previous settings or if you need the same settings for different projects.

A configuration is composed by two files of the same name but of different suffices, that is, an INI file and an IFD file, which both are indispensable.

> How to save your current configuration:

In the input box **Configuration file** you can specify the name of the file to which the current settings shall be stored. If the file name doesn' t exist and if you click on **Apply**, the INI file will be created and linked to the VARCHART ActiveX instance.

> How to re-load a configuration:

In the input box **Configuration file** you can specify the name of the file from which the settings shall be loaded. If the file exists and you click on **Apply** the configuration will be loaded and from then on, it will be linked to the VARCHART XGantt AcitveX instance. All current settings will expire irrevocably.

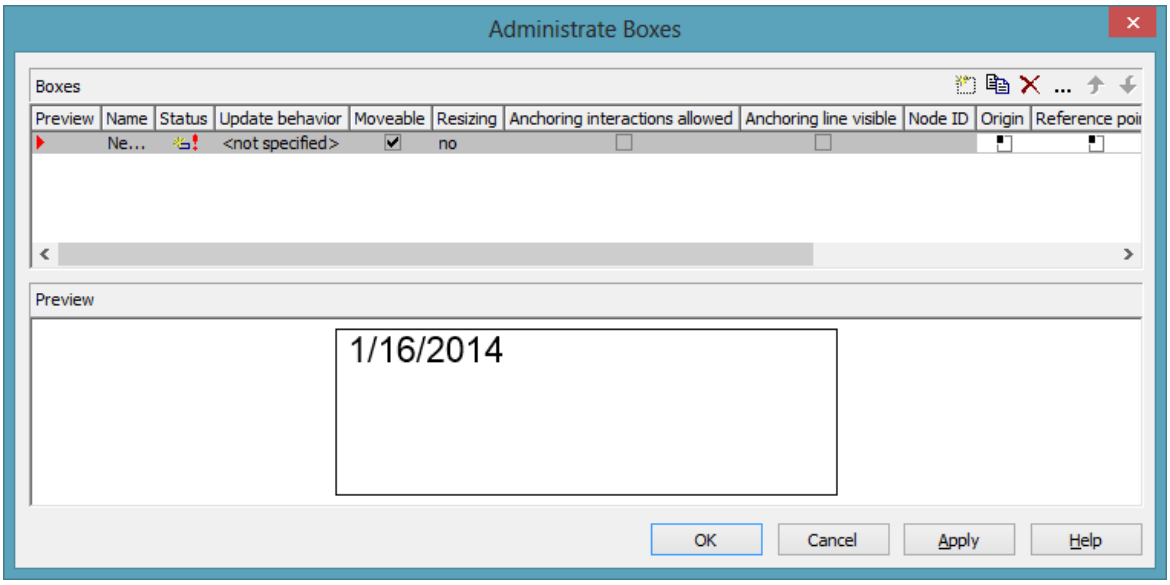
Note: The settings of the configuration file are loaded only once. VARCHART XGantt will not load them for a second time from the same file. Instead, the settings will be loaded from the internal storage, which are the same as those in the configuration file.

Thus, modifying the data of the configuration file by an editor will not work. If you want VARCHART XGantt to accept a modified configuration file, you have to rename the modified *ini* file and the corresponding *ifd* file and enter the name of the modified *ini* file on the **General** property page into the **Configuration file** field.

3 Important Concepts

3.1 Boxes

In a diagram area, boxes that contain texts or graphics can be displayed. On the property page **Objects**, please click on the **Boxes...** button to open the dialog **Administer Boxes...**. You can add, copy, delete or edit boxes.



By the properties **Origin**, **Reference point**, **X Offset** and **Y Offset** you can position a box in the diagram area. Relative positions of boxes do not depend on diagram size.

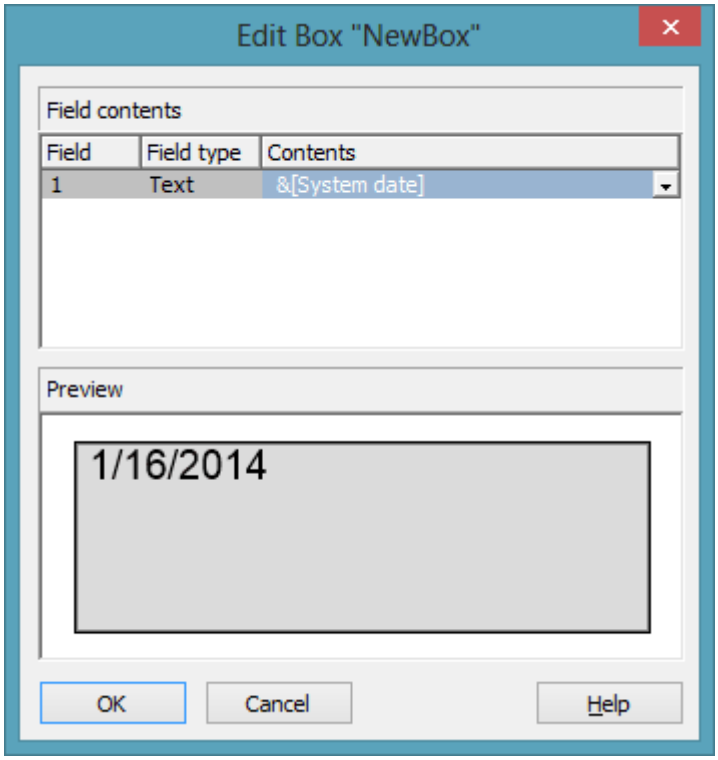
To a box you can set the below features:

- its name
- whether it can be moved in the diagram at run time
- whether and how ist size can be modified interactively
- whether anchoring interactions by mouse or over context menu are possible
- whether the reference points of the node and of the box (origin, reference point) shall be linked by a line when using the anchoring tool
- a node ID to identify the node to which the respective box shall be tied
- ist origin (the point to which the reference point refers in x and y direction)

- ist reference point (the point to which the origin refers in x and y direction)
- ist x or y Offset (distance between origin and reference point in x or y direction)
- type, thickness and color of the box frame line
- ist priority in relation to other diagram objects (nodes, grids, etc.)
- whether the box should be visible
- the box format

> Editing boxes

The **Edit Box** dialog lets you specify the contents of the fields. At desing time, you can make it appear by clicking on the **Edit box** button in the **Administer Boxes** dialog box. At run time you can make it pop up by double-clicking on a box. You also can edit the texts of boxes directly at run time after having selected **Allow in-place editing** on the property page **General**.



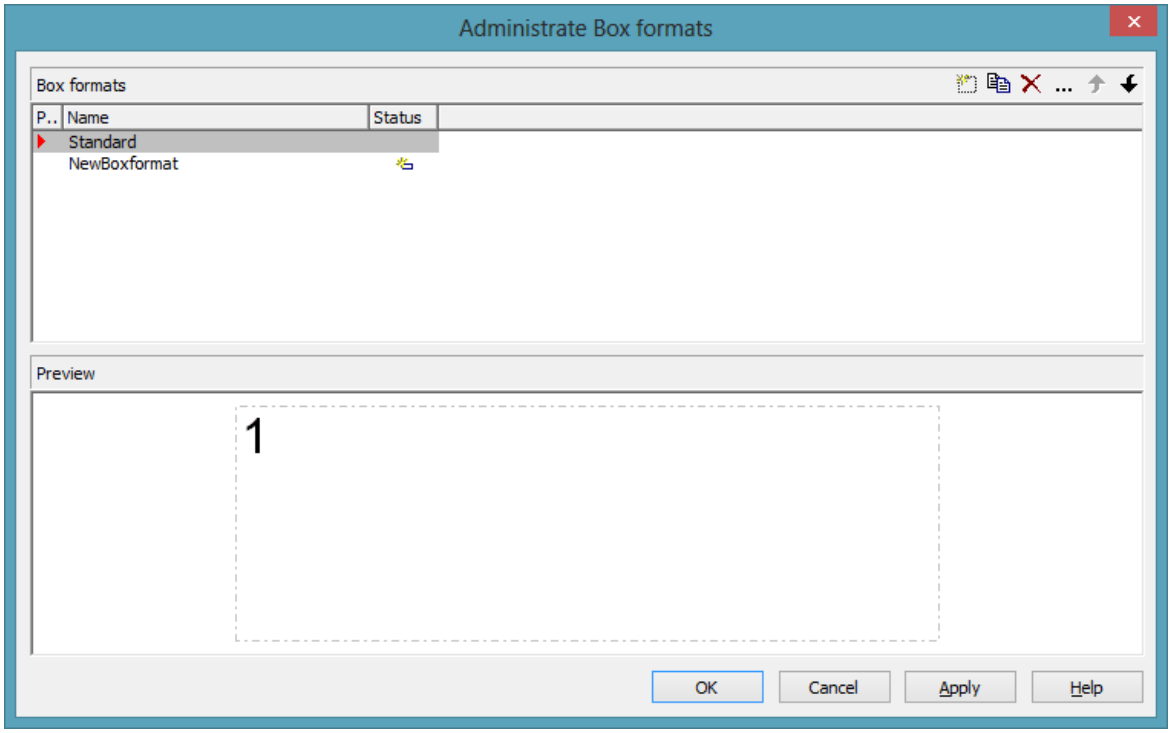
The **Field** column contains the numbers of the box fields. The number of fields depends on the selected box format (see further below).

The **Field type** column displays the field types (text or graphics).

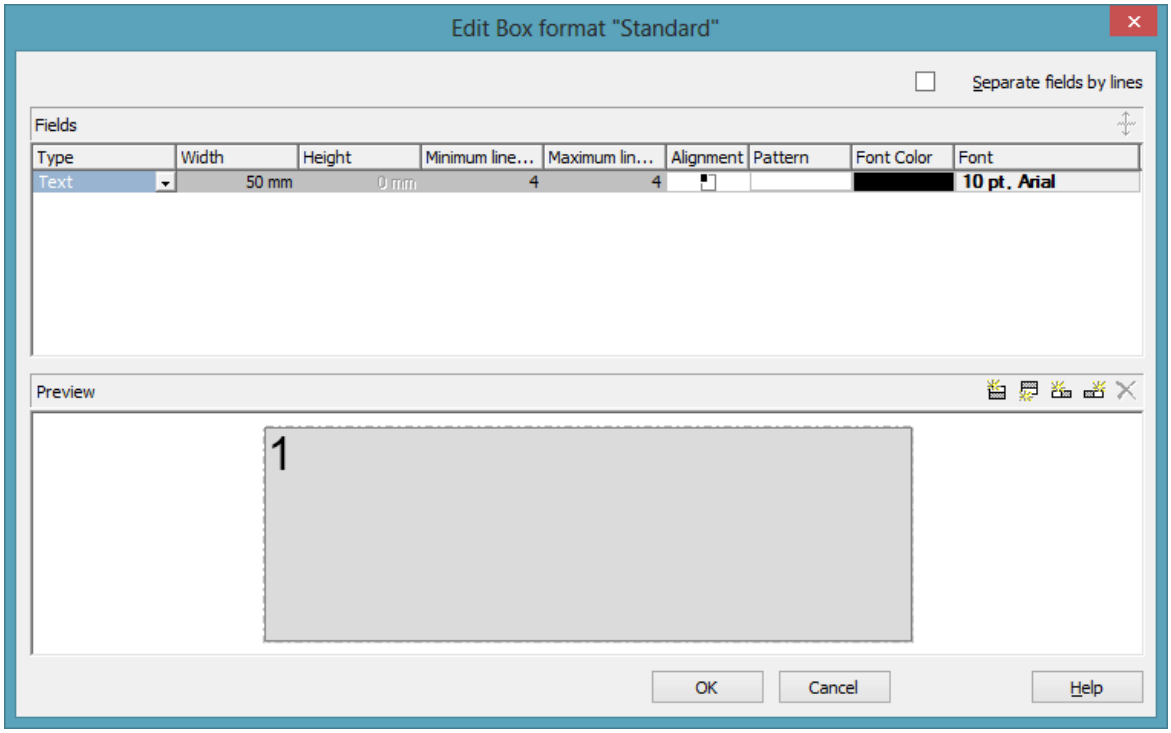
You can type the contents of the field or a graphics file name into the **Contents** column. If a text field contains more than one line, you can use "\n"

to set line breaks (Example: "Line1\nLine2"). If you do not set line breaks, the lines will automatically be divided where blanks are.

For a box, a format can selected which can be configured. In the **Administer Box Formats** dialog box you can add, copy, delete or edit box formats. The dialog box will appear after clicking on the **Edit** button of the **Box format** field in the **Administer Boxes** dialog box.



In the **Edit Box Format** dialog box you can specify the box format. This dialog box will appear if you click the **Edit box** button in the **Administrate Box Formats** dialog box.



You can tick whether the box fields are to be separated by lines.

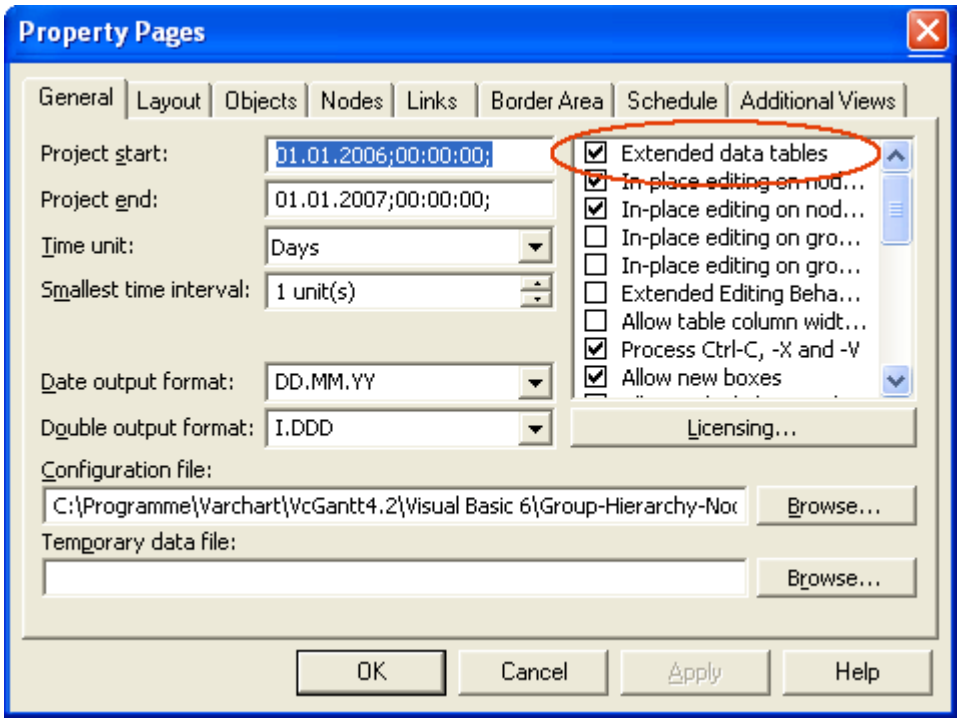
Beside, the below features can be set to a box:

- field type (text or graphics)
- width and height
- how many lines of text can be displayed in the current field
- alignment
- background color and fill pattern
- font attributes

3.2 Data Tables

As a data base for the graphical display of Gantt charts VARCHART XGantt uses two standard data tables for nodes and links, the fields of which can be individually defined. In version 4.0 this concept was extended. Up to 90 data tables can be defined and 1:n relations can be set up between the tables. Similar to data bases, the data is structured in data sets that depend on each other, which avoids data redundancies and supplies the data required by the integrated resource scheduling module.

For reasons of compatibility to existing applications VARCHART XGantt continues to operate in the previous mode by default. Only by activating the corresponding option at design time or at run time the extended data tables can be used. You can find the option **Extended data tables** on the property page **General**:



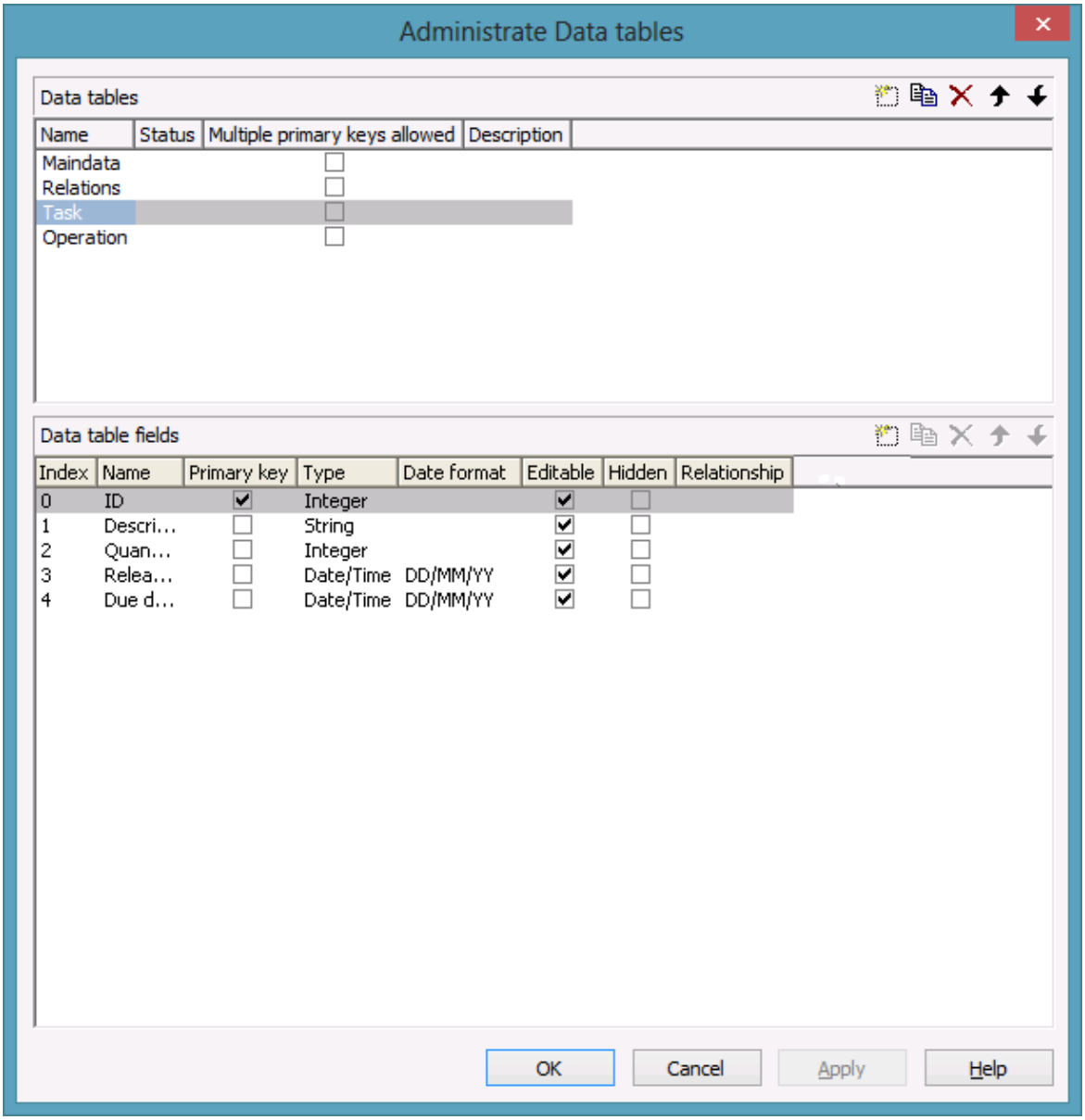
In the programming interface, the extended data tables are switched on at runtime by setting the VcGantt property **ExtendedDataTables** to **True**.

> Handling Data Tables

By default, the data tables **Maindata** and **Relations** exist. On the property page **Objects** you can click on the button **Data Tables…** to get to the dialog **Administrate Data Tables**. Generating new data tables requires to have switched on the **Extended data tables** mode before. The data tables**Task** and

Operation in the picture below were created by clicking on in the section **Data Tables**.

In the section **Data Table Fields** you can edit the fields of the above selected table. You can generate new fields by , delete existing fields by or copy fields by , as shown below.



The column **Index** is essential when using the API, since the contents of the data fields can only be addressed via the index. If you modify the sequence of fields in this dialog, i.e. the index, after having produced programming code, you need to adapt the programming code that accesses the corresponding field.

If you modify the data type, you may accordingly have to adapt formats and layers already defined to ensure that the appropriate data type is used when the fields are accessed.

The primary key feature is to be set to a field if you want a data record to be identified uniquely. For a data table referred to by a relation, setting a primary key is compulsory. The primary key may also consist of more fields - *but only up to three*. For a detailed description of the use of composite primary keys see chapter **The Administrate Data Tables Dialog Box**.

Relating tables is useful if the content shows a 1:n relation and if a subordinated data record should directly refer to a data field of the main data record.

Between two tables A and B at the moment only a single 1:n relationship can be established; a second field of B is not allowed to refer to the primary key of A. Nevertheless, a field of a third table C is allowed to refer to the primary key of table A.

Note: If a data table with a composite primary key is used in a relationship, the relationship has to match the primary key. Otherwise a unique connection is not possible. If the relationship is not defined correctly - which is checked neither at the API nor in the **Administrate Data Tables** dialog, the data record will not be connected. This leads to the event **OnDataRecord-NotFound**.

In the sample below a relation is created between the tables **Operation** and **Task** by setting **Task:ID** in the column **Relationship**.

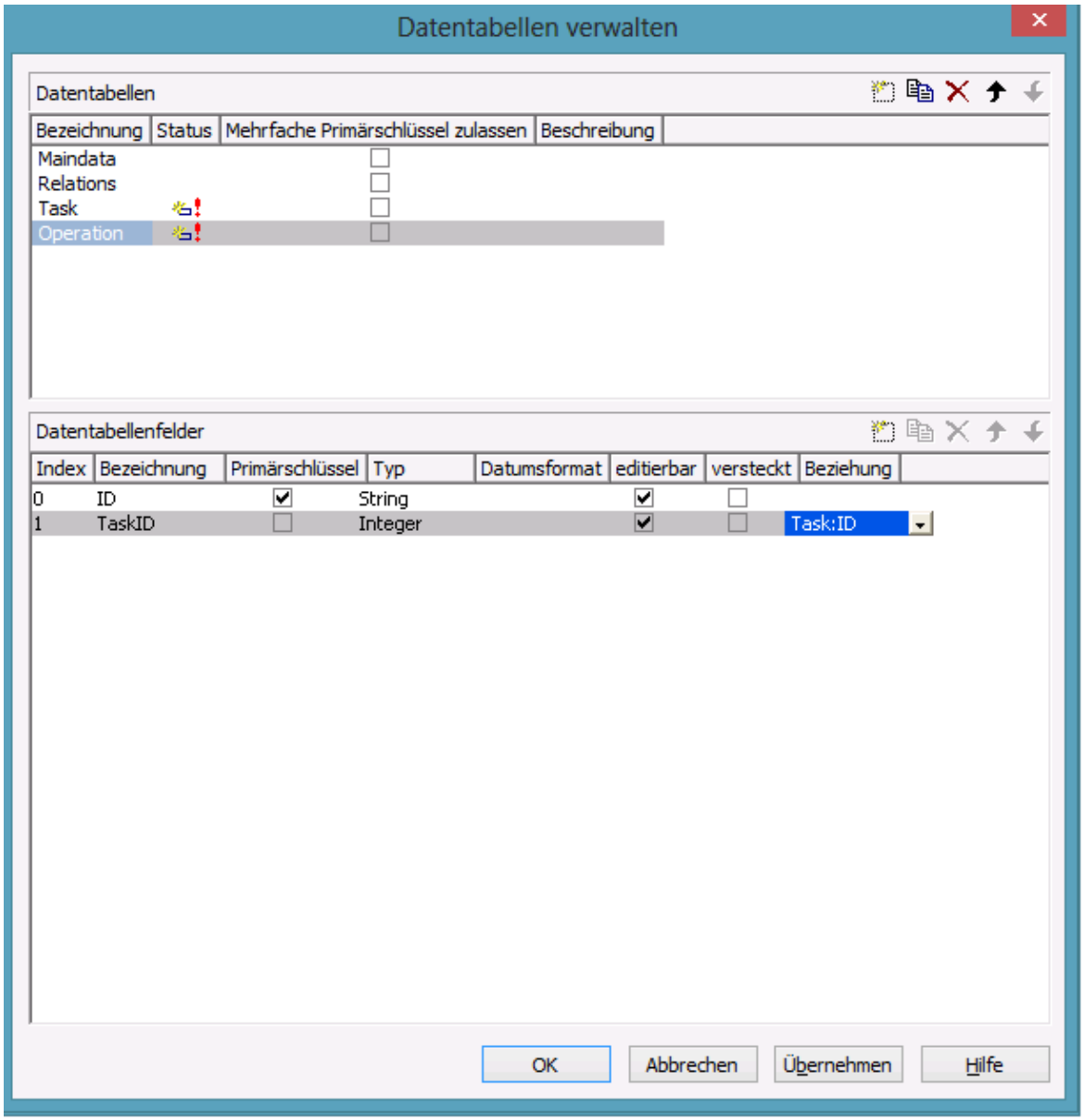


Table Task:

ID	Description	Quantity	Release date	Due date
1	Task 1	10	12.05.07	20.05.07
2	Task 2	20	01.06.07	15.06.07

Table Operation:

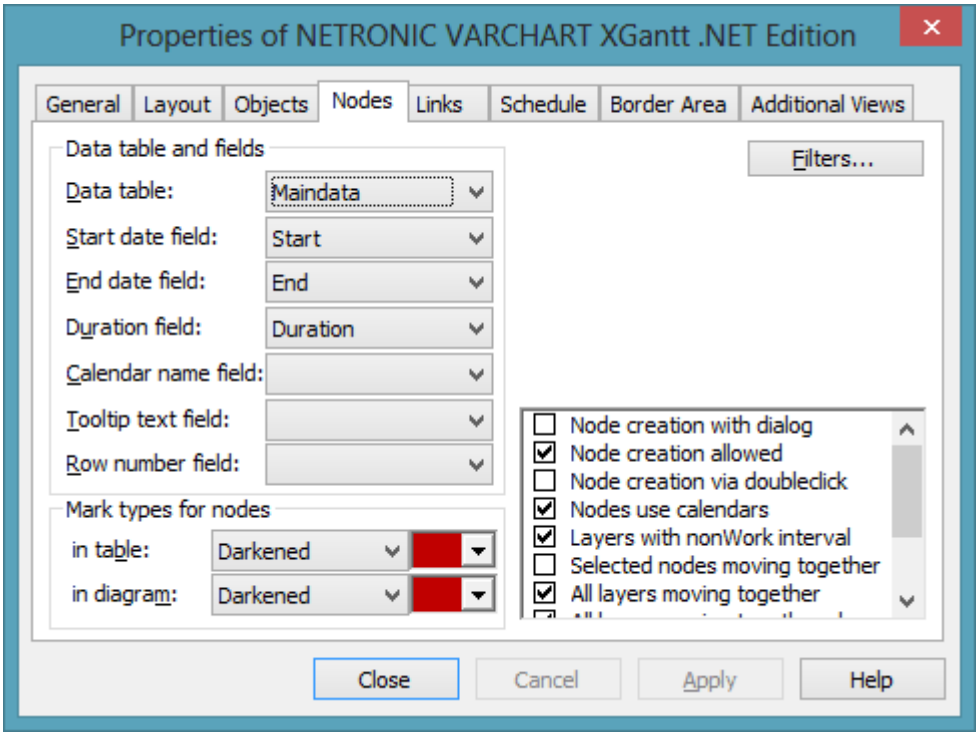
ID	TaskID	Description	Start	End
1	1	Operation 1	12.05.07	14.05.07
2	1	Operation 2	15.05.07	19.05.07

ID	TaskID	Description	Start	End
3	2	Operation 3	01.06.07	05.06.07
4	2	Operation 4	05.06.07	11.06.07
5	2	Operation 5	11.06.07	15.06.07

Example Code

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable

Set dataTableCltn = VcGantt1.DataTableCollection
Set dataTable = dataTableCltn.DataTableByName("Task")

dataTable.DataRecordCollection.Add ("1;Task 1;10;12.05.2007;20.05.2007")
dataTable.DataRecordCollection.Add ("2;Task 2;10;01.06.2007;15.06.2007")

Set dataTable = dataTableCltn.DataTableByName("Operation")
dataTable.DataRecordCollection.Add ("1;1;Operation
1;12.05.2007;14.05.2007")
dataTable.DataRecordCollection.Add ("2;1;Operation
2;15.05.2007;19.05.2007")
dataTable.DataRecordCollection.Add ("3;2;Operation
3;01.06.2007;05.06.2007")
dataTable.DataRecordCollection.Add ("4;2;Operation
4;05.06.2007;11.06.2007")
dataTable.DataRecordCollection.Add ("5;2;Operation
5;11.06.2007;15.06.2007")

VcGantt1.EndLoading
```

Depending on the data table selected on the property page **Node** in the **Data table** section, the graphical display of the nodes may originate from different bases. When creating nodes interactively, the base is the table to which new data records are added automatically. The corresponding rows displayed by the visualization are influenced by the active node filter, by grouping and by display options.



This is the result in the table of the Gantt chart if the table **Operation** was selected as base. The entries for Description, Quantity and Due date originate from the main table **Task**.

Description	Quantity	Due date	Operation
Task1	10	20.05.07	Operation1
Task1	10	20.05.07	Operation2
Task2	20	15.06.07	Operation3
Task2	20	15.06.07	Operation4
Task2	20	15.06.07	Operation5

If the table **Task** instead of **Operation** is used, the visible table in XGantt will consist of two entries only.

ID	Description	Quantity	Due date	Operation
1	Task 1	10	20.05.07	
2	Task 2	20	15.06.07	

In version 4.0 of VARCHART XGantt new object types are available that will replace the former ones. For reasons of compatibility, the former object types have been preserved in the present version. In new applications and in updates of existing applications the new objects should be used only.

Former	Present from Version 4.0 Onward
VcDataDefinition	VcDataTable
VcDefinitionTable	VcDataTableFieldCollection
VcDefinitionField	VcDataTableField
	VcDataRecord

Please find a graphical display of objects, methods and properties here:

> Creating and modifying data records

After having defined the data table fields, you can add data records to a table by the API. There are two ways of adding data to your records. We recommend the common practice of defining an array of the type variant with the number of its elements corresponding to the number of the data table fields.

Example Code

```
Dim dataTable As VcDataTable
Dim dataRecCltn As VcDataRecordCollection

Dim dataRecVal() As Variant
Dim dataRec1 As VcDataRecord
Dim dataRec2 As VcDataRecord

Set dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
Set dataRecCltn = dataTable.DataRecordCollection

ReDim dataRecVal(dataTable.DataTableFieldCollection.Count)

dataRecVal(0) = 1
dataRecVal(1) = "Node 1"
dataRecVal(2) = DateSerial(2007, 1, 8)
dataRecVal(4) = 8
```

A data record can be added by the method Add() of the **DataRecord-Collection**, the variant array being passed as parameter.

Example Code

```
Set dataRec1 = dataRecCltn.Add(dataRecVal)
```

As a second way you can use a string consisting of data values which are separated by semicolons.

Example Code

```
Set dataRec2 = dataRecCltn.Add("2;Node 2;15.01.07;;9")
```

If the data value itself contains a semicolon, the string has to be enclosed in double inverted commas.

Example Code

```
Set dataRec2 = dataRecCltn.Add("2; ""Node 2;"";15.01.07;;9")
```

The reference to a data base object can quickly be found by the method **DataRecordByID ()** and the primary key.

Example Code

```
Set dataRec1 = dataRecCltn.DataRecordByID(1)
Set dataRec2 = dataRecCltn.DataRecordByID(2)
```

The contents of the single data fields of a data record can easily be modified by using the indexed property **DataField()**. For replacing the contents of all data fields of a record the property **AllData** is very useful.

Example Code

```
dataRec1.DataField(Main_ID) = 1
dataRec1.DataField(Main_Name) = "Activity X"
dataRec1.DataField(Main_Start) = DateSerial(2007, 1, 4)
dataRec1.DataField(Main_Duration) = 12
dataRec1.UpdateDataRecord

dataRec2.AllData = "2;Activity Y;18.01.07;;5"
dataRec2.UpdateDataRecord
```

A modification of a record is only displayed in the chart after the method **Update()** of the object **DataRecord** was called.

Reading data field values by **Alldata** serves to quickly display data during design time and to easily transfer the contents of a data record to the record of a different table.

Example Code

```
Dim content As String
content = dataRec1.AllData & vbCr & dataRec2.AllData & vbCr &
dataRec1.DataField(Main_Name)
MsgBox (content)
```

Note: In order to improve the legibility when accessing data fields you can define global constants the names of which are more descriptive than index numbers.

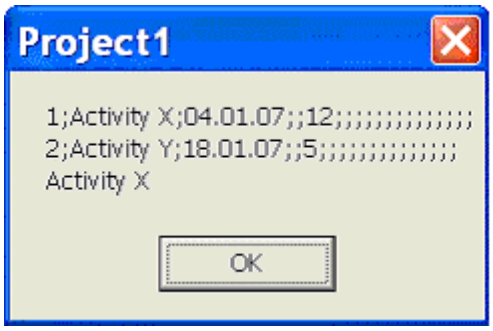
Below please find the coherent code partition.

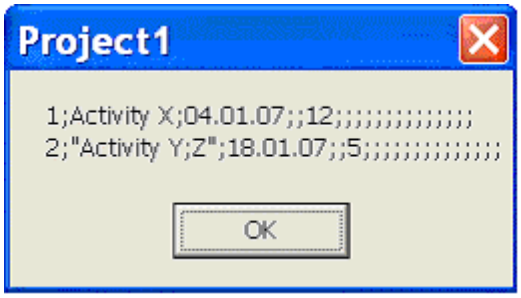
Example Code

```
Const Main_ID = 0
Const Main_Name = 1
Const Main_Start = 2
Const Main_Duration = 4

'...
```

```
Dim dataRec1 As VcDataRecord
Dim dataRec2 As VcDataRecord

Dim content As String

VcGantt1.TimeScaleEnd = DateSerial(2008, 1, 1)
VcGantt1.TimeScaleStart = DateSerial(2007, 1, 1)

VcGantt1.ExtendedDataTablesEnabled =  True
Set dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
Set dataRecCltn = dataTable.DataRecordCollection

ReDim dataRecVal(dataTable.DataTableFieldCollection.Count)

dataRecVal(Main_ID) = 1
dataRecVal(Main_Name) = "Node 1"
dataRecVal(Main_Start) = DateSerial(2007, 1, 8)
dataRecVal(Main_Duration) = 8
Set dataRec1 = dataRecCltn.Add(dataRecVal)

dataRecCltn.Add("2;Node 2;15.01.07;;9")

VcGantt1.EndLoading

'...

Set dataRec1 = dataRecCltn.DataRecordByID(1)
Set dataRec2 = dataRecCltn.DataRecordByID(2)

dataRec1.DataField(Main_ID) = 1
dataRec1.DataField(Main_Name) = "Activity X"
dataRec1.DataField(Main_Start) = DateSerial(2007, 1, 4)
dataRec1.DataField(Main_Duration) = 12
dataRec1.UpdateDataRecord

dataRec2.AllData = "2;Activity Y;18.01.07;;5"
dataRec2.UpdateDataRecord

content = dataRec1.AllData & vbCr & dataRec2.AllData & vbCr &
dataRec1.DataField(Main_Name)
MsgBox (content)

'...

dataRec2.AllData = "2;""Activity Y;Z"";18.01.07;;5"
dataRec2.UpdateDataRecord
content = dataRec1.AllData & vbCr & dataRec2.AllData
MsgBox (content)
```

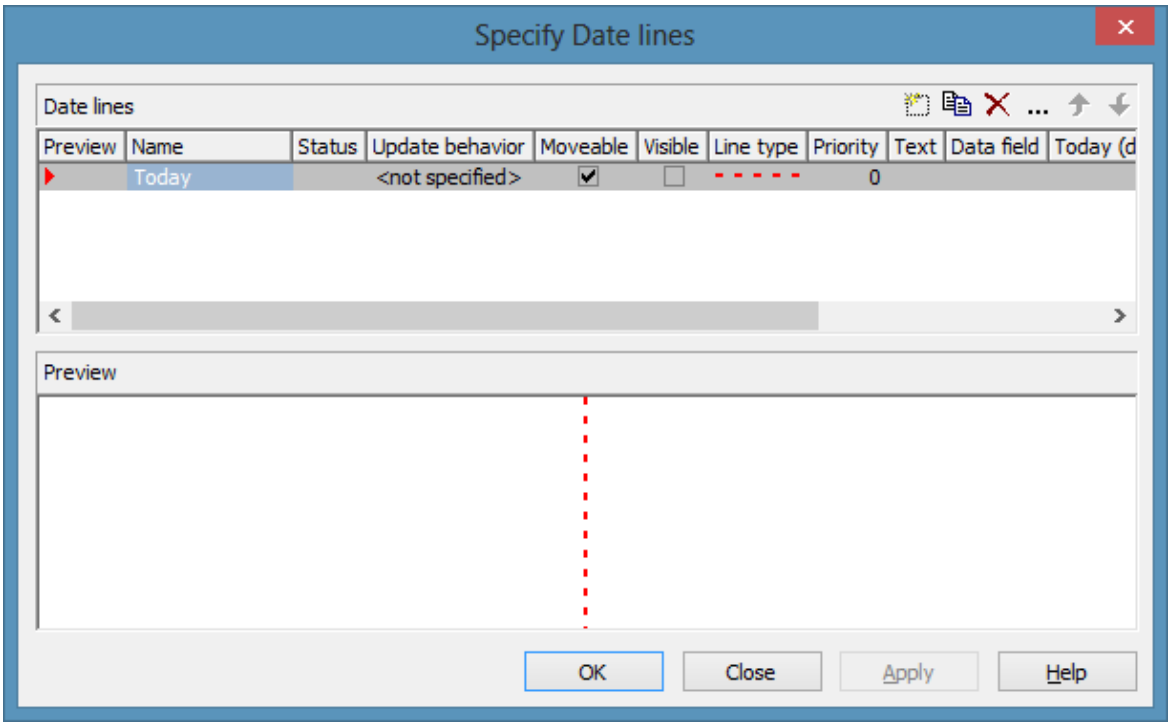
This is the output:


Project1
1;Activity X;04.01.07;;12;;;;;;;;;;;;;;
2;Activity Y;18.01.07;;5;;;;;;;;;;;;;;
Activity X
OK

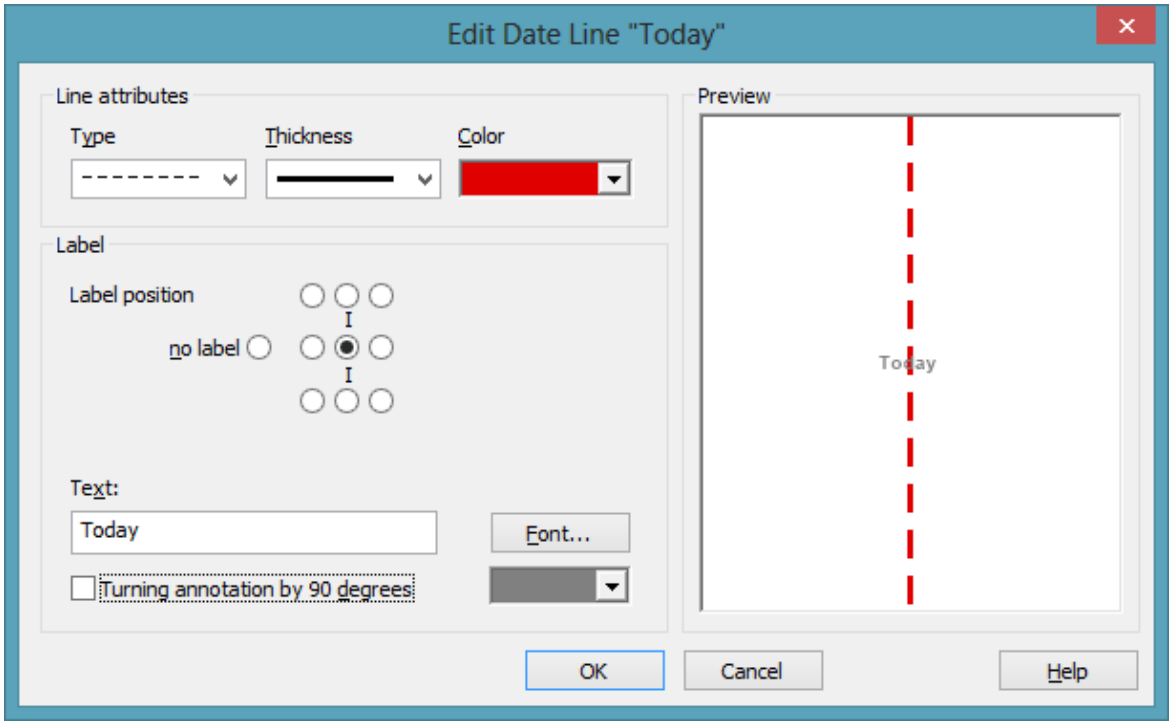

Project1
1;Activity X;04.01.07;;12;;;;;;;;;;;;;;
2;"Activity Y;Z";18.01.07;;5;;;;;;;;;;;;;;
OK

3.3 Date Lines

Date lines (vertical lines in the diagram) allow to highlight certain dates. The attributes of date lines (date, line type, priority (in relation to other date lines), whether they are visible or can be moved are defined in the below dialogs. Click on **Date lines** on the **Objects** property page to open the first one; the next one pops up by clicking the **Edit** button:



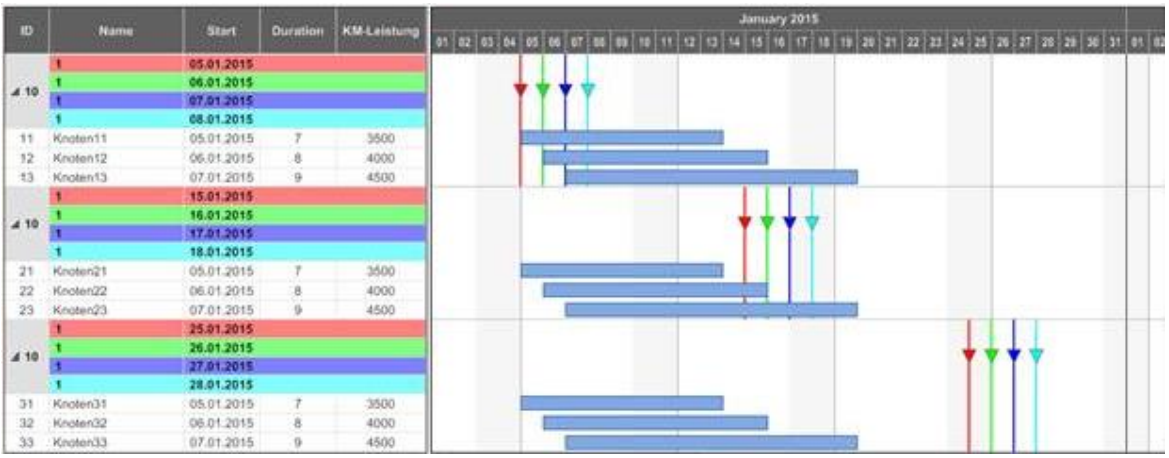
Specify Date Lines



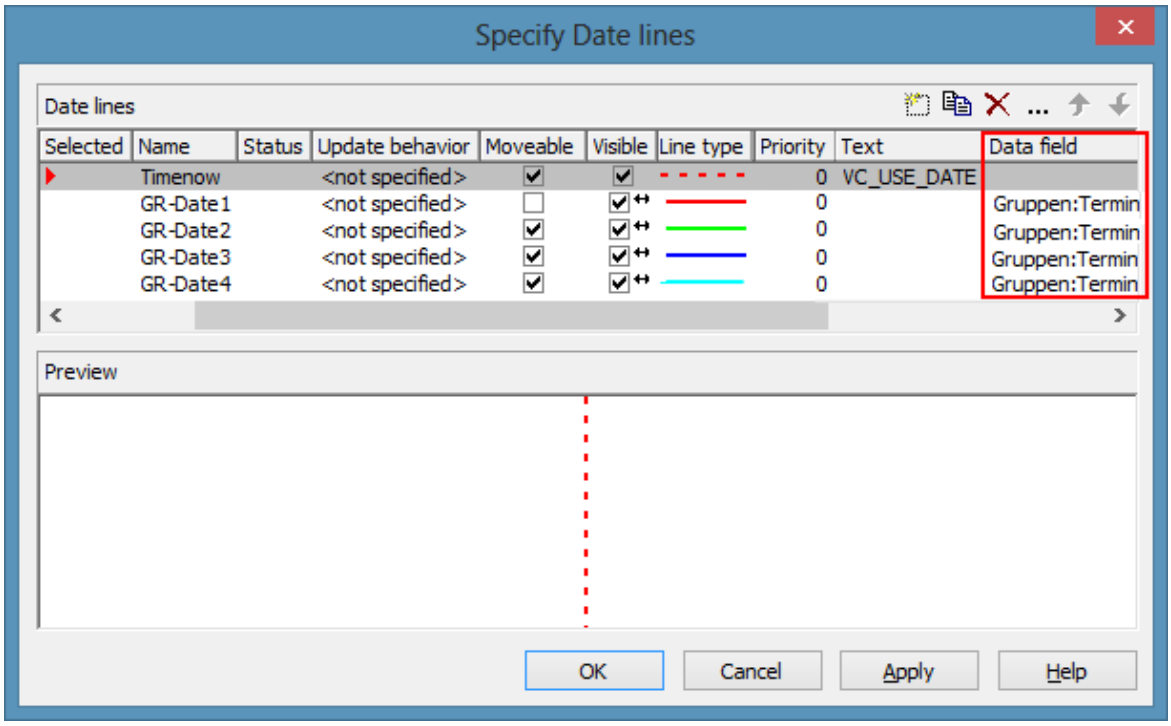
Edit Date Line

Individual, data-based date lines

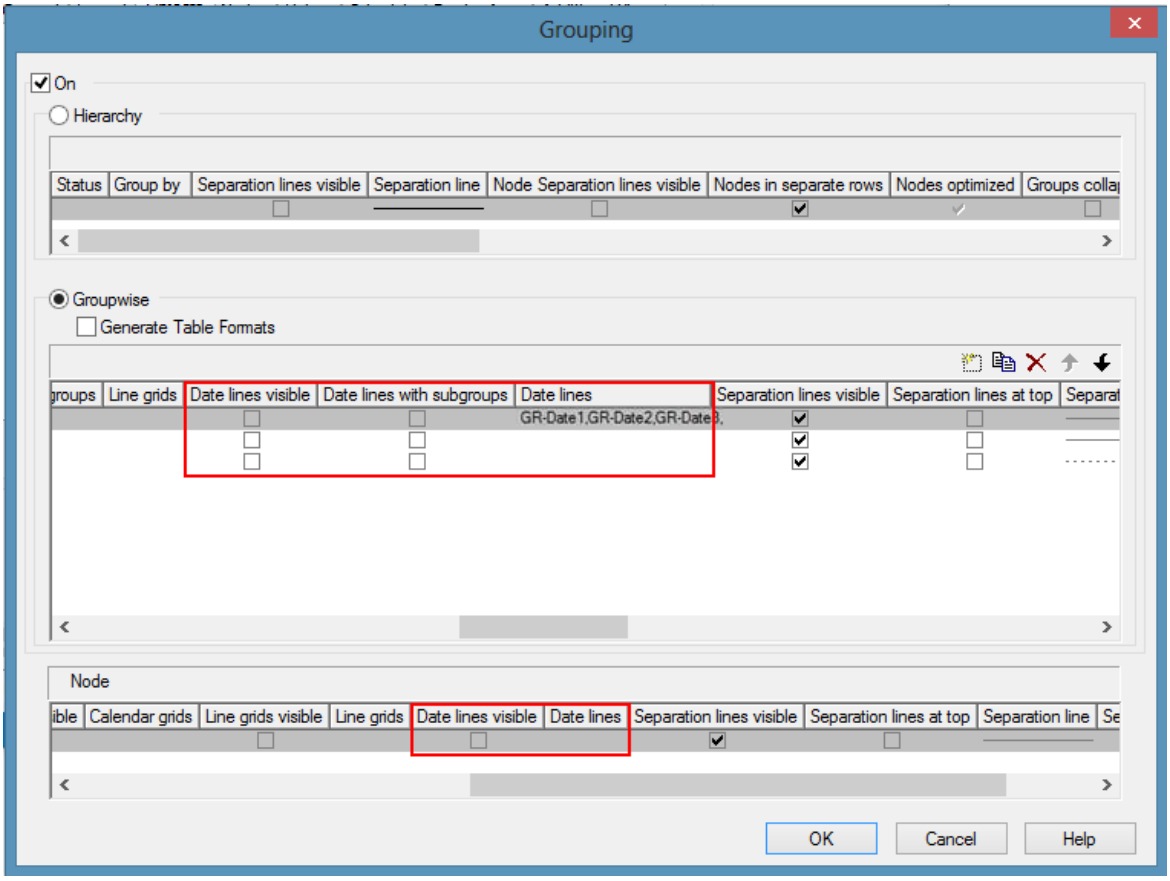
Besides the fixed date, date lines can also use a date from a node or group record. This means that for each node or group record an indivdual date line as graphic copy can be created, using the properties (color etc.), except date, from the underlying date line of the DateLine Collection, date and position in the plan being individual, however. Such date lines are only drawn within the ribbons of nodes or groups by using **NodeLevelLayout** or. **GroupLevelLayout**, resp., (see picture below: four date lines have been created and placed for three groups individually; four symbol layers of the activated group node use the same dates as the date lines).



For this, a data field has to specified for the date:



Note: When a data field has been individually specified, the date from the record has priority over the fixed date (**VcDateLine.Date**). When no date could be identified, e.g. because the data field is empty in the record, the date line has to be linked to a data record. This is done by the according settings in the **Grouping** dialog:



The corresponding API commands:

VcGroupLevelLayout.ShowDateLines

VcGroupLevelLayout.DateLinesWithChildGroups

VcGroupLevelLayout.DateLineName

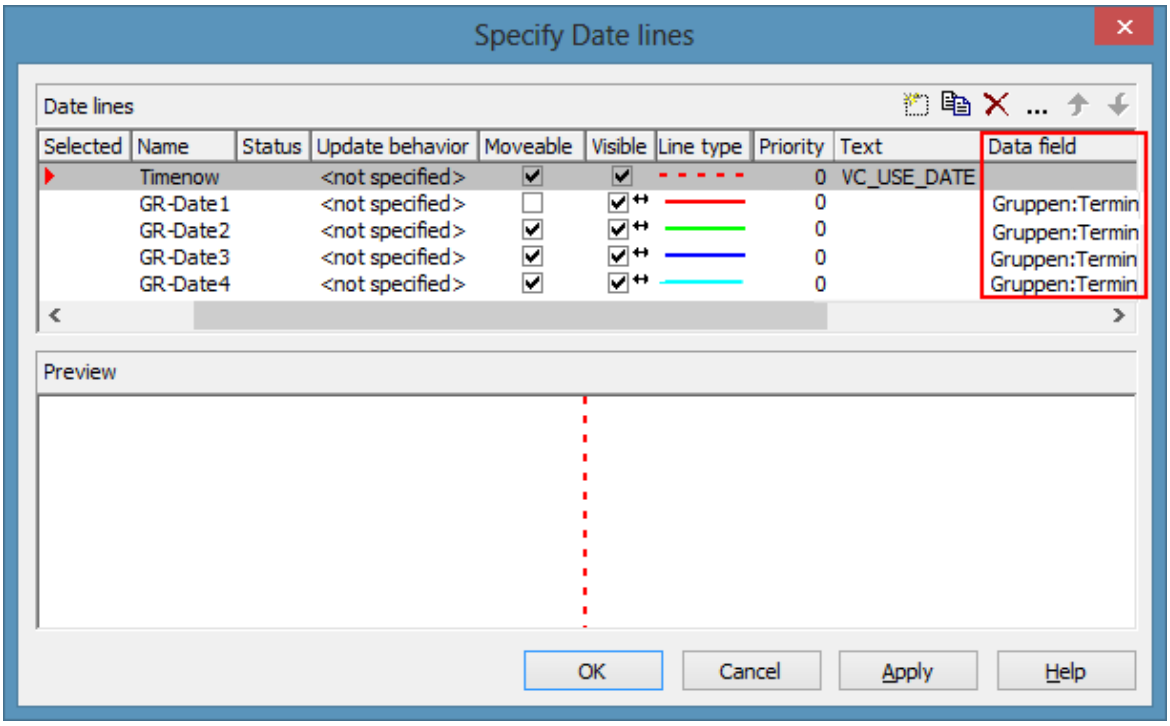
VcNodeLevelLayout:.ShowDateLines

VcGroupLevelLayout.DateLineName

Labeling Date Lines

Date lines can be labeled. As a rule, this is done by a fixed text, Displaying the indvidual date might in some cases be wished for at at all, but especially at individiual date lines. The key word **VC_USE_DATE** manages to display the corresponding date at the specified place of the date line (**VcDateLine.LabelPosition**) in the specified date format (**VcGantt.DateOutputFormat**).

Tto make date lines visible individually the option **Visible** can be mapped and thus be set individually.



The corresponding API properties:

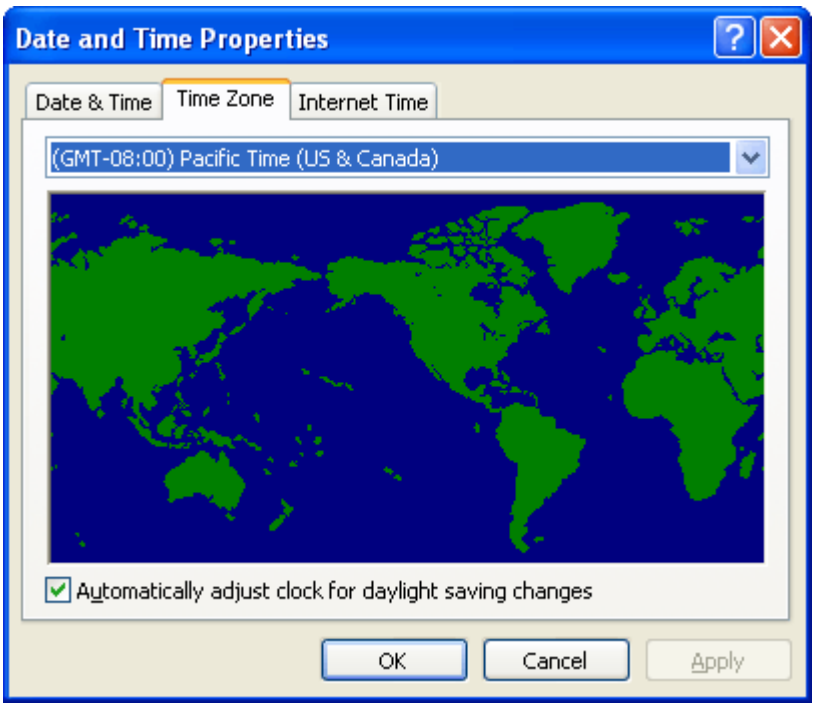
VcDateLine.VisibleDataFieldIndex

VcDateLine.VisibleMapName.

3.4 Dates and Daylight Saving Time

Dates in VARCHART components always refer to the time zone set in the system that the program is running on. It is not possible to set dates from different time zones; the dates have to be converted into dates of the time zone set to the system that your VARCHART component is running on before they are passed to the component. The component automatically refers to the information on the beginning and the end of daylight saving time which is present in the system.

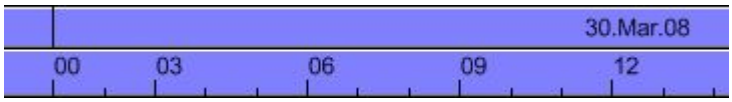
To make switching times known to a VARCHART component, the check box in the time zone dialog **Automatically adjust clock for daylight saving changes** needs to be ticked, as shown in the picture. You can find the dialog in the Windows operation system by clicking on the button **Start**, then on the menu item **Control Panel**, then on the icon **Date and Time**, or simply by double-clicking on the time display in the task bar of the main window.



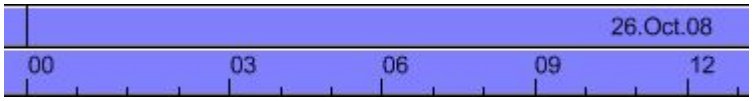
When switching, a VARCHART component uses the start date and the end date including hour, month and day of daylight saving time that usually are communicated by the system. This implies that the DST times of the years before and after the current year are extrapolated and true deviations probably existing of those years are ignored, since they are also unknown to the system. For example, a couple of years ago daylight saving time was prolonged for some weeks at the beginning and end. Since the system only knows the current rules, consequently dates in those periods will be interpreted in the wrong way.

At present, VARCHART components can only take into account a DST time offset of exactly one hour. Besides, the switch can only take place at full hour. Since a VARCHART component always receives and displays the date values of local time, at the beginning of the DST period there is an hour missing and at the end there are two hours of the same number. At present, the identical numbers are not discriminated when passed, returned or displayed.

The switching becomes visible in the time scale if its resolution is hours.



Switching between 0 and 3 o'clock in spring (1 hour missing)



Switching between 0 and 3 o'clock in autumn (1 hour twice)

> New Default Date From Version 4.3 Onward

If in a VARCHART component a date is retrieved that does not exist, up to version 4.3 the date **31.12.1899 00:00:00** was returned. From version 4.3 onward, a different date **01.01.0001 00:00:00** will be returned.

In certain situations this can lead to an argument-out-of-range exception which you can intercept by treating the exception.

If within your application program, for example a date is handled by DateTimePicker controls of .NET, and if you try to display an “empty” date, up to version 4.3 the date **31.12.1899 00:00:00** was displayed. The new default though, which is **01.01.0001 00:00:00** cannot be displayed by using the default settings of the DateTimePicker, so it will throw an **ArgumentOutOfRange** exception.

Your program should react to this; in any case you should write some treatment to this exception, otherwise an untreated exception could occur and could entail an unexpected end of program.

3.5 Dragging Tools

Gantt charts enable the planner to easily re-plan orders, tasks or resources by shifting them back and forth. However, positioning a node at a certain point of the timeline or directly after another node can be tricky because a certain spot in the Gantt has to be exactly hit by mouse.

Besides, in many Gantt charts, multi-level groups are used. In large plans dragging a node from one group or its subgroup to another one by mouse can at times get a bit inconvenient and confusing if the target group is located quite far away.

Snap Tools: Support for horizontal dragging

Many dragging applications or design tools already offer the so-called snap-grids as help for exactly positioning objects by means of a predefined grid, usually pixel-spaced. VARCHART XGantt now offers a similar functionality. The moved objects are not adjusted to a fixed grid but to other objects in the graphic, these objects thus defining a snap grid with irregular distances.

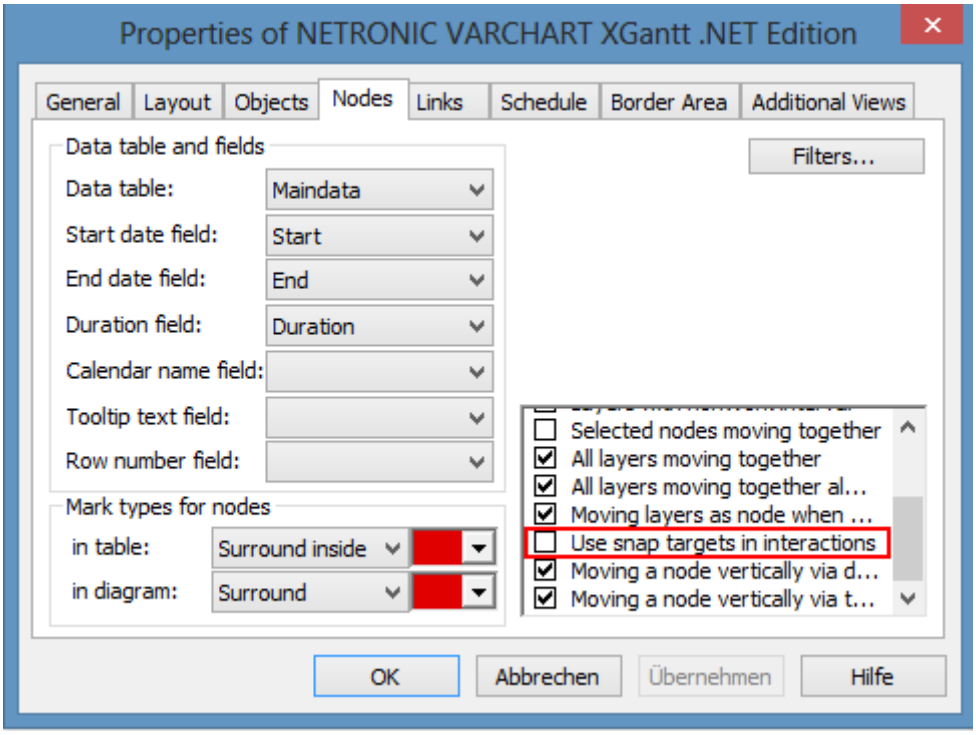
Nodes (or their layers), date lines, line grids and calendar grids allow to define so-called snap targets. That means that these objects define certain places at themselves serving as targets of a snap action of other objects. When moving a node horizontally or modifying the size of a node or a layer, start or end date of this node or layer will be chronologically adjusted to the defined snap tools of the other objects. The start or end date will move towards the snap target within 5 pixels next to it thus taking over the exact date of the target.

Special behaviors have been defined for each node layout (ungrouped, grouped, hierarchical arrangement; given that the according objects define snap tools):

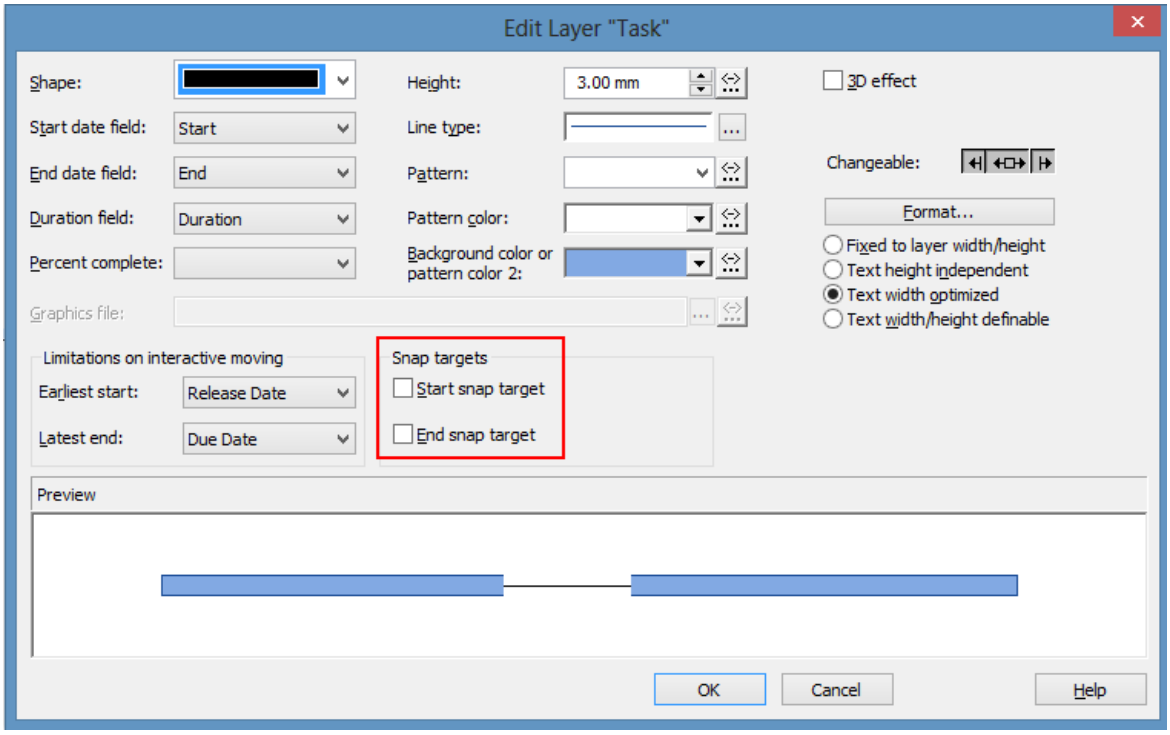
- All node layouts: the layer-to-be-moved is adjusted to date lines, line grids and calendar grids.
- Ungrouped layout: The layer-to-be moved is adjusted to the layers of all nodes.
- Grouped layout: The layer-to-be-moved is adjusted to the layers of the nodes of one group (without subgroups). If the group is changed during the interaction, the layer will be adjusted to the objects of the new group.
- Hierarchical arrangement: The-layer-to-be-moved will be adjusted to the layers of the nodes of the same branch (with sub-branches). If the branch

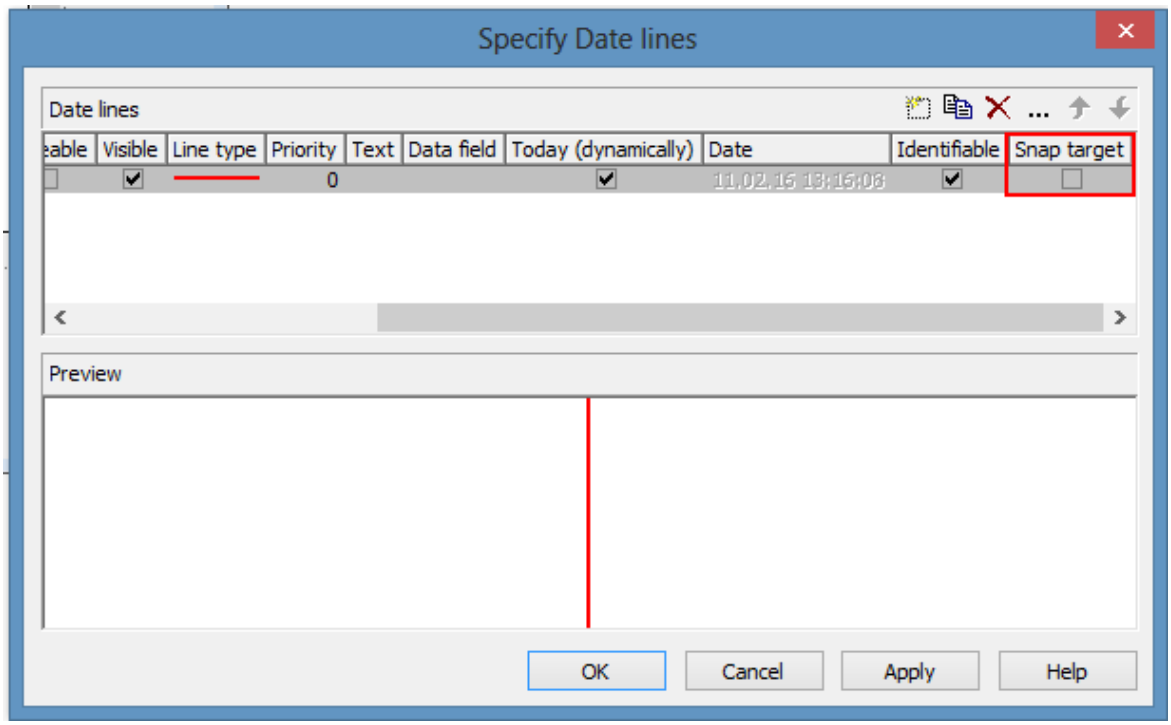
is changed during the interaction, the layer will be adjusted to the objects of the new branch.

- For the snap tools to take effect, they have to be enabled on the **Nodes** property page



- API call: **vcGantt.UseSnapTargetsInInteractions = true/false**
- Layers can be defined as snap targets in the Edit Layer dialog. Ticking the checkboxes Start snap target and End snap target sets the layer's position (i.e. its dates) as snap targets for dragging a node or layer.

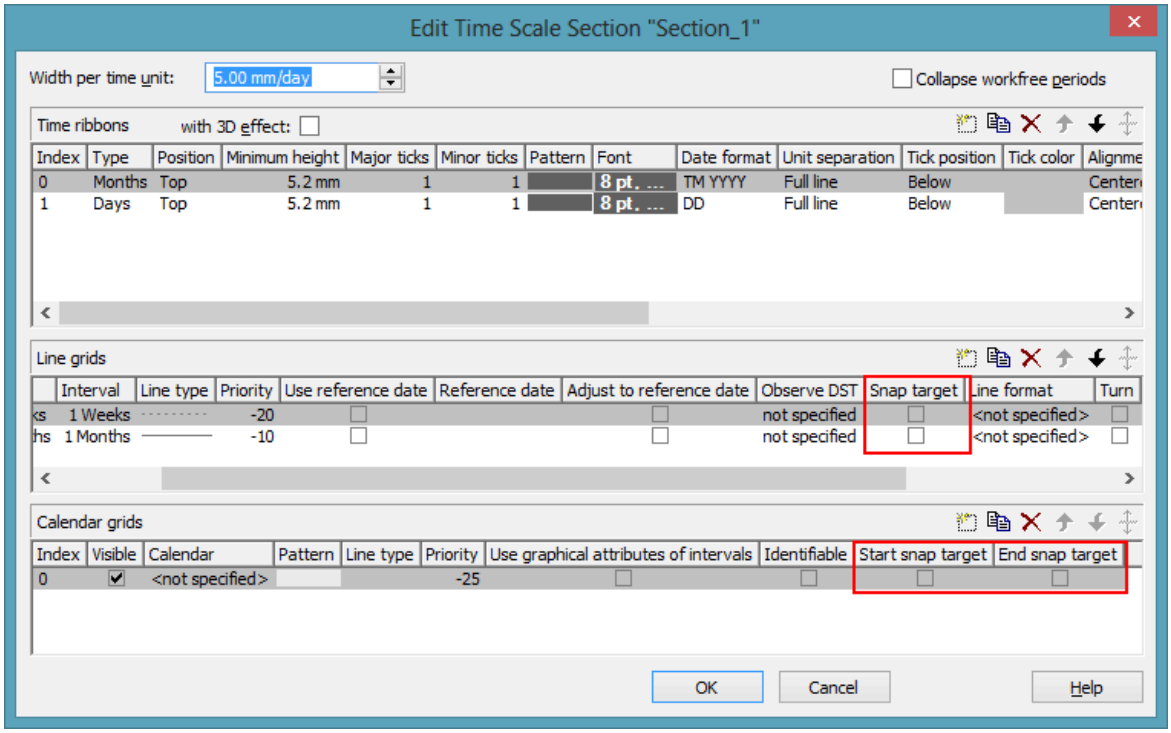
- 

- API calls:
- **VcLayer.StartSnapTarget = true/false**
- **VcLayer.EndSnapTarget = true/false**
- Date lines can be defined as snap targets in the Specify Date Lines dialog. Ticking the checkbox Snap target sets the date line's position (i.e. its dates) as snap target for dragging a node or layer.
- 

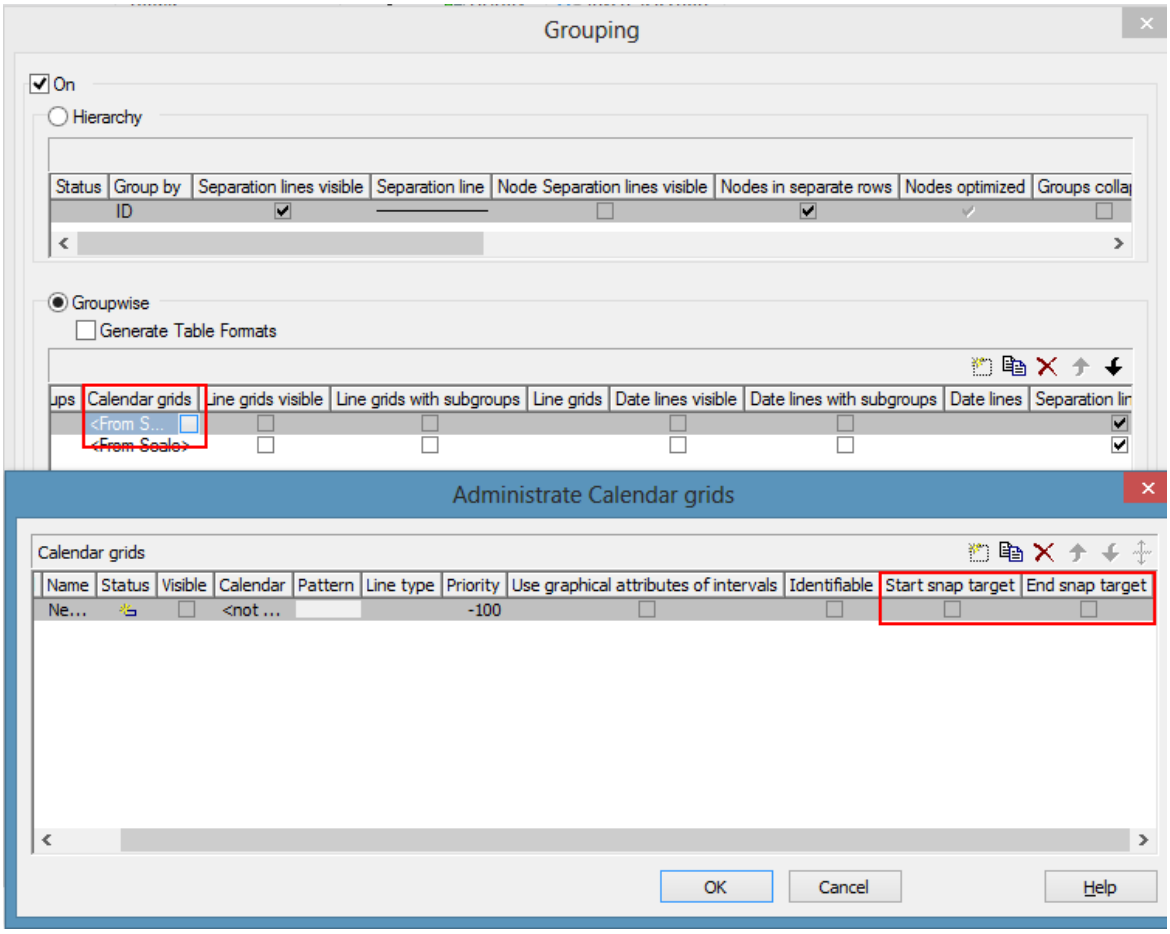
- API call: **VcDateLine.SnapTarget = true/false**

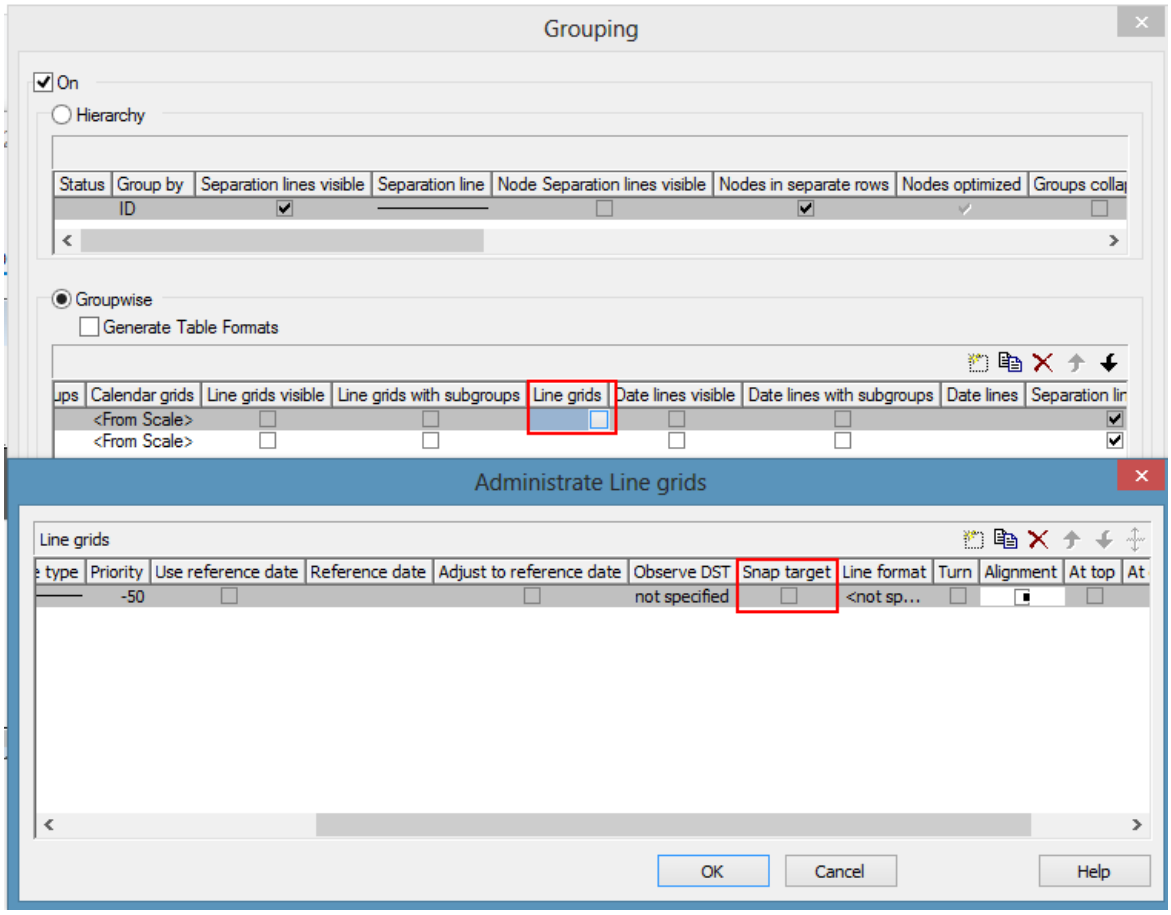
- **Snap target LINE GRIDS/CALENDAR GRIDS**
- Line grids and calendar grids can be defined as snap targets at two different places:
- In the Edit time scale section for not individual objects
- Below the Grouping dialog for individual, group- or node-related objects.

Ticking the according checkboxes in the Edit time scale section dialog sets the related objects' position (i.e. their dates) as snap targets for dragging a node or layer.



In the **Grouping** dialog you can access the dialogs **Administrate Calendar Grids** and **Administrate Line Grids**, where ticking the according checkboxes sets the related objects' position (i.e. their dates) as snap targets for dragging a node or layer.


Grouping
On
Hierarchy
Status
Group by
Separation lines visible
Separation line
Node Separation lines visible
Nodes in separate rows
Nodes optimized
ID
Groupwise
Generate Table Formats
Calendar grids
Line grids visible
Line grids with subgroups
Line grids
Date lines visible
Date lines with subgroups
Date lines
<From S...
<From Scale>
Administrate Calendar grids
Calendar grids
Name
Status
Visible
Calendar
Pattern
Line type
Priority
Use graphical attributes of intervals
Identifiable
Start snap target
End snap target
Ne...
<not ...
-100
OK
Cancel
Help



API calls:

VcDateLineGrid.SnapTarget = true/false

VcCalendarGrid.StartSnapTarget = true/false

VcCalendarGrid.StartSnapTarget = true/false

Please note: Since it makes no sense to mix the snap targets of all objects (i.e. the objects from several ribbons) when moving several nodes, snap targets of individual objects are only taken into account if a single node is moved. A separate snapping of a node to the snap target of the ribbon it is situated in is not provided for.

> Moving a node by arrow keys

Nodes can not only be moved interactively by mouse but also by the mouse keys on the keyboard. To do this, the following setting is needed:

vcGantt1.ArrowKeyMode = VcArrowKeyMode.vcResizeOrMoveNode

The value **vcNodeJumpToSnapTarget** was added to the enumeration **VcArrowKeyMode**. If this value is set, pressing CTRL + left or right arrow key causes a marked node to snap to the next or the last snap target, this

being s a cyclical operation: If the end is reached, everything starts at the beginning again.

Auto collapse/expand: Support for vertical dragging

Everybody has already moved files in the Windows explorer and knows the automatical expanding of the folder structure: You move the file onto a collapsed folder, pause the mouse shortly, the folder is opened and you can move further until you have reached the desired folder.

> Behavior in older versions

Up to now, when moving a node vertically to another group in VARCHART XGantt, searching for the target group could take quite a bit of time, if the chart had many nodes in many expanded groups. In most cases, automatic vertical scrolling was needed to reach the target group, this sometimes being tedious and therefore uncomfortable.

> New: Easy orientation and fast vertical dragging

The new functionality considerably shortens the search for the target group. The combination and setting options being quite manifold, we'd like to confine ourselves to introducing one possible configuration here.

Example: Collapse all groups except the current one

One possible configuration of VARCHART XGantt might be that when moving a node, all groups but the one having just been touched get collapsed. The status of this group will be maintained, in case the node is to be moved within the same group only. By collapsing the other groups, the vertical extension of the plan is reduced to a fraction of its original size, thus allowing to show considerably more groups than before and ideally, the target group will be already visible by now. If not, VARCHART XGantt can automatically scroll over the collapsed groups so that the target group can be found much faster than before. On reaching the target group, one pauses a moment, the target group is expanded and the movement can go on. The group having been touched before gets collapsed so that the plan size remains minimized. The dragging goes on, perhaps to another group that is expanded, the group having been expanded before being collapsed again etc. until reaching the target. On releasing the node in the target group, the interaction is finished and, if desired, VARCHART XGantt can restore the original condition, scrolling to the new position of the moved node.

> Many combination options

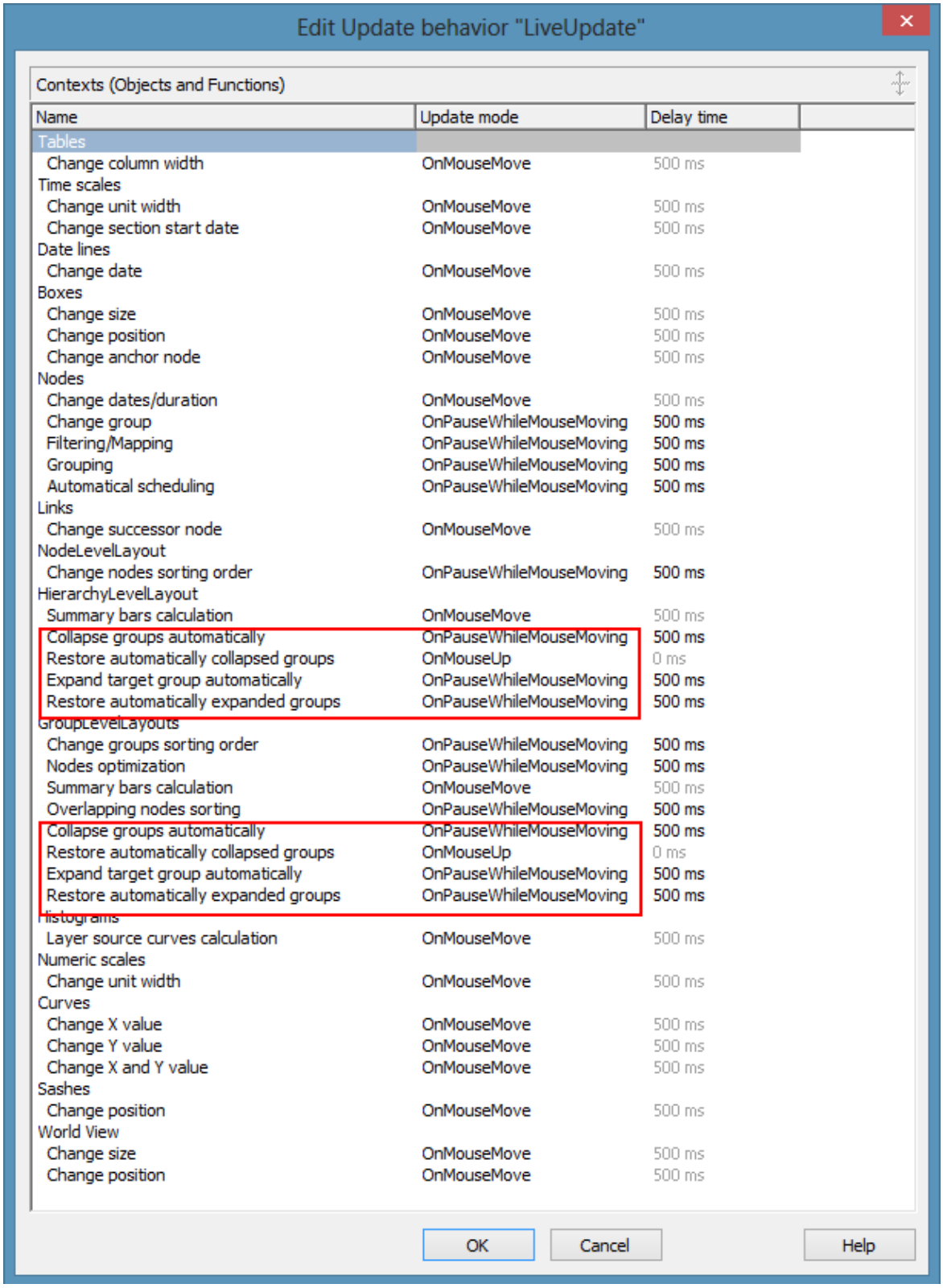
This was only one example of the new functionality. There are further options available for:

- Automatic collapsing of groups
- Automatic expanding of groups
- Automatic restoring of automatically collapsed or expanded groups, an update behavior allowing for a precise temporal control of this option.

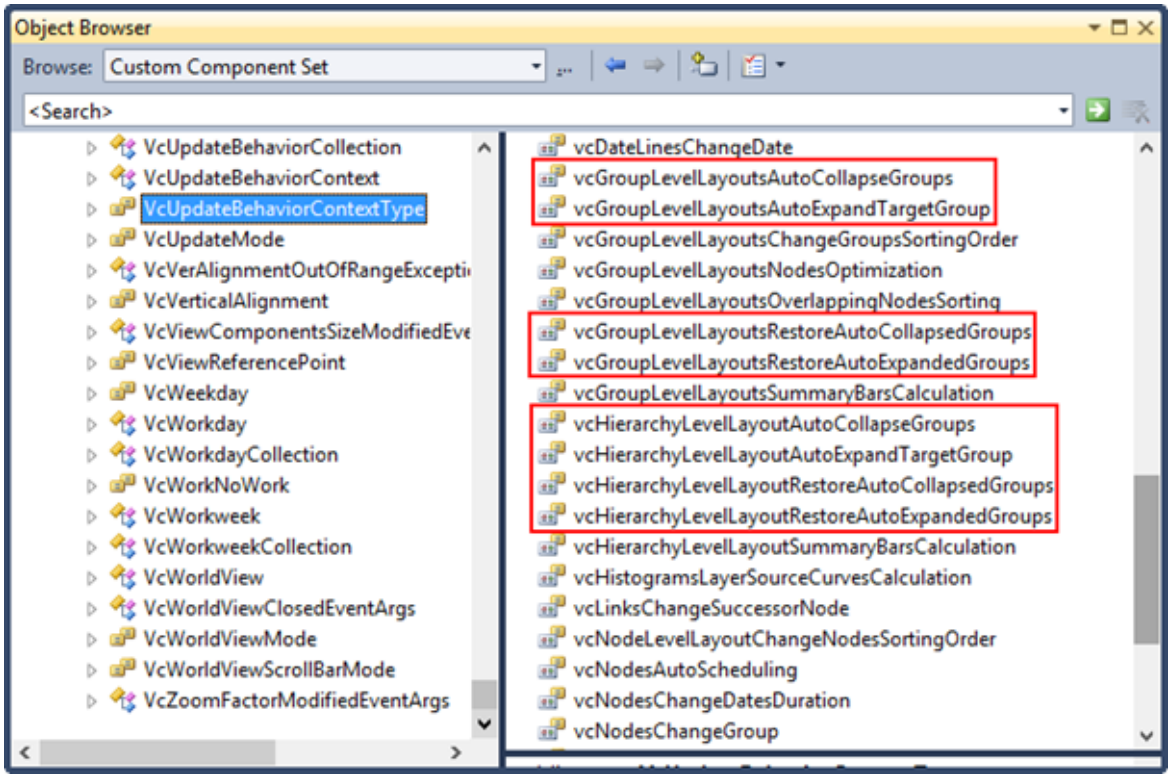
These settings can be made per grouping level and also for the hierarchical arrangement of the nodes, allowing for very detailed dragging operations.

> New properties and API calls

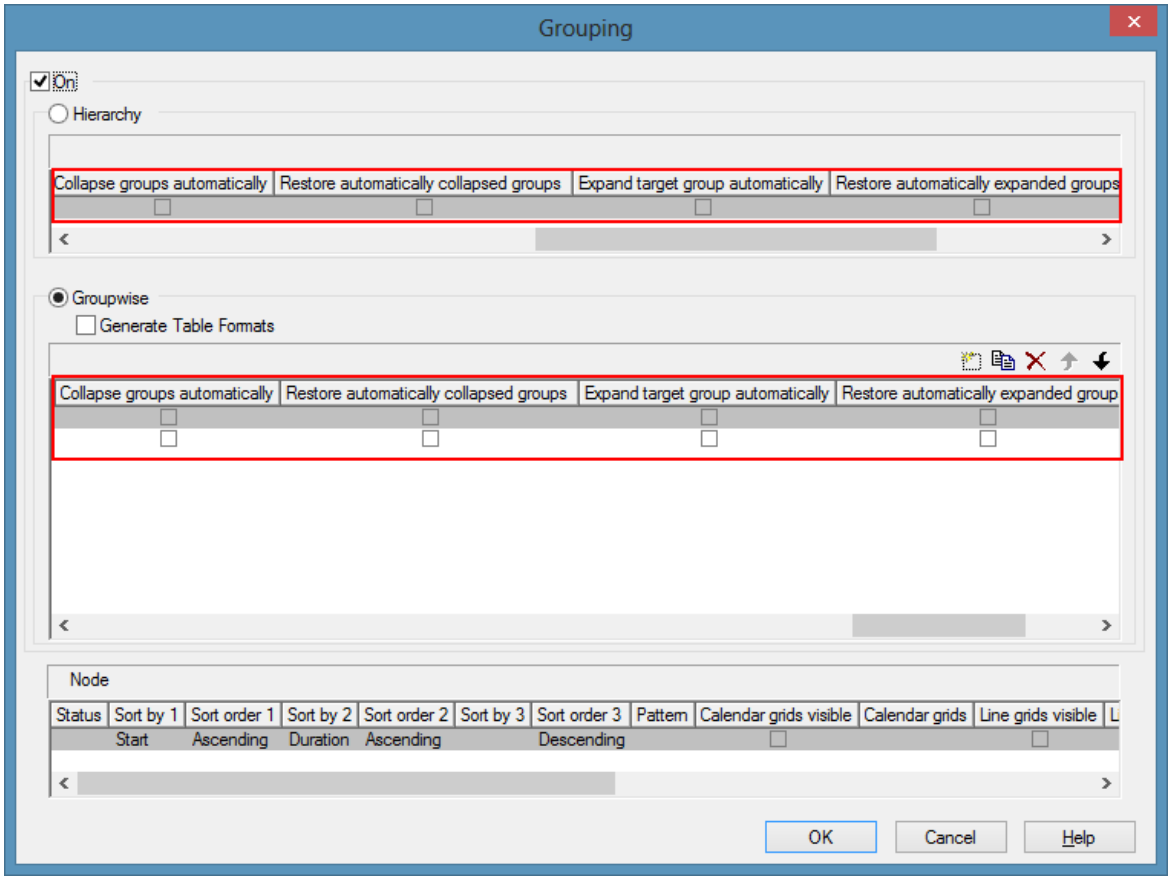
The **Edit Update behavior** dialog offers eight related contexts, four each in Grouping Line Layouts and Hierarchy Layout:



The enumeration VcUpdateBehaviorContextType has also got 8 new values so that the new contexts can also be set at runtime.



The functionalities that are activated by this contexts by way of timer can be enabled or disabled in the Grouping dialog.



API calls:

VcGroupLevelLayout.AutoCollapseGroups = true/false

VcGroupLevelLayout.AutoExpandTargetGroup = true/false

VcGroupLevelLayout.RestoreAutoCollapsedGroups = true/false

VcGroupLevelLayout.RestoreAutoExpandedGroups = true/false

VcHierarchyLevelLayout.AutoCollapseGroups= true/false

VcHierarchyLevelLayout.AutoExpandTargetGroup = true/false

VcHierarchyLevelLayout.RestoreAutoCollapsedGroups = true/false

VcHierarchyLevelLayout.RestoreAutoExpandedGroups = true/false

3.6 Events

Events are the elements that pass information on the user's interactions with the VARCHART ActiveX control to the application. Each time a user interacts with the VARCHART ActiveX control, for example by modifying data or clicking on somewhere in the control, a corresponding event is invoked. You can react to these events by the programming code of your application.

In all programming environments, functions which already contain the parameters provided by the control are supplied for events. Each event is described in detail by the API Reference Manual.

Note: By the **returnStatus** parameter of the events you can deactivate all context menus offered in the VARCHART ActiveX control (and replace them with your own, if you want) plus you can control all interactions and revoke them where required.

> Return Status

The below table contains the return status values of VARCHART ActiveX events:

Constant	value	description
vcRetStatDefault	2	default value
vcRetStatFalse	0	revoking the action
vcRetStatNoPopup	4	revoking the popup menu

3.7 Filters

A filter consists of conditions that are to be fulfilled by layers, histogram curves, links or table formats. Filters let you select layers, curves, links or table formats that fulfill the criteria defined, e.g. in order to highlight them in the diagram.

When you apply a filter, the data of the record is compared to the criteria of the filter. Layers, curves, links or table formats that fulfill the filter criteria will be selected.

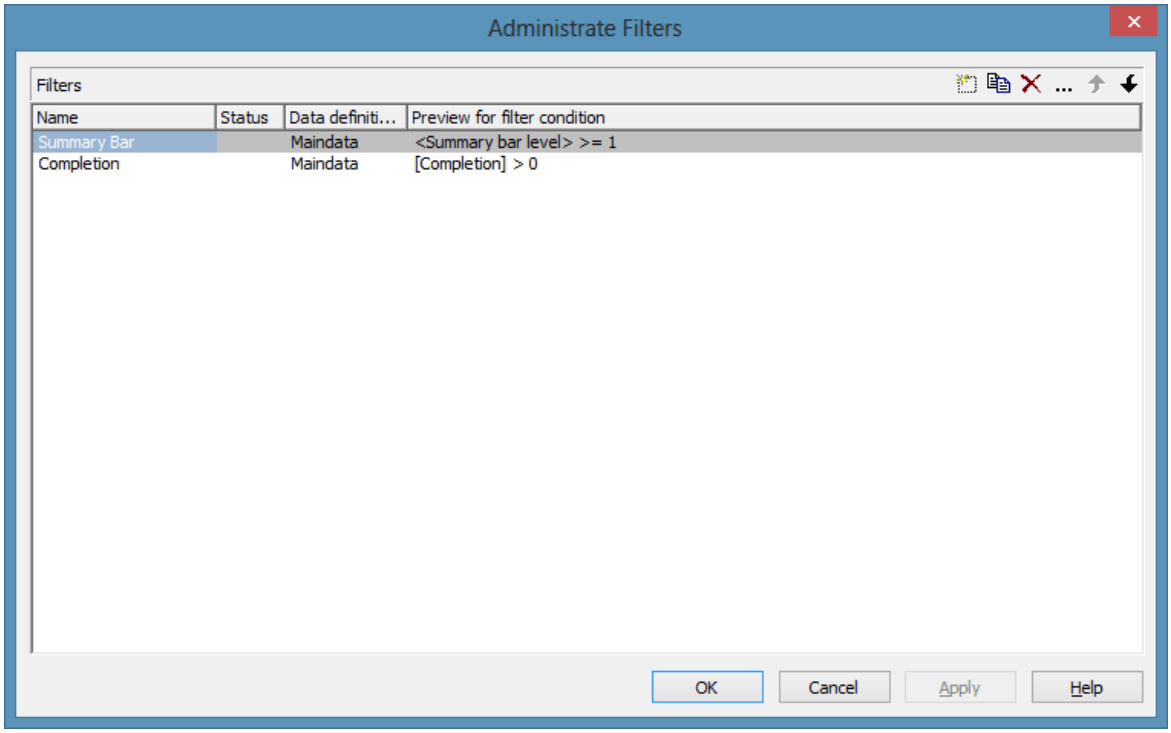
For example, you can create a filter that specifies "All activities starting after January 2010".

Filters can only be generated and configured in design mode.

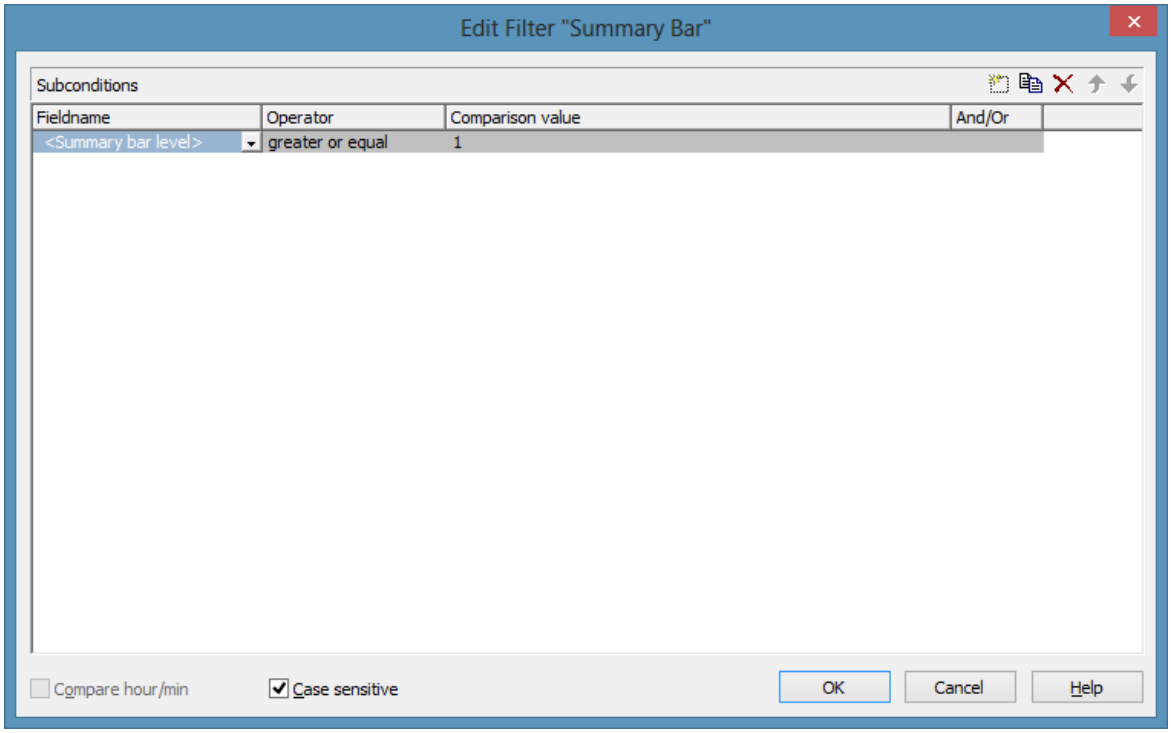
There are several ways to get to the **Administrate Filters** dialog box:

- on the **Objects** property page
- for layers: in the **Specify Bar Appearance** dialog box
- for table formats: in the **Edit Table** dialog box
- for links: in the **Filter** button of the **Link** property page
- for histogram curves: in the **Filter** combo box of the **Edit Histogram** dialog
- for nodes: by the **Filter** button of the **Nodes** property page.

Use the **Administer Filters** dialog box to rename, create, copy, delete or edit filters.



To edit a filter press the **Edit filter** button of the **Administrate Filters** dialog box. Then the **Edit Filter** dialog box will open.



3.8 Graphics Formats

VARCHART supports the below graphics formats, which is important to exporting charts, affecting mainly the calls **VcGantt1.ShowGraphics-ExportDialog** and **VcGantt1.ExportGraphics**.

The XGantt control supports both the import of graphics files e.g. for displaying in nodes or in boxes and the export of complete charts to graphics files. There is a connection beween the chosen (supported) graphics format and the graphic's display quality in the control (after the import) or in an external viewer program (after the export). Please find below a description of the advantages and restrictions of the individual graphics formats. Basically there are two different types:

Vector graphics formats store single geometrical figures such as lines, ellipses or rectangels as descriptions of the figure with corresponding parameters as start coordinates, dimension and color. Thus they are resolution-independent and lines are still displayed precisely, regardless of the zoom level. There is just one restriction concerning the size of the available coordinate space, especially with the WMF format. In general, the vector graphics formats' great advantage lies in their resolution independence and also often in the resulting file size. Unfortunately a platform-independent, standardized format has not established itself.

Bitmap graphics formats store pixels together with their color in a preset dimension. If the graphics are heavily zoomed in they automatically get "pixelly". To limit the file size, bitmap graphics are often compressed lossless or lossy even. A loss, however, can only be accepted with photos, not with diagrams. The only advantage that the bitmap graphics formats offer is the fact that they have become widely accepted via digital cameras and the internet and are widespread platform-independent.

> WMF (Windows Metafile Format)

This vector graphics format has been in existence since Windows 3.0. It internally consists of command data sets that correspond to the GDI commands of the Windows API. By them, the GDI commands can be persisted to all intents and purposes. Nevertheless, this format was incomplete already when it was developed. It had and today still has a limited coordinate space. Beside, it lacks clipping, transforming coordinates and filling complex polygons. The problem of the missing option to transform the "real" coordinates into inches and centimeters was encountered by the Aldus company already at an early stage. They developed the "Aldus Placeable Header" which for long has been recognized and used by virtually all

programs that display and use WMF files, except for the Windows API itself, which up to now is unable to generate or process the header, although it is mentioned and explained in the Microsoft documentation.

When Microsoft released Windows NT and 95, the WMF format became dispensable and its successor called EMF entered the market. Still, WMF is quite popular up to now, especially with ClipArt graphics that do not require the extended options of the successor format. The innovations of Windows 95 and NT have not been not transferred to the format, it has remained unchanged since.

In WMF, a comment data set is available which can be used to place EMF commands. If a display program discovers those kinds of comments, i.e. if it can display EMF files, it automatically will discard the WMF command data sets and will display the EMF command data sets instead. Thus a single file can contain a WMF graphics as well as an EMF graphics. Presumably, this was implemented for reasons of compatibility, but it inflates the file size considerably.

For the description of the format please see:

http://msdn.microsoft.com/en-us/library/cc215212.aspx

On the limitations of the format see:

http://support.microsoft.com/kb/81497/en-us

> EMF (Enhanced Metafile Format)

This vector graphics format was introduced simultaneously with the 32bit operation systems Windows NT and 95. It suspends the limitations imposed by the WMF format and internally consists of graphics commands that correspond to the GDI32 commands of the Windows API. The coordinates' space is 32 bits large, transformation and clipping are supported. The commands of masking and alpha-blending equipped blitting of storage bitmaps added to GDI32 later on are not supported though.

In spite of the advantages that it features compared to WMF, the format has remained largely unknown, although all display programs and Office packages can handle EMF.

A disadvantage when using GDI+ is that some of the new GDI+ graphical features such as color gradients and transparencies are not fully supported. In addition, when exporting the chart to an EMF file, discontinuous lines (for example dashed ones) are stored as a set of short, continued lines, which on one hand increases storage demand and on the other hand consumes more time when the file is loaded.

EMF also offers a comment data set that can be used to place EMF+ commands. If a display program discovers those kinds of comments, i.e. if it can display EMF+ files, it automatically will discard the EMF command data sets and will display the EMF+ command data sets instead. Thus a single file can contain a EMF graphics as well as an EMF+ graphics. Presumably, this was implemented for reasons of compatibility, but it inflates the file size considerably.

By the way, if required, printing jobs in Windows internally are cached as EMF data streams and passed to the printer driver.

For the format description please see:

http://msdn.microsoft.com/ en-us/library/cc204166.aspx

> EMF+ (Enhanced Metafile Format Plus)

Although the name suggests this format to be an extension of EMF, it is a vector graphics format of ist own which was introduced simultaneously with the GDI+ Windows API. Internally, it consists of graphics command data sets that correspond to the GDI+ commands. By the way, GDI+ is not an extension of the GDI API, but a graphics library of its own. In addition to EMF also transparencies and color gradients are completely supported.

Up to now the format has remained quite unknown and quite often ist not supported by the common display programs, except by Microsoft Office from 2003 onward. Microsoft has published the structure of the EMF+ format only in 2007.

For the format description please see:

http://msdn.microsoft.com/ en-us/library/cc204376.aspx

> GIF (Graphics Interchange Format)

This bitmap format was developed by CompuServe for a lossless, compressed storage of graphics files before the World Wide Web came into existence. It can only display 256 colors simultaneously and is therefore unable to store today's graphics files reasonably. This format is only supported for reasons of compatibility.

The subformat "Animated GIF" is not supported at all.

> JPEG (Joint Photographic Experts Group)

This bitmap format was developed by the JPEG for compressed storage of photographs, accepting loss. Storing charts and diagrams requires a precise

storage of lines, so using this format does not make much sense. This format is only supported by the VARCHART products for reasons of compatibility.

> BMP (Windows Bitmap)

This bitmap format was developed by Microsoft for a lossless, uncompressed storage of graphics files. Internally, the format is used directly in the memory of the Windows API GDI. A restraint is given by this format not supporting the alpha channel, so merely 24 bits per pixel can be stored. Due to its high memory demand this format should be abandoned. It is only supported by the VARCHART products for reasons of compatibility.

> TIFF (Tagged Image File Format)

This bitmap format was developed by Aldus (merged into ADOBE) for a lossless, uncompressed storage of graphics files. Graphics files can be stored with or without loss. The format has not been enhanced for quite some time. It is only supported by the VARCHART products for reasons of compatibility.

> PNG (Portable Network Graphics)

This bitmap format was developed by the World Wide Web Consortium (W3C) for a lossless, compressed storage of graphics files to replace the copyright-afflicted and limited GIF format. PNG is brilliantly qualified to store VARCHART charts; transparent elements are actually drawn as such. It is universally used by virtually all display programs and internet browsers. The format itself is free of copyrights and completely documented.

From version 4.2 onward the free library **libpng** is used which is freely available, in order to set a resolution and thus store bitmaps of any size. It has to be taken into account though that very large PNG files may cause problems when loaded, since usually PNG files get completely unpacked in the memory and then are displayed.

For the format description please see:

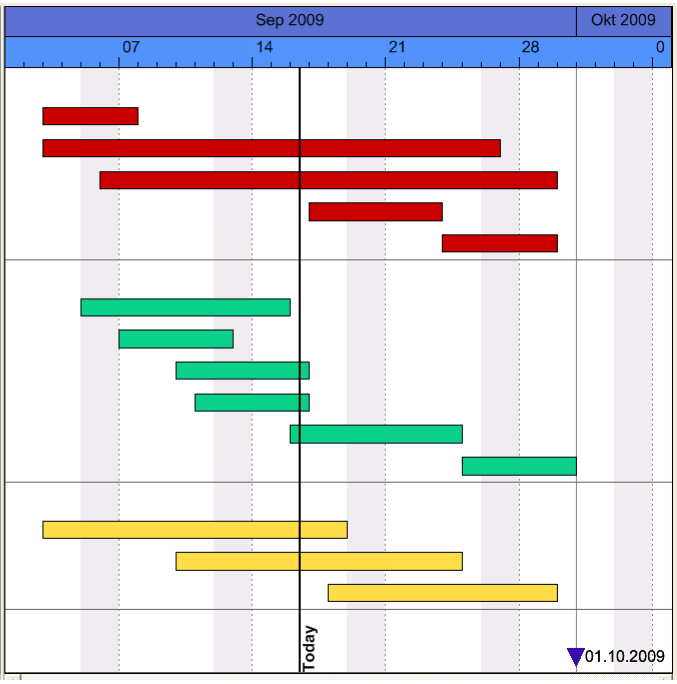
http://www.libpng.org/pub/png/spec/1.1/PNG-Contents.html

3.9 Grouping

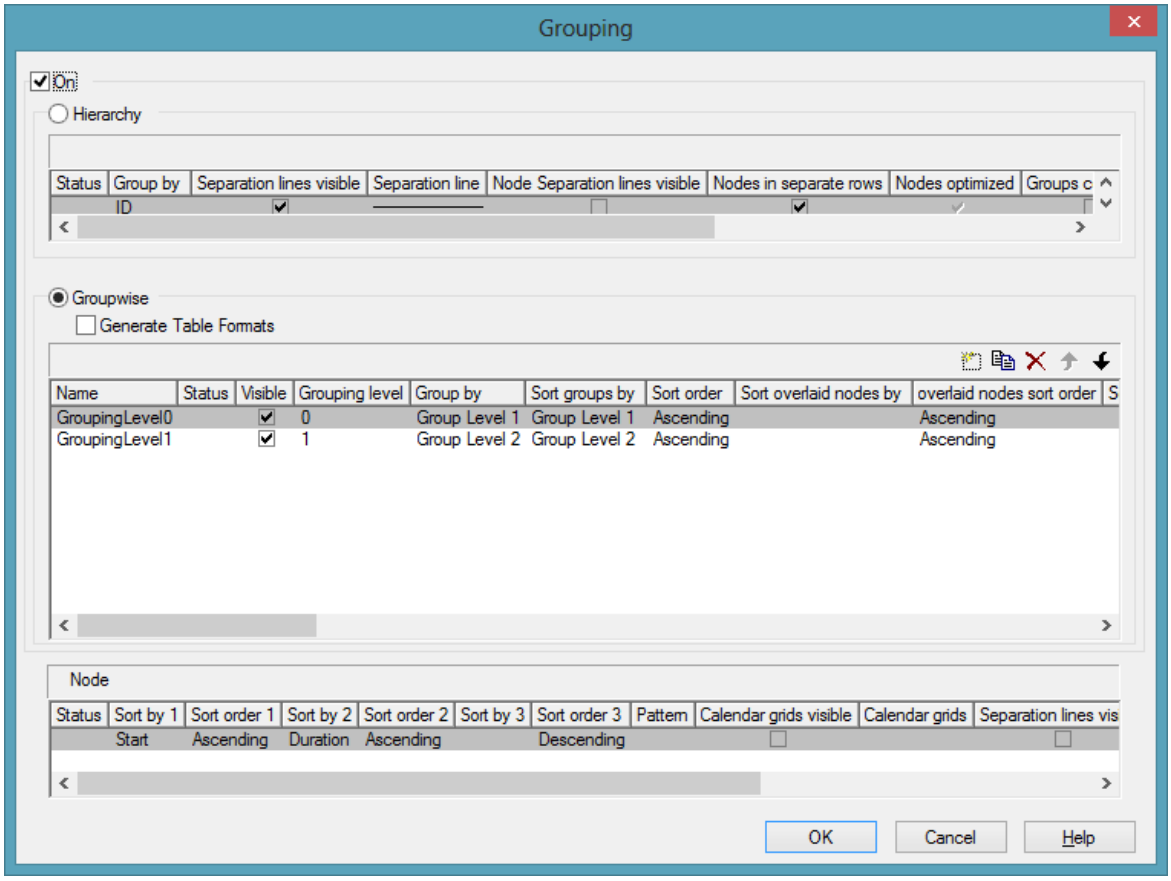
It often is necessary to split activities into groups and then visually emphasize the groups in your diagram. For example, activities are frequently grouped by project phases (e.g. planning, construction, manufacturing, etc.) or by departments (Construction Dept., Accounts Dept., etc.).

A grouped diagram could look something like this:

ID	Name	Start	End
Group A			
3	Requirements	03.09.2009	08.09.2009
1	SW Development	03.09.2009	27.09.2009
13	Sales & Marketing	06.09.2009	30.09.2009
7	Final Check	17.09.2009	24.09.2009
12	QA Requirement Check	24.09.2009	30.09.2009
Group B			
4	GUI Design	05.09.2009	16.09.2009
5	Class design	07.09.2009	13.09.2009
9	Phase A (DB)	10.09.2009	17.09.2009
6	Data structure	11.09.2009	17.09.2009
10	Phase B (GUI)	16.09.2009	25.09.2009
14	Delivery	25.09.2009	01.10.2009
Group C			
2	Design&Concept	03.09.2009	19.09.2009
8	Coding	10.09.2009	25.09.2009
11	Testing	18.09.2009	30.09.2009
Group E			
15	Final Party	01.10.2009	01.10.2009



Groups are formed by a value, that all members of a group have in common. Nodes that show the same entry in their grouping data field belong to the same group. The grouping field and all other grouping criteria can be set in the corresponding dialog which you can open by clicking the **Grouping** button on the **Objects** property page.



Activities that have the same value in the **Group by** data field will be allocated to the same group.

In the diagram, an extra row above the group contains the group title. The appearance of the group title can be defined individually in the **Edit Table Format** dialog box, depending on whether the groups are expanded or collapsed (table formats **Subtitle** and **Collapsed**), e.g. by using different colors or data fields.

The small plus or minus symbol next to the group headings indicates whether the associated group is collapsed or expanded. By clicking on the sign, you can switch from the collapsed status to the expanded status and vice versa. To enable the feature, the **Modifications allowed** check box in the **Grouping** dialog has to be ticked.

You can use the **Sort groups by** and the **Sort order** options to set the order of the groups.

More options can be selected for groups:

- whether **table formats** are to be generated
- a **pattern** for the title row of the group (only in the diagram)
- display and style of **calendar** and **line grids**

- whether all activities of a group should be displayed in a single row or not (switching on/off the option **Nodes in separate rows**) and, if so, whether the node layout should be optimized automatically (**Optimized**)
- whether the groups should be collapsed when starting the program (**Groups collapsed**)
- display and style of **Separation lines**
- whether the collapse/expand function (**Modifications allowed**) should be available to the user
- whether summary bars are to be displayed (**Summary Bar**)
- whether **Group nodes** are to be displayed
- whether the **order of groups** can be changed by drag interactions in the diagram and/or the table
- whether **page breaks** are to be carried out after each group

> Creating Groups Interactively

Each time a new node is created interactively in an empty chart, a group node will be created automatically. In the **Edit Data** dialog you can enter a group name into the data field that was selected for **Group by** in the **Grouping** dialog.

If you want to create a new group, please proceed as follows: Create a node in an existing group. Double-click on the node to open the **Edit Data** dialog box. Then enter a group name into the data field that has been selected for **Group by** in the **Grouping** dialog. Then the new group will be created.

> Regrouping Nodes Interactively

If a user drags an activity from one group to another one, the value in the grouping field will be adapted automatically.

> Empty Groups

If you delete all nodes of a group, the title of the group will remain in the table. Only if you switch the grouping off and on again or end and re-start the program, the titles of all empty groups will not be displayed any more.

> Moving subgroups interactively

You can modify the sorting order of subgroups interactively. To do so, please mark the summary bar of the subgroup which you want to move. Then drag the phantom of the subgroup to a place of your choice within the diagram. If you place the phantom onto a different summary bar of the same grouping

level, an arrow will indicate whether you can insert the summary bar above or below it. When releasing the mouse button, the group and its subordinated nodes will be inserted in the selected place.

> All nodes of all groups in one line/in separate lines/expanded/collapsed

By a few lines of code you can control in which way the nodes of groups are to be displayed. In the below example the nodes of two different grouping levels will be displayed in a single line by a menu call.

Example Code

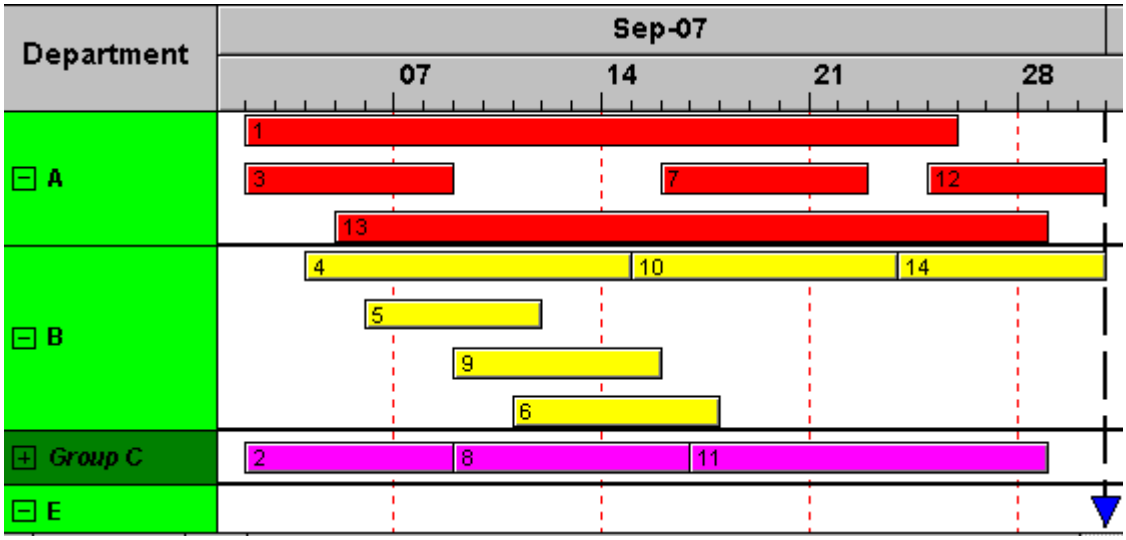
```
Private Sub mnuAllNodesOneRow_Click()

Set groupcollection = VcGantt1.groupcollection

For Each group In groupcollection
    Set subgroupcollection = group.SubGroups
    group.AllNodesInOneRow = True
    For Each subgroup In subgroupcollection
        subgroup.AllNodesInOneRow = True
    Next
Next

End Sub
```

In a similar way you can display the nodes of groups in separate rows for each group (group.AllNodesInOneRow = False), expand them (group.Collapsed = False) or collapse them (group.Collapsed = True).

> Diagram with Grouping Option "Nodes in One Line"

This section gives a brief description of the **Nodes in separate rows** option for the group layout of the activities.

A diagram with this option enabled may look like the below sample:



The grouping procedure is the same as previously described, where each activity was displayed in a separate line. If the **Nodes in separate rows** option of the **Grouping** dialog was not set, a whole group is displayed in one row. Naturally, the activities may overlap within the row. In order to make overlays visible, the group can be expanded, which means that, strictly speaking, the option should be called "In as few lines as possible". In their expanded state, you are free to move overlapping activities until all overlays have gone. Thus an expanded diagram ensures that overlapping activities (even if they do so for only a second) can instantly be recognized.

When a group is collapsed (as is Group C in the example), it shows that it comprises several activities, but there is no way to recognize whether there are overlays.

Naturally, with this type of diagram, it makes no sense to arrange the activities in a table format. Therefore, we recommend to display annotations on layers instead or to use tooltips for their identification.

> Displaying Overlaying Nodes

If the **Nodes in separate rows** mode was not selected, the sorting order will determine which nodes are drawn last and therefore are completley visible, in case they overlap.

> Summary bar

In group lines, summary bars can be displayed. You can specify whether and for what grouping levels summary bars are to be displayed.

To display summary bars at grouping levels defined by **Grouping level**, in the **Grouping** dialog, the check box **Summary Bar** needs to be ticked for the corresponding level.

The VcGantt property **SummaryBarsVisible** at run time lets you set or retrieve, whether summary bars are visible. If grouping is a true grouping (not a hierarchy), you can switch on or off the summary bars by levels, using the parameter **GroupingLevel**.

On the **Layer** property page you can specify the appearance of summary bars by creating layers that display any desired shape. You may define one layer for all or for some levels, as well as a different layer for each level, e. g. the layer "Summary bar 1" for the first level, "Summary bar 2" for the second level etc.

To display the desired shapes of summary bars, filters need to be assigned to them to select for defined features at each level. Filters can be created in the **Administer Filters** dialog, e.g. the filter "Summary bar 1" for the first level.

To select a level to which the filter conditions apply, please invoke the **Edit Filter** dialog. In the column **Field name** select **summary bar-level**, select an **Operator** (equal, greater or equal, greater than, etc.) and enter the desired level number in the **Comparison** field.

When you start the program, the specified summary bars will be displayed.

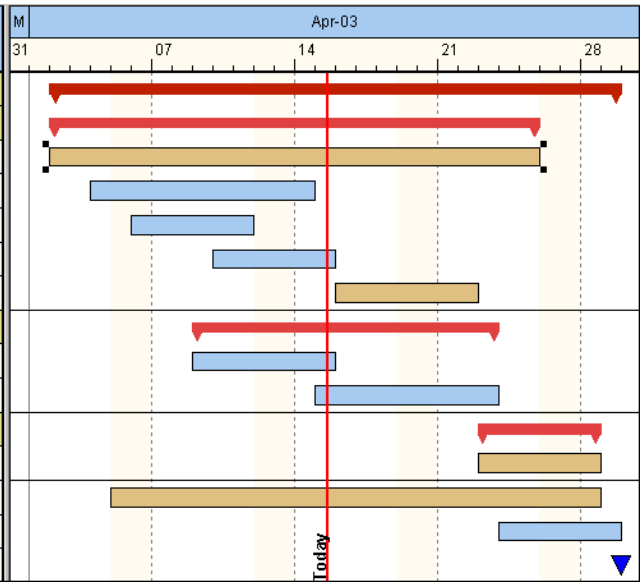
3.10 Hierarchical Order

An alternative way of arranging activities by levels is to use a hierarchy. For a hierarchical order the project data has to contain a hierarchy code of the format:

1., 1.1, 1.1.1, 1.2, 1.2.1, ...

A hierarchical layout could look something like this:

Structure	Name	Start	End	ID
⊟ 1	*SW Development*	*02.04.03*	*30.04.03*	*1*
⊟ 1.2	*Design&Concept*	*02.04.03*	*26.04.03*	*2*
1.2.1	Requirements	02.04.03	26.04.03	3
1.2.2	GUI Design	04.04.03	15.04.03	4
1.2.3	Class design	06.04.03	12.04.03	5
1.2.4	Data structure	10.04.03	16.04.03	6
1.2.5	Final Check	16.04.03	23.04.03	7
⊟ 1.3	*Coding*	*09.04.03*	*24.04.03*	*8*
1.3.1	Phase A (DB)	09.04.03	16.04.03	9
1.3.2	Phase B (GUI)	15.04.03	24.04.03	10
⊟ 1.4	*Testing*	*23.04.03*	*29.04.03*	*11*
1.4.1	QA Requirement C	23.04.03	29.04.03	12
1.5	Sales & Marketing	05.04.03	29.04.03	13
1.6	Delivery	24.04.03	30.04.03	14
1.7	Final Party	30.04.03	30.04.03	15

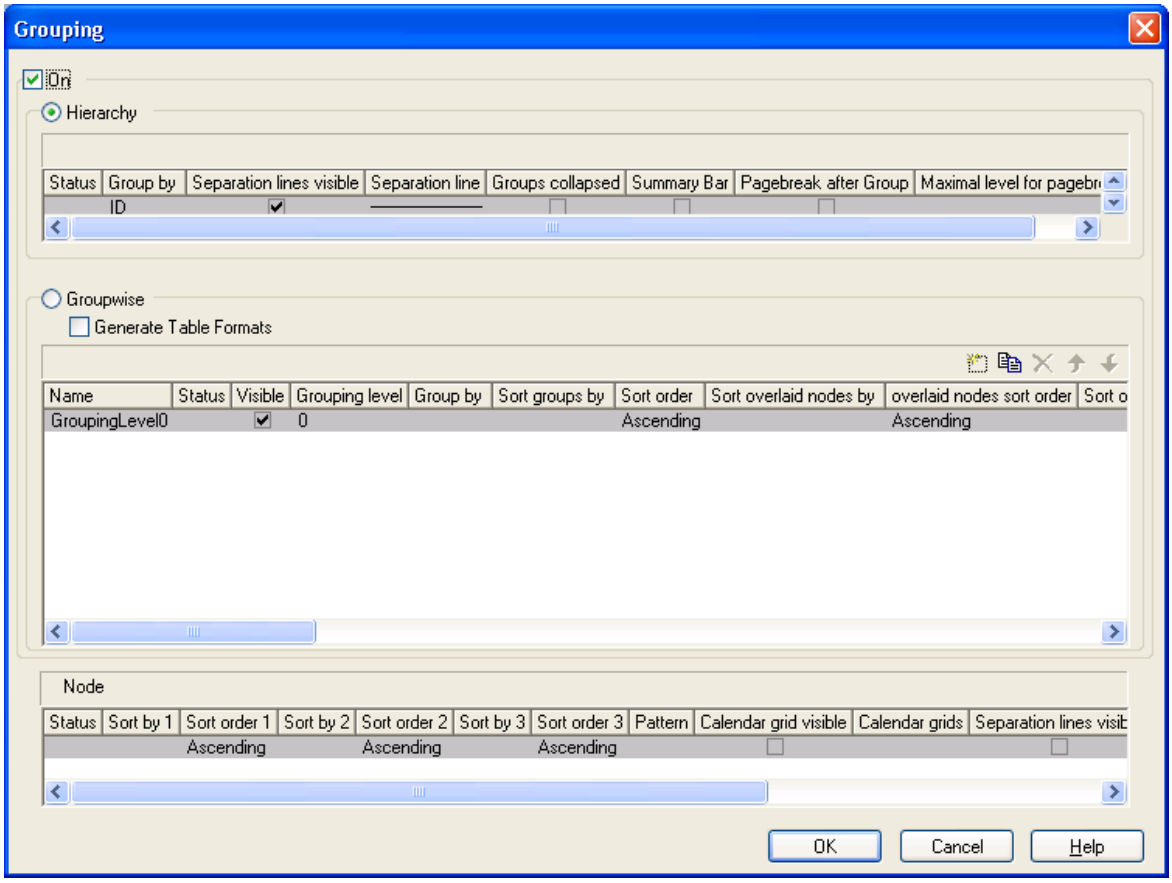


The symbols + and - are automatically displayed in front of the superordinate activities. Sublevels are indented automatically. By clicking on the - symbol, the structure of subordinate activities will fold (collapse); by clicking on the + symbol it will unfold (expand).

The program does not check whether the dates of the superordinate activities comprehend the dates of the subordinate ones, i.e. the program does not verify or set activity durations.

If the hierarchical order is selected, no other grouping or sorting option can be set.

A hierarchical arrangement can be set in the **Grouping** dialog:



To apply a hierarchical order, the check box **Hierarchy** needs to be ticked. After this, a data field that contains the structure code has to be selected from the combo box (**Group by**).

In addition, the below hierarchy features can be set:

- Display and style of **Separation lines**
- whether the activities should be collapsed on the start of the program (**Groups collapsed**)
- whether summary bars are to be displayed (**Summary Bar**)
- whether **page breaks** are to be carried out after each group and up to which level they are to be carried out

The table formats **Hierarchy** and **HierarchyCollapsed** are used to display the summary activities. They can be modified in the **Edit Table Format** dialog.

> Moving nodes interactively

You can move nodes interactively. The node moved will be inserted before or behind the reference node, also in collapsed groups.

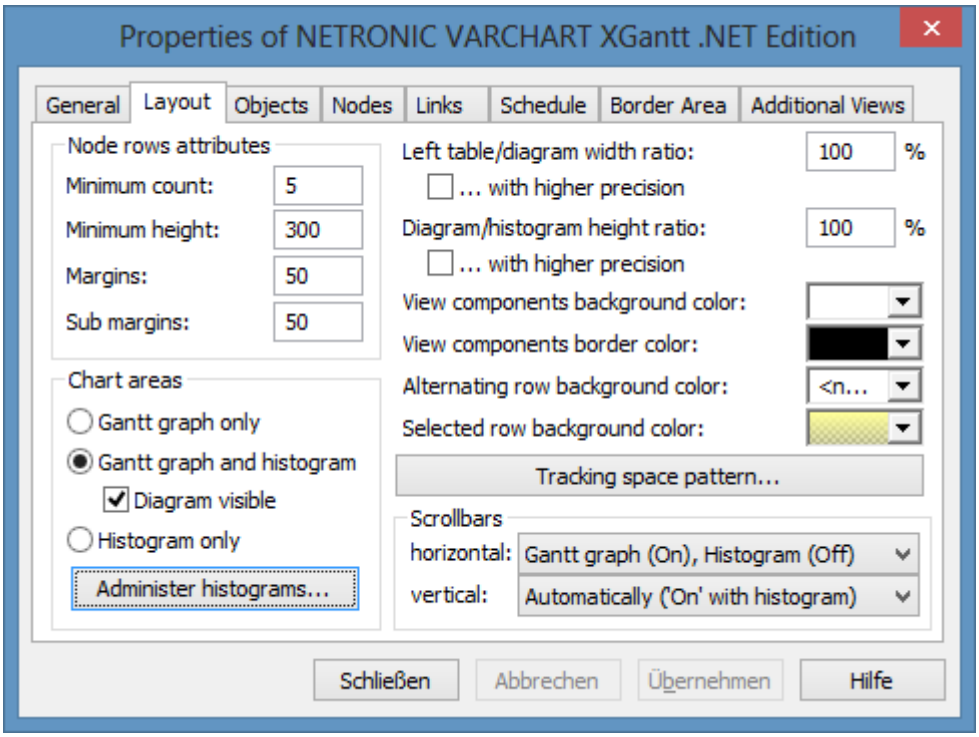
> Moving summary bars interactively

Summary bars can be moved interactively in the same way as nodes. The nodes subordinated to the summary bar will be moved simultaneously.

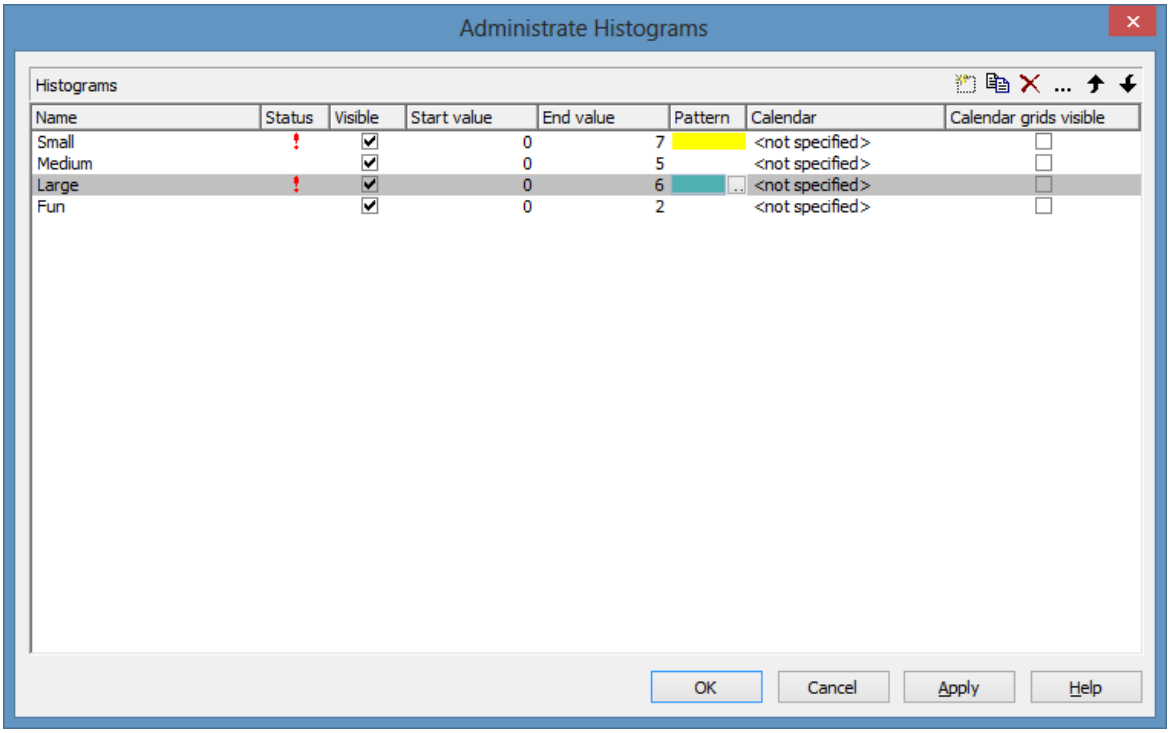
3.11 Histograms

Histograms are used to add up activities to curves above the time axis, with the activities fulfilling certain criteria.

On the **Layout** property page you can set whether just a Gantt chart, just a histogram or both, a Gantt chart and a histogram should be displayed.



To select the histograms to be displayed and to edit histograms, please click on the **Administer Histograms** button. The below dialog will appear:

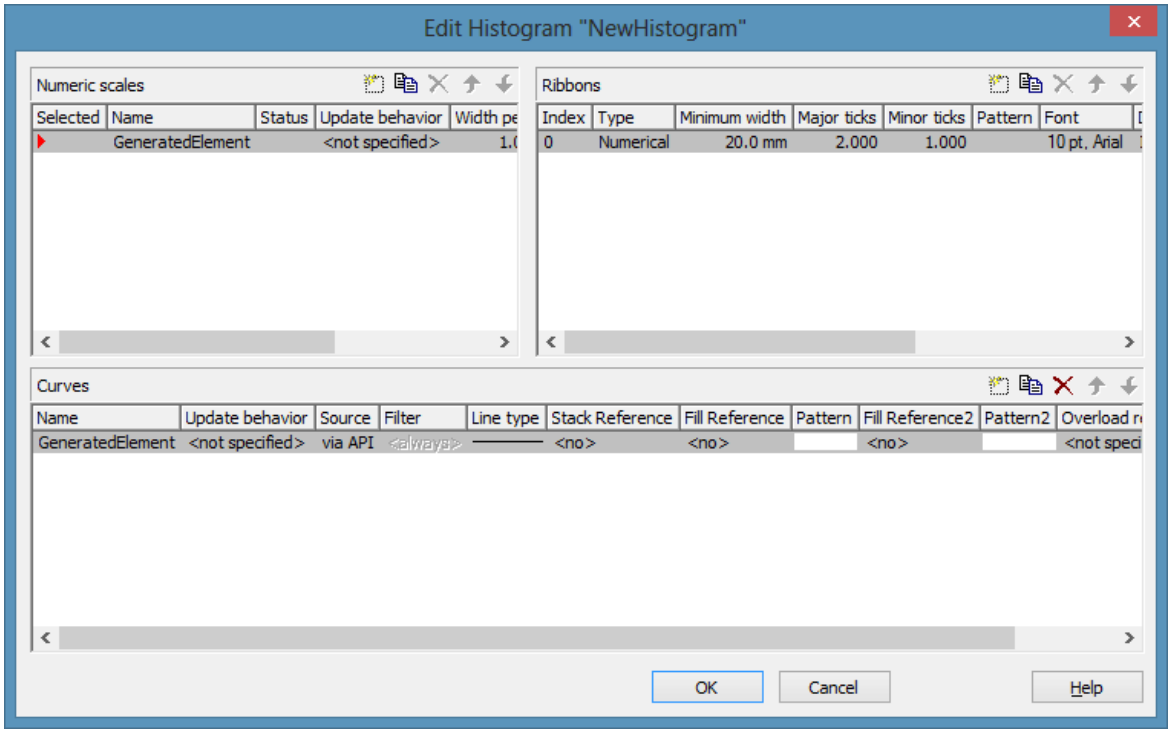


On this page you can select one or more histograms to be displayed.

A histogram comprises a numeric scale (y axis) and curves; the Gantt chart timescale serves as the x axis.

To each histogram, you can specify the start and the end value of the numeric scale separately.

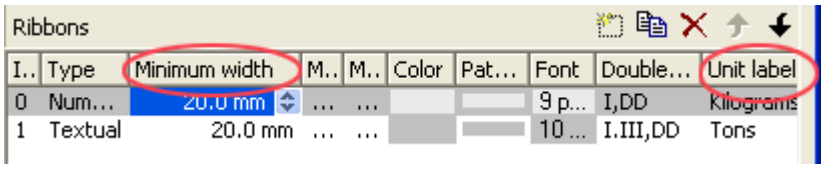
To edit a histogram, mark it and click onto the **Edit** button (···). The below dialog will appear:



> Numeric scales

In the above dialog you can define different numeric scales and select one to apply to the histogram. You can define the grading of a numeric scale in y direction (**Width per unit**). Beside, you can decide whether a line grid is to be displayed and define ist features.

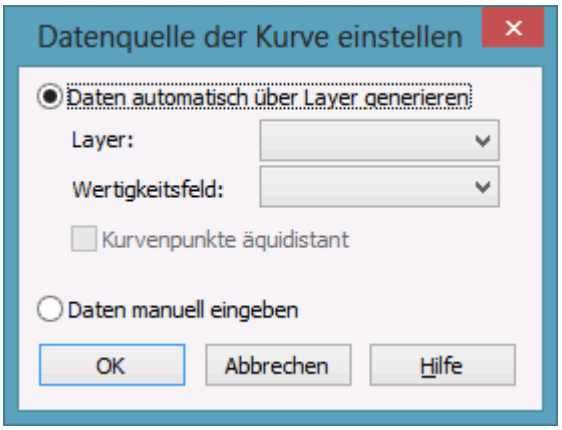
In the **Ribbons** area you can assign one or more ribbons to the numeric scale being edited. To each ribbon you can set a **Type**, a **Minimum width**, a number to define after how many units a **Major** or a **Minor tick** should occur, you can assign a background **Color**, **Font** features and a **Double format**. Furthermore you can tick the option **Object draw events** if you want to design the contents of the ribbon by yourself and you can specify a **Unit label** to designate the units used in the ribbon. For the unit label, please ensure that sufficient space is provided by the minimum width of the ribbon; otherwise the label cannot be displayed and will remain invisible.



> Histogram curves

A histogram may contain several capacity curves, for which you can individually define a name, the line type and a pattern to be displayed below or above the curve line. A curve requires a source to be specified to supply

the curve data. To set the source of a curve, please click on the **Source** field and then on the **Edit** button (···). The below dialog will appear:



You can choose between two basic alternatives:

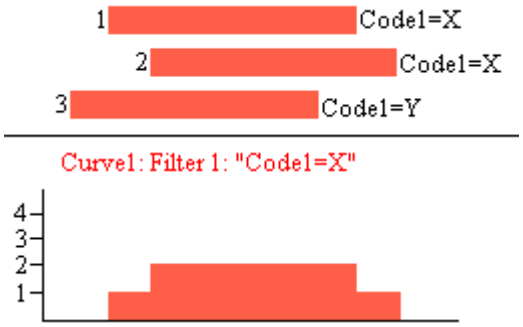
> 1. Data generated by layer

The curves are generated from the activities. When the activities are added up to a curve, the start and end dates of the selected layers (e.g. the "Start-End" layer) of each activity are picked up by the curve.

If the curve is generated from layers, in the **Edit Histogram** dialog you can select the activities that compose the curve by setting a **Filter** to the curve.

Example:

Only those activities that fulfill the conditions of Filter1 add to Curve1. Filter1 contains the expression **Code1 = X**, i.e. only the activities 1 and 2 to which **Code1 = X** applies, contribute to Curve 1.



For curves generated by layers, you can select the data field from which for each activity the valency for the capacity sum is to be taken (**Valency field**).

> 2. Data specified manually (via API)

This option allows to set the values by the API. Here you can freely define the values of a histogram curve by the VcCurve method **SetValues**.

For curves generated by the API, in the **Select curve data source** dialog you can set whether the curve points are to be created at equal distances (**Curve points equidistant**). Alternatively, curve points can be created only in points where y values change.

Curve points equidistant: Specify the start value (**startDate**) and the y values of the histogram curve. The coordinates of the histogram curve are calculated from the start value, combined with the **Time Unit** and **Smallest time interval** (Property page **General**).

Set Values X, Y1, Y2, Y3, ...

Curves generated in this way cannot be edited interactively.

Curve points not equidistant: Pairs of x and y values need to be specified:

Set Values X1, Y1

Set Values X2, Y2

Set Values X3, Y3...

The **Time Unit** and **Smallest time interval** do not play any part in this type of curve. This curve can be edited interactively.

> Reference curve

A curve defined by the API can be used as a reference curve to display the availability, for example.

> Stacking Curves

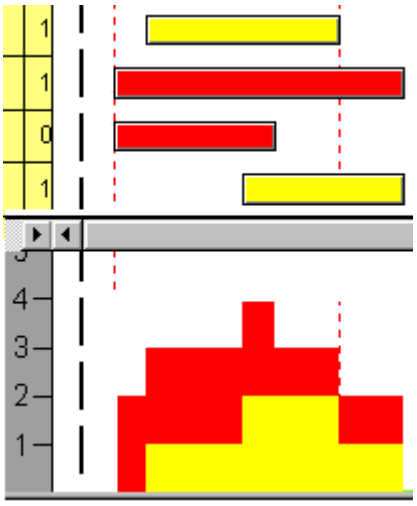
Stacking curves is useful, for instance, to visualize the total occupation of resources. Stacking curves implies all y values of an x value to be added up. To stack curves, filters need to be defined to select for activities that occupy certain resources.

To stack curves, for each curve, in the **Stack Reference** field specify the curve on which the edited curve is to be stacked. If you do not want to stack a curve, select the entry **No**.

If you set **No** to all curves, they may overlap each other. In order to make them differ, assign different line attributes to them.

Two curves can form the delimiters of a **fill area**, to which you can assign a color and a pattern. The **Fill Reference** field allows to set a reference curve to the curve being edited to form a fill area. The reference curve may be a curve or the x axis (**Flatline**).

A fill area may hide other curves, so therefore appropriate drawing priorities need to be assigned to curves.



Name	Stack Reference	Fill Reference	Pattern
Curve1	<No>	<No>	
Curve2	Curve1	Curve1	

Curve2 is stacked on Curve1.

If you do not want the curve line to be displayed when stacking curves, in the API set the VcCurve property **LineType** to **vcNone** or leave the **Type** field in the **Line Attributes** dialog box empty.

The curve line and fill pattern between curves are set in the **Line type** and **Pattern** fields, respectively.

If you click on the entry in the **Line type** field, the **Line Attributes** dialog box will appear, where you can define the color, thickness and type of a curve line. If you click on the **Pattern** field, the **Pattern Attributes** dialog box will appear, where you can define a pattern and the foreground and background colors for the fill pattern below a curve.

You can specify a second reference curve, if you activate the **2nd Ref.** box.

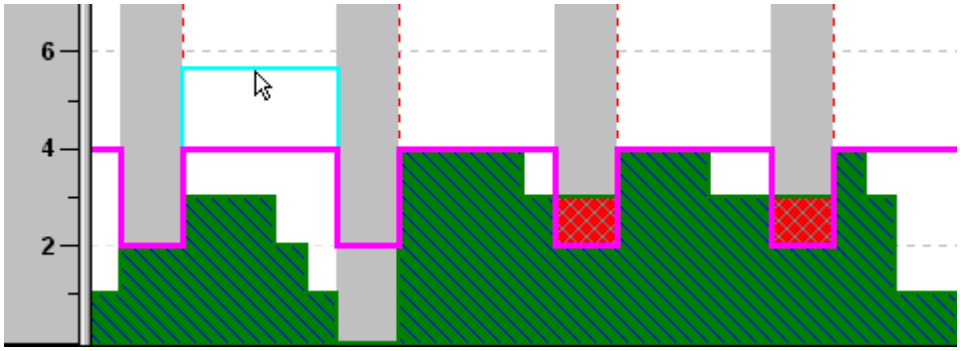
In the **Fill Reference2** field, select the second reference curve. The filling below the second reference curve is displayed only if the y values of the curve being edited are higher than the y values of the second reference curve.

In the 2nd **Pattern** field, specify the pattern and the color of the filling above the second reference curve.

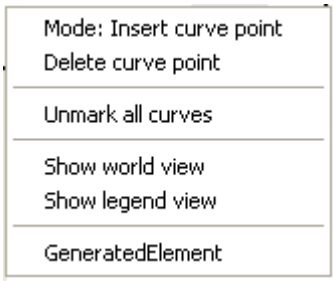
Examples of handling histograms you can find in "Tutorial: Creating Histograms".

> Interactive modification of non-equidistant availability curves

Modifications of available capacities can be set interacitvely. Non-equidistant curves (availability curves) that were generated by the API can be dragged upward or downward. A phantom supports the user's interaction by anticipating the new position.



You can add or delete single curve points interactively. To do so, please press the right mouse button in the histogram area. The below context menu will appear:



If several availability curves were defined, their names will be indicated in the context menu. If you click on a curve name, the corresponding curve will be marked.

Select **Mode: Insert curve point** and click on the availability curve by using the left mouse button. Each click succeeding will add a curve point.

To delete a curve point, click on it using the right mouse button and select the option **Delete curve point** in the context menu.

> Marking curve points

If you click on a non-equidistant curve, the curve points set up by the API will be marked by small black squares. By clicking again on the histogram curve you can make the curve points disappear.

3.12 How to Use a Calendar

A calendar represents a gapless sequence of working and non-working times. In a calendar that has a variable profile (shift calendar) different periods succeed repeatedly, such as morning, late or night shifts. A calendar itself has no visual appearance, it merely is the logic differentiation of working and non-working times. A calendar can become visible only if assigned to a **CalendarGrid** object.

In VARCHART XGantt a calendar also serves to derive start and end dates of nodes from durations. If no other option is set, a pre-defined base calendar named **BaseCalendar** is used for all calculations. In the base calendar the days Monday to Friday are defined as working periods, while Sunday and Saturday are free of work. The base calendar can be modified if required.

Defining a Calendar

A calendar can be defined at design time by the property pages or at runtime by the application programming interface (API). In this chapter we explain the basic handling of calendars from a developer's point of view and give some programming samples in C#. Defining a calendar by property pages is described in detail by the chapter **Property Pages and Dialog Fields**.

In the **VcGantt** control, an object **VcCalendarCollection** exists which takes care of the administration of all calendars. It has similar administrative functions as other collections have in VARCHART XGantt. The pre-defined **BaseCalendar** and any other calendar created at design time automatically form a part of the collection.

A new calendar can be created by the method **Add** of the **Calendar-Collection** object. The method requires a unique name for a calendar to be identified. Initially, a new calendar merely consists of working time.

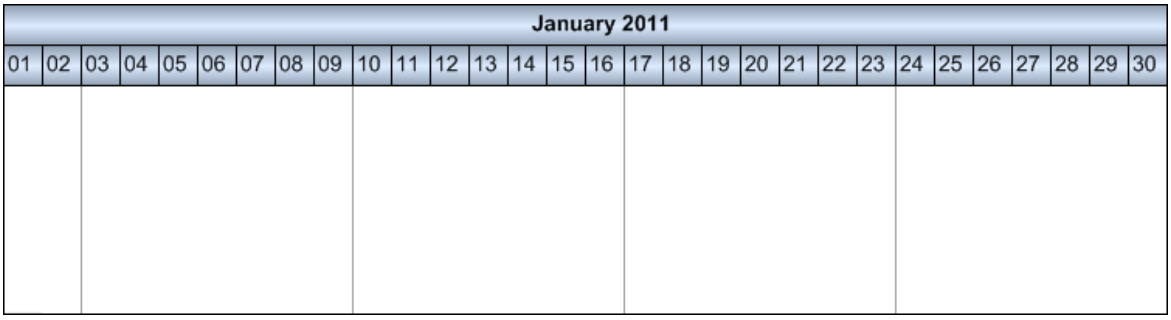
Please note: A calendar must contain at least a single time interval, since a calendar containing but non-working time cannot exist.

To make the results of our programming samples verifiable in the pictures of the Gantt diagrams, a constant time period is defined from 1.1.2011 to 31.12.2011 for the time scale in the programming samples. A calendar can only become visible in the background of a Gantt diagram if it was activated in the collection:

Example Code

```
'To Create and to activate a new calendar
Dim calendar As VcCalendar
VcGantt1.TimeScaleEnd = "01.01.2012"
VcGantt1.TimeScaleStart = "01.01.2011"
```

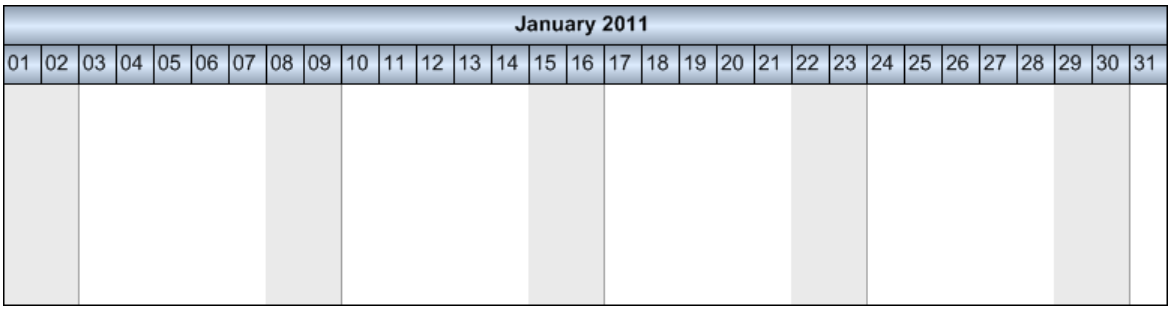
```
Set calendar = VcGantt1.CalendarCollection.Add("CompanyCalendar1")
VcGantt1.CalendarCollection.Active = calendar
```



If you now wish to re-activate the default base calendar, you can do this by the below settings:

Example Code

```
'To re-activate the default calendar
Dim calendar As VcCalendar
Set calendar =
VcGantt1.CalendarCollection.CalendarByName("BaseCalendar")
VcGantt1.CalendarCollection.Active = calendar
```

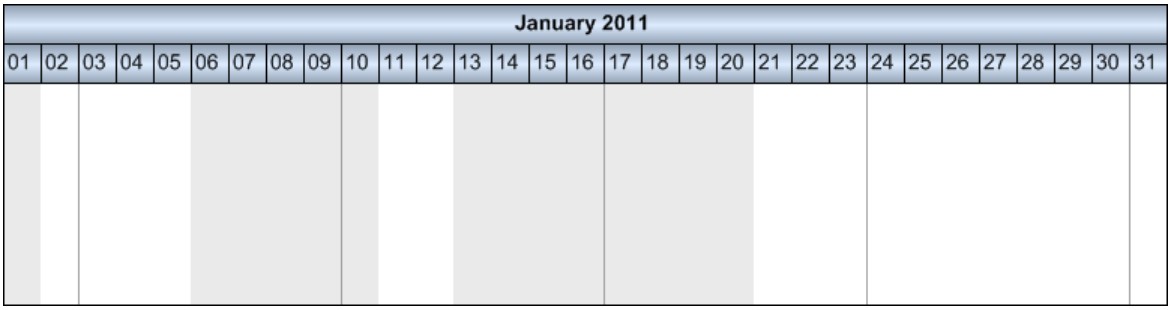


In the below example we will show how to define a working time profile by **intervals**. An irregular pattern of non-working days is to be defined: January 1st of 2011 and the period from January 6th to January 20th 2011, except for the two days of the 10th and 11th:

Example Code

```
'Defining non-working times
VcGantt1.TimeScaleEnd = "01.01.2012"
VcGantt1.TimeScaleStart = "01.01.2011"

Dim calendar As VcCalendar
Set calendar = VcGantt1.CalendarCollection.Add("CompanyCalendar1")
VcGantt1.CalendarCollection.Active = calendar

Dim interval As VcInterval
Set interval = calendar.IntervalCollection.Add("NewYear")
interval.CalendarProfileName = "<NONWORK>"
interval.StartDateTime = "01.01.2011"
interval.EndDateTime = "02.01.2011"

Set interval = calendar.IntervalCollection.Add("NonworkPeriod")
interval.CalendarProfileName = "<NONWORK>"
interval.StartDateTime = "06.01.2011"
```

```
interval.EndDateTime = "21.01.2011"

Set interval = calendar.IntervalCollection.Add("WorkPeriod")
interval.CalendarProfileName = "<WORK>"
interval.StartDateTime = "11.01.2011"
interval.EndDateTime = "13.01.2011"
VcGantt1.CalendarCollection.Update
```

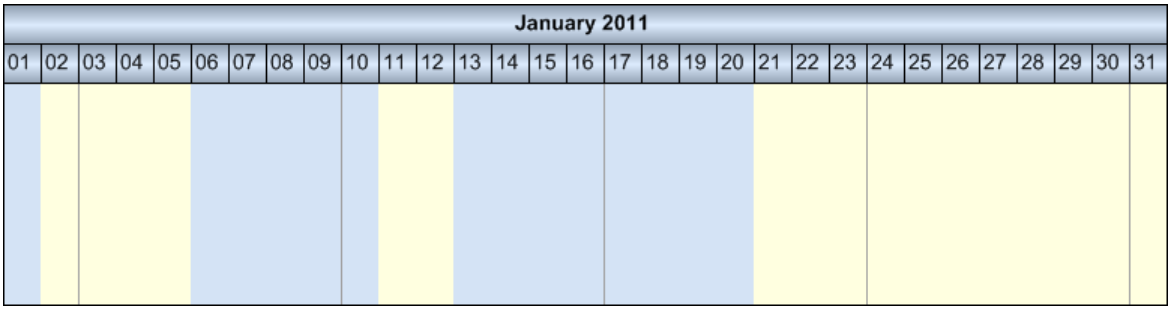


Visually, non-working times can be identified by the light gray shade. Since working times by default do not have a color, the white background of the diagram remains visible in them. In the next step, we want working times to appear in a light yellow color and non-working times in light blue. The colors are produced by graphical attributes that can be defined at the intervals.

Example Code

```
'Assigning colors to intervals
VcGantt1.TimeScaleEnd = "01.01.2012"
VcGantt1.TimeScaleStart = "01.01.2011"

Dim calendar As VcCalendar
Set calendar = VcGantt1.CalendarCollection.Add("CompanyCalendar1")
VcGantt1.CalendarCollection.Active = calendar

VcGantt1.TimeScaleCollection.FirstTimeScale.Section(0).CalendarGridEx(0)
.UseGraphicalAttributesOfIntervals = True

Dim interval As VcInterval

Set interval = calendar.IntervalCollection.Add("Work")
interval.CalendarProfileName = "<WORK>"
interval.BackColorAsARGB = &HFFFFFFFE0
interval.UseGraphicalAttributes = True

Set interval = calendar.IntervalCollection.Add("NewYear")
interval.CalendarProfileName = "<NONWORK>"
interval.StartDateTime = "01.01.2011"
interval.EndDateTime = "02.01.2011"
interval.BackColorAsARGB = &HFFD4E3F5
interval.UseGraphicalAttributes = True

Set interval = calendar.IntervalCollection.Add("NonworkPeriod")
interval.CalendarProfileName = "<NONWORK>"
interval.StartDateTime = "06.01.2011"
interval.EndDateTime = "21.01.2011"
interval.BackColorAsARGB = &HFFD4E3F5
interval.UseGraphicalAttributes = True

Set interval = calendar.IntervalCollection.Add("WorkPeriod")
```

```
interval.CalendarProfileName = "<WORK>"
interval.StartDateTime = "11.01.2011"
interval.EndDateTime = "13.01.2011"
interval.BackColorAsARGB = &HFFFFFFE0
interval.UseGraphicalAttributes = True

VcGantt1.CalendarCollection.Update
```



The below sample shows how to define a week where Monday to Friday are a working time while the weekend is free of work. The options introduced so far do not suffice for this; an object of the type **VcCalendarProfile** is required.

Please note: In VARCHART XGantt, VcCalendarProfile objects can be defined on a global or on a local level. Local calendar profile objects can only be used in the calendar in which they were defined, while global objects simultaneously can be used in different calendars. In our programming samples, merely local calendar profile objects are used. In terms of functions, local calendars do not differ from global ones. If a local and a global profile of identical names were created, within the corresponding calendar only the local profile is addressed; the global profile cannot be accessed.

A **calendar profile** of the type **vcWeekProfile** allows to describe working and non-working times of the days of a week. A week profile becomes effective only after it was added to the interval collection of the calendar. Setting **StartDateTime** and **EndDateTime** can be omitted, since we want our settings to be valid for the complete period of the calendar without any restriction. The calendar profiles of the pre-set names **<WORK>** and **<NONWORK>** have a defined meaning: they are used to allocate working and nonworking times.

Example Code

```
'Defining a week profile
Dim calendar As VcCalendar
Dim interval As VcInterval
Dim calendarProfile As VcCalendarProfile

Set calendar = VcGantt1.CalendarCollection.Add("CompanyCalendar1")
VcGantt1.CalendarCollection.Active = calendar
Set calendarProfile =
calendar.CalendarProfileCollection.Add("WeekProfile")
calendarProfile.Type = vcWeekProfile
```

```
VcGantt1.TimeScaleCollection.FirstTimeScale.Section(0).CalendarGridEx(0)
.UseGraphicalAttributesOfIntervals = True

Set interval = calendarProfile.IntervalCollection.Add("Mo-Fr")
interval.CalendarProfileName = "<WORK>"
interval.StartWeekday = vcMonday
interval.EndWeekday = vcFriday

Set interval = calendarProfile.IntervalCollection.Add("Sa")
interval.CalendarProfileName = "<NONWORK>"
interval.BackColorAsARGB = &HFFFFF69F
interval.StartWeekday = vcSaturday
interval.EndWeekday = vcSaturday

Set interval = calendarProfile.IntervalCollection.Add("Su")
interval.CalendarProfileName = "<NONWORK>"
interval.BackColorAsARGB = &HFFFBD3AA
interval.StartWeekday = vcSunday
interval.EndWeekday = vcSunday

Set interval = calendar.IntervalCollection.Add("StandardWeek")
interval.CalendarProfileName = "WeekProfile"
```

Distinguishing working and non-working times within a single day requires a day profile that allows to specify a precise clock time, for example from 8.00 h to 12.00 h am and from 1.00 h to 5.00 h pm. Since a day profile newly created consists of working time only, any interruption is to be defined as a non-working interval.

Example Code

```
'Defining a day profile
Dim interval As VcInterval
Dim calendarProfile As VcCalendarProfile

Set calendarProfile =
calendar.CalendarProfileCollection.Add("DayProfile")
calendarProfile.Type = vcDayProfile

Set interval = calendarProfile.IntervalCollection.Add("Interval_1")
' 00:00-8:00
interval.CalendarProfileName = "<NONWORK>"
interval.StartTime = "1.1.2011 0:00"
interval.EndTime = "1.1.2011 8:00"

Set interval = calendarProfile.IntervalCollection.Add("Interval_2")
' 12:00-13:00
interval.CalendarProfileName = "<NONWORK>"
interval.StartTime = "1.1.2011 12:00"
interval.EndTime = "1.1.2011 13:00"

Set interval = calendarProfile.IntervalCollection.Add("Interval_3")
' 17:00-24:00
interval.CalendarProfileName = "<NONWORK>"
interval.StartTime = "1.1.2011 17:00"
interval.EndTime = "1.1.2011 00:00"
```

The clock time is set by the object **DateTime**. The date fraction is ignored since it is meaningless in this context. The date only needs to be set in the constructor, to set a value to all parameters required by the constructor. In **Interval_3** it is important to specify 0 h instead or 24 h, since the latter is not accepted in the **DateTime** object.

Recurring days of a year, such as **New Year's Eve** on the 1st of January or **Christmas** and **Boxing Day** on the 25th and 26th of December are defined by a calendar profile which covers a whole year.

Example Code

```
'Setting a profile of fixed annual holidays
Dim calendarProfile As VcCalendarProfile
Dim interval As VcInterval

Set calendarProfile =
calendar.CalendarProfileCollection.Add("YearProfile")
calendarProfile.Type = vcYearProfile

Set interval = calendarProfile.IntervalCollection.Add("New Year")
interval.CalendarProfileName = "<NONWORK>"
interval.DayInStartMonth = 1
interval.StartMonth = vcJanuary
interval.DayInEndMonth = 1
interval.EndMonth = vcJanuary
Call SetAppearanceForHolidays(interval)

Set interval = calendarProfile.IntervalCollection.Add("Christmas")
interval.CalendarProfileName = "<NONWORK>"
interval.DayInStartMonth = 25
interval.StartMonth = vcDecember
interval.DayInEndMonth = 26
interval.EndMonth = vcDecember
Call SetAppearanceForHolidays(interval)
```

To avoid repeated settings that produce identical appearances of holidays, we collect the calls in a method named **SetAppearanceForHolidays**:

Example Code

```
'Method to set the visual appearance of holidays

Private Sub SetAppearanceForHolidays(ByVal interval As VcInterval)
 interval.BackColorAsARGB = &HFFFFA4A4
  interval.Pattern = vcWeavePattern
  interval.PatternColorAsARGB = &HFF404040
  interval.LineColor = &HFF808080
  interval.LineThickness = 1
  interval.LineType = vcSolid
  interval.UseGraphicalAttributes = True
End Sub
```

Please note: The color properties become effective only in those intervals, the CalendarProfileName of which was set either to **<WORK>** or to **<NONWORK>**. In addition, the interval property **UseGraphicalAttribute**

needs to be set to **true**. The same is valid for the calenderGrid property **UseGraphicalAttributesOfIntervals**.

Floating holidays such as Easter, and other holidays that depend on them have to be calculated for each year and need to be assigned to the calendar as fixed dates. The below method is very useful for this:

Example Code

```
'Method to find floating holidays
Const AshWednesday = 0
Const GoodFriday = 1
Const EasterSunday = 2
Const EasterMonday = 3
Const FeastOfCorpusChristi = 4
Const AscensionOfChrist = 5
Const WhitSunday = 6
Const WhitMonday = 7
Const CentralEuropeanSummerTimeStart = 8
Const CentralEuropeanSummerTimeEnd = 9

Private Function calculateAnniversaryForYear(ByVal year As Integer,
ByVal specialDay As Integer) As Date
  Dim g As Integer
  Dim c As Integer
  Dim h As Integer
  Dim i As Integer
  Dim j As Integer
  Dim month As Integer
  Dim day As Integer
  Dim dayOffset As Integer

   g = year Mod 19
   c = year Mod 100
   h = (c - c / 4 - (8 * c + 13) / 25 + 19 * g + 15) Mod 30
   i = h - (h / 28) * (1 - (29 / (h + 1)) * ((21 - g) / 11))
   j = (year + year / 4 + i + 2 - c + c / 4) Mod 7
   month = 3 + (i - j + 40) / 44
   day = i - j + 28 - 31 * (month / 4)
   dayOffset = 0

   Select Case specialDay
      Case AshWednesday
         dayOffset = -40
      Case GoodFriday
         dayOffset = -2
      Case EasterSunday
         dayOffset = 0
      Case EasterMonday
         dayOffset = 1
      Case AscensionOfChrist
         dayOffset = 39
      Case WhitSunday
         dayOffset = 49
      Case WhitMonday
         dayOffset = 50
      Case FeastOfCorpusChristi
         dayOffset = 60
      Case CentralEuropeanSummerTimeStart
```

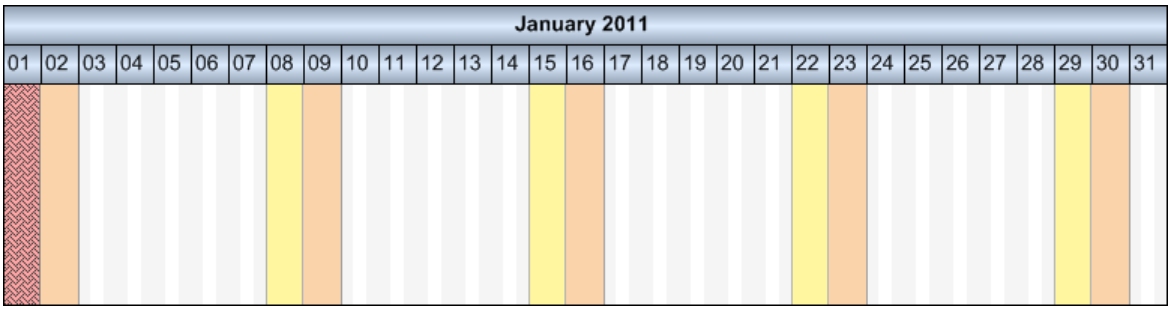
```
         month = 3
         day = 31 - Weekday("31.3" + yearConvert + 1)
      Case CentralEuropeanSummerTimeEnd
         month = 10
         day = 31 - Weekday("31.10" + yearConvert + 1)
   End Select
      Dim tmpDate As Date
      tmpDate = day & "." & month & "." & year
      calculateAnniversaryForYear = tmpDate + dayOffset
End Function
```

In the next step, the week profile and the holiday profile are assigned to the calendar as intervals. Then the floating holidays are calculated and assigned to the calendar in the same way:

Example Code

```
'Assembling the week profile, the holiday profile and the floating
holidays into an interval
Set interval = calendar.IntervalCollection.Add("Weekly_Pattern")
interval.CalendarProfileName = "WeekProfile"

Set interval = calendar.IntervalCollection.Add("Yearly_Pattern")
interval.CalendarProfileName = "YearProfile"

Dim startYear As Integer
Dim endYear As Integer

startYear = year(VcGantt1.TimeScaleStart)
endYear = year(VcGantt1.TimeScaleEnd)

Dim i As Integer
For i = startYear To endYear Step i + 1
   Set interval = calendar.IntervalCollection.Add("GoodFriday_" & i)
   interval.CalendarProfileName = "<NONWORK>"
   interval.StartDateTime = calculateAnniversaryForYear(i, GoodFriday)
   interval.EndDateTime = calculateAnniversaryForYear(i, EasterMonday)
   'interval.StartDateTime
   Call SetAppearanceForHolidays(interval)

   Set interval = calendar.IntervalCollection.Add("EasterMonday_" & i)
   interval.CalendarProfileName = "<NONWORK>"
   interval.StartDateTime = calculateAnniversaryForYear(i, EasterMonday)
   interval.EndDateTime = interval.StartDateTime
   Call SetAppearanceForHolidays(interval)

   Set interval =
calendar.IntervalCollection.Add("FeastOfCorpusChristi_" & i)
   interval.CalendarProfileName = "<NONWORK>"
   interval.StartDateTime = calculateAnniversaryForYear(i,
FeastOfCorpusChristi)
   interval.EndDateTime = interval.StartDateTime
   Call SetAppearanceForHolidays(interval)

   Set interval = calendar.IntervalCollection.Add("AscensionOfChrist_" &
i)
   interval.CalendarProfileName = "<NONWORK>"
   interval.StartDateTime = calculateAnniversaryForYear(i,
AscensionOfChrist)
   interval.EndDateTime = interval.StartDateTime
```

```
   Call SetAppearanceForHolidays(interval)

   Set interval = calendar.IntervalCollection.Add("WhitMonday_" & i)
   interval.CalendarProfileName = "<NONWORK>"
   interval.StartDateTime = calculateAnniversaryForYear(i, WhitMonday)
   interval.EndDateTime = interval.StartDateTime
   Call SetAppearanceForHolidays(interval)
Next

VcGantt1.CalendarCollection.Update
```



These are the steps in summary that are required to put assemble a calendar. Depending on the requirements single steps may be omitted:

1. Creating day profiles of different working days
2. Assembling a week profile by using the day profiles
3. Defining a holiday profile
4. Assigning the week profile and the holiday profile to the interval collection of the calendar
5. Assigning additional dates (e.g. floating holidays) to the interval collection

The interval object allows to define periods that can be interpreted as working time or as non-working time. The periods are distinguished to be **<WORK>** or **<NONWORK>** by the **CalendarProfileName** property. By this property, a calendar can also refer to other existing profiles and adopt their settings. When setting this property please take into account that only certain profile types can be assigned, depending on the interval type. The interval type implicitly is selected by the chosen profile type. The pre-set default value of the **calendar profile**, which is **vcDayProfile**, can be modified by a corresponding setting initially, that is, before defining intervals.

Object	Profile Type Chosen	Interval Type Assigned
VcCalendar		vcCalendarInterval
VcCalendarProfile	vcYearProfile	vcYearProfileInterval
	vcWeekProfile	vcWeekProfileInterval
	vcDayProfile	vcDayProfileInterval
	vcVariableProfile	vcVariableProfileInterval

The profile type suggests the allowed interval type. For example, a day profile always requires intervals of the type **vcDayProfileInterval**.

Interval Type	<WORK>	<NONWORK>	vcDayProfile	vcWeekProfile	vcYearProfile	vcVariableProfile
vcCalendarInterval	■	■	■	■	■	■
vcVariableProfileInterval	■	■	■	■	■	
vcYearProfileInterval	■	■	■	■		
vcWeekProfileInterval	■	■	■			
vcDayProfileInterval	■	■				

Calendar profiles can show the types **day profile, week profile, year profile** and **variable profile**. In a day profile, intervals can only be defined by clock times that range within the limits of a day. A week profile holds day profiles to apply on certain days. A year profile assigns selected day profiles that apply to a single recurring day or to a couple of recurring days. A variable profile contains a sequence of different working times. Depending on the interval types **vcCalendarInterval, vcDayProfileInterval, vcWeekProfile-Interval, vcYearProfileInterval** and **vcVariableProfileInterval** only some properties of the object are of relevance. The below table maps profile types and relevant properties.

vcCalendar-Interval	vcYearProfile-Interval	vcWeekProfile-Interval	vcDayProfile-Interval	vcVariable-Interval
StartDateTime	StartMonth	StartWeekday	StartTime	Duration
EndDateTime	EndMonth	EndWeekday	EndTime	TimeUnit
	DayInEndMonth			
	DayInStartMonth			

A **CalendarInterval** describes a unique time span in a precisely defined interval. Example: May 5th, 2010 from 11:30 h to September, 15th 2010 17:00 h.

A **YearProfileInterval** allows to define days or a time spans that recur once a year. Example: May 1st or December, 24th - 26th.

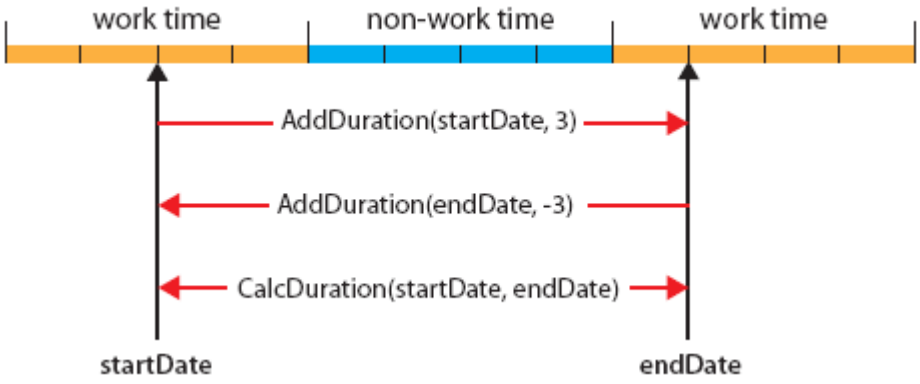
A **WeekProfileInterval** handles a single or several days of a week. Example: Saturday, or Monday - Friday.

A **DayProfileInterval** deals with time specifications that range within a day. Example: 8.00 h to 17.00 h.

A **VariableProfile** describes a time span without referring to a defined date or time. The unit of the time span may be days, hours, minutes or seconds and is specified by the property TimeUnit of the interval object. Example: 4 hours.

How to Calculate with Calendars

Calculations in a calendar are not necessarily visible in the time scale. The method **AddDuration** of the object **Calendar** calculates the final date from the start date and the specified number of working time units while taking into account non-working periods. Passing time units of negative signs will result in calculating the start date from a given end date. The method **CalcDuration** being a complement of the method AddDuration calculates the number of working time units (duration) from a given start and an end date.



> How the Calculating Methods Work

Please note: Working time units specified as days, hours, minutes or seconds need to correspond to what was defined by the property TimeUnit of the VcGantt object.

The method **AddDuration** ensures, that the dates calculated always are located in a working time interval. At the same time, a backward calculation does not necessarily provide a result equal to the source value of the forward calculation, if the source value had been situated in a non-working time.

> Limited Reversibility of calculations

When activities are interactively created or modified, VARCHART XGantt automatically cares that activities cannot start or finish within non-working times. If you wish the behavior to be consistent while creating or modifying nodes by the API, you need to ensure this by manually correcting the start or end date. For this, a start date being situated in a non-working time needs to be moved to the beginning of the succeeding working time interval, and an end date correspondingly to the end of the previous working time interval. There are methods to identify the limits of intervals. They are discussed in detail in the below chapter.

Example Code

```
If calendar.IsWorktime(StartDate) = False Then
   StartDate = calendar.GetNextIntervalBorder(StartDate)
End If

If calendar.IsWorktime(EndDate) = False Then
   EndDate = calendar.GetNextIntervalBorder(EndDate)
End If
```

> Daylight Saving Time

VARCHART XGantt automatically supports daylight saving time. In central Europe, DST starts on the last Sunday in the month of March and finishes on

the last Sunday in the month of October. On the start of DST the clocks are put forward from 2:00 h to 3:00 h and at its end they are put back from 3:00 h to 2:00 h.

Start of daylight saving time:

00:00 h	01:00 h	03:00 h	04:00 h	05:00 h	06:00 h	...

End of daylight saving time:

00:00 h	01:00 h	02:00 h	02:00 h	03:00 h	04:00 h	...

On the start day of daylight saving time, the method **calcDuration** retrieves a time span of 23 hours while on its final day, 25 hours are returned, if **TimeUnit** is set to hours. If set to days, the time span in both cases will be exactly 1 day.

Retrieving the Limits of Time Intervals

The methods of the **Calendar** object to retrieve the limits of a time interval **GetStartOfInterval, GetNextIntervalBorder** and **GetPreviousIntervalBorder** allow to iterate over working time intervals and non-working time intervals. The results returned are relative and refer to a reference date which is passed by the methods as a parameter.

A date can be checked for being located in a working time or in a non-working time by the method **IsWorkTime** of the Calendar object. Although the start date of a new interval equals the end date of the previous one, the start date always belongs to the new interval (open to the right).

The methods **GetEndOfPreviousWorkTime** and **GetStartOfNextWorkTime** do not provide new options but merely simplify the handling of working time intervals.

In the below programming sample, the time intervals of the calendar are retrieved and written to a file. Beside, the working time available in the given period is calculated:

Example Code

```
Private Sub writeCalendarIntervalsToFile(ByVal filename As String, ByVal
calendar As VcCalendar, ByVal startDate As Date, ByVal endDate As Date,
ByVal listWorkIntervals As Boolean, ByVal listNonWorkIntervals As
Boolean)
Dim tmpStartDate As Date
Dim nextStartDate As Date
Dim totalWorkTime As Integer

Open filename For Output As #1
Print #1, "Time Intervals of " & calendar.Name & "between " & startDate
& " - " & endDate
```

```
   tmpStartDate = startDate
   Do While tmpStartDate < endDate

     nextStartDate = calendar.GetNextIntervalBorder(tmpStartDate)

     If tmpStartDate = nextStartDate Then
        nextStartDate = endDate
     End If

     If nextStartDate > endDate Then
        nextStartDate = endDate
     End If

     If calendar.IsWorktime(tmpStartDate) Then
        If listWorkIntervals Then
           Print #1, "WorkInterval" & " " & tmpStartDate & "  " &
nextStartDate
        End If
     Else
          If listNonWorkIntervals Then
            Print #1, "NonWorkInterval" & " " & tmpStartDate & " " &
nextStartDate
          End If
     End If

      tmpStartDate = nextStartDate
Loop

   totalWorkTime = calendar.CalcDuration(startDate, endDate)
    Print #1, "Total work time: " & totalWorkTime & " Units"
 Close #1
End Sub
```

Please note: Intervals in the calendar can be specified as exactly as by seconds and may comprise an interval of 137 years (ulong in seconds) at maximum.

> Code to Write Intervals to a File

Example Code

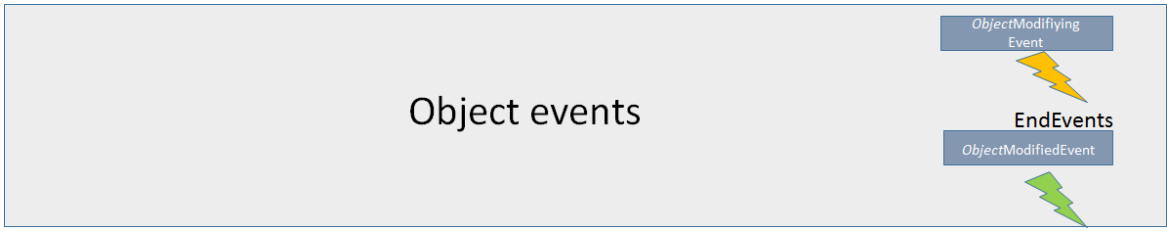
```
Call writeCalendarIntervalsToFile("C:\text.txt", calendar,
VcGantt1.TimeScaleStart, VcGantt1.TimeScaleEnd, True, True)

Time Intervals of CompanyCalendar_1 between
01.01.2011 00:00:00 - 01.01.2012 00:00:00

01.01.2011 00:00:00 - 02.01.2011 00:00:00 non-work time
02.01.2011 00:00:00 - 03.01.2011 00:00:00 non-work time
03.01.2011 00:00:00 - 03.01.2011 08:00:00 non-work time
03.01.2011 08:00:00 - 03.01.2011 12:00:00 work time
03.01.2011 12:00:00 - 03.01.2011 13:00:00 non-work time
03.01.2011 13:00:00 - 03.01.2011 17:00:00 work time
03.01.2011 17:00:00 - 04.01.2011 00:00:00 non-work time
04.01.2011 00:00:00 - 04.01.2011 08:00:00 non-work time
04.01.2011 08:00:00 - 04.01.2011 12:00:00 work time
04.01.2011 12:00:00 - 04.01.2011 13:00:00 non-work time
```

```
04.01.2011 13:00:00 - 04.01.2011 17:00:00 work time
04.01.2011 17:00:00 - 05.01.2011 00:00:00 non-work time
...
30.12.2011 00:00:00 - 30.12.2011 08:00:00 non-work time
30.12.2011 08:00:00 - 30.12.2011 12:00:00 work time
30.12.2011 12:00:00 - 30.12.2011 13:00:00 non-work time
30.12.2011 13:00:00 - 30.12.2011 17:00:00 work time
30.12.2011 17:00:00 - 31.12.2011 00:00:00 non-work time
31.12.2011 00:00:00 - 01.01.2012 00:00:00 non-work time

Total work time: 2064 Units
```

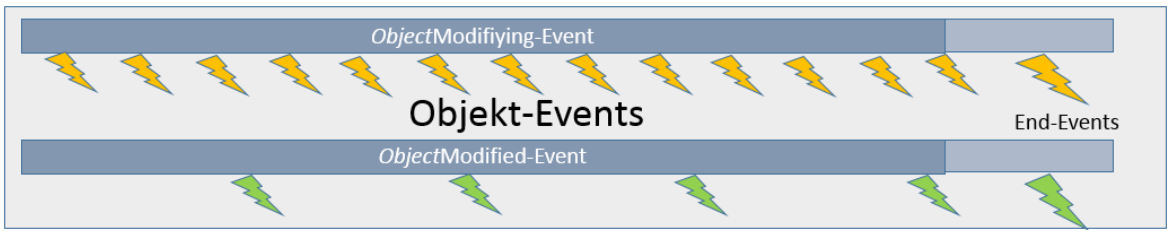
3.13 Interaction Events

During drag & drop interactions with the live update being enabled, receiving and processing information on the object would be quite useful.

In the default behavior, no feedback is given as to the status of the concerned object. Only when the mouse key is released, information on the old (before pressing the mouse key) and the new (after having released the mouse key) status is given by an **ObjectModifying** event. In addition, an **ObjectModified** event indicates that the operation is finished internally.



To solve this problem of not receiving information during mouse interactions, use the Interaction events that accompany and describe the interaction. Moreover, the object events' time of calling and frequency were modified as of XGantt version 5.0.



Interactions involved

We will explain events that describe the process of an interaction in VARCHART XGantt and the objects involved in greater detail, i.e. "Drag(Drop)" events during interactions that

- start with pressing the left mouse key at an object
- carry out movements with the mouse key being pressed
- end with releasing the left mouse key
- are treated in the course of "Live Update"

Terminology

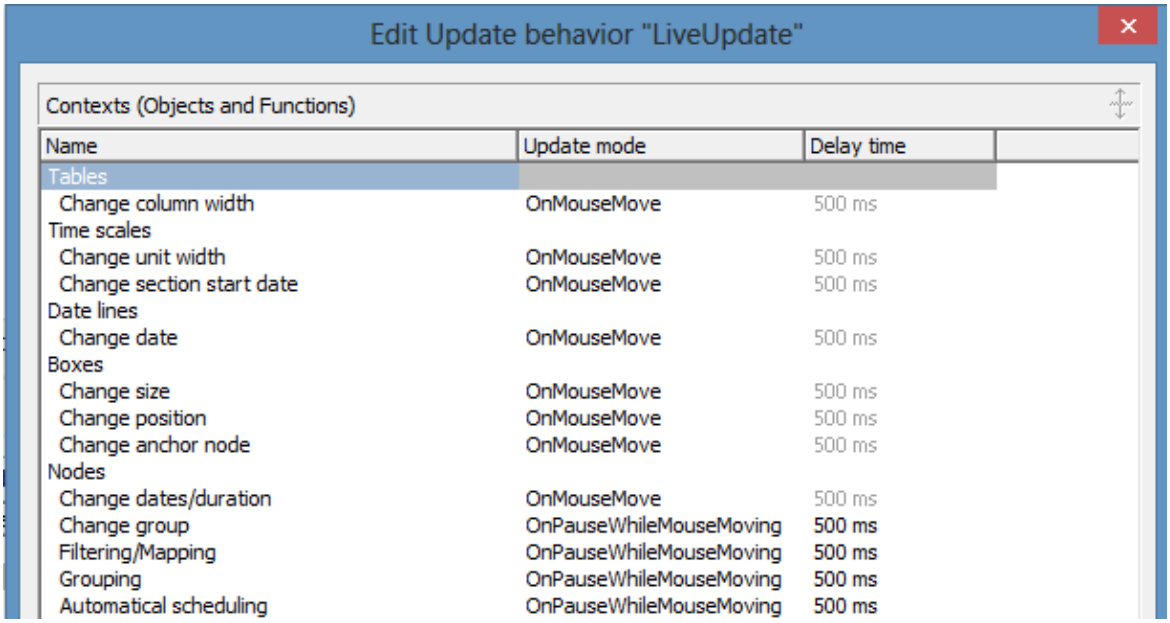
For a better understanding we'd like to further explain some terms that are used in the text.

> Object Events

Object events, such as **VcDateLineModifying, VcDateLineModified, VcNode¬Modifying, VcNode¬Modified** etc., are events, that, according to the practice already known up to now, are thrown at the end of an action during the addressed interactions.

> Live Update

Live update means that a "Drag Drop" action causes a "What if the object was updated here?"scenario to be shown permanently, this resulting in processing different contexts, such as direct or dependent functionalities during an interaction, at different times. If, for instance, a node is being moved, this results in modifying various data and the node's position, this in turn resulting in modifying the histogram curves or the summary bars, for instance. Depending on the settings in the Live Update dialog, the modifications will either come into effect at once or after hovering with the mouse a time span to be specified or at the end of the action on releasing the mouse key.



Example: What does the updates look like if the update behavior "OnMouseMove" is selected for the moving of nodes?

Immediate effects on the node:

- every date value of the node
- filters are evaluated, thus causing other colors, e.g., to appear in the table area

- osummary bars
- histogram curves

Modifications after a waiting period (500 ms)

- positioning the node in a group, for instance
- optimization with corresponding layout of the node order

Only updates that are necessary and meaningful in the total context of the action should be carried out, because otherwise the chart would become too restless.

InInteraction Events

From VARCHART XGantt 5.0 SR3 onward, object events can be processed already while the interaction is running, this objects being called InInteraction events.

Important: Be sure **to enable** the InInteraction events beforehand, either by the property **VcGantt.InInteractionEventsEnabled = true** or on the **General** property page.

Please note that when talking about interactions with nodes in the real mode, we will call the display object **Real (node)** and the data element in the chart **Chart** node. The chart node is not visible during the live interaction in the chart area because it will be replaced temporarily by the real node there, its presence, however, affecting the diagram in terms of ribbon height, optimization, colors in the table area etc.

This way, according information on the normal objects are delivered during the interaction matching with the displayed phantom or real node.

When a node is moved, every snapping into place of the node (depending on its time unit and increment) causes a **VcNodeModifying** to be thrown (yellow lightnings). The real node shows the possible position and the possible layout and describes this status by the **VcNodeModifying** event. The node (e.Node) being passed in the event args, represents the real node's status.

Important: This is why queries for properties of the chart nodes don't make sense or are not possible. Only the properties **get/setDataField, AllData, ID** can be retrieved or set.

If, depending on the selected updating context, e.g. "On pause while mouse moving", the real object is updated, this will be indicated by the **Modified**

event (green lightning). This can but doesn't have to happen at the same time as the Modifying events.

If a node is moved while the updating behavior "On mouse move" is selected, both events will appear at the same time.

To sum up the facts:

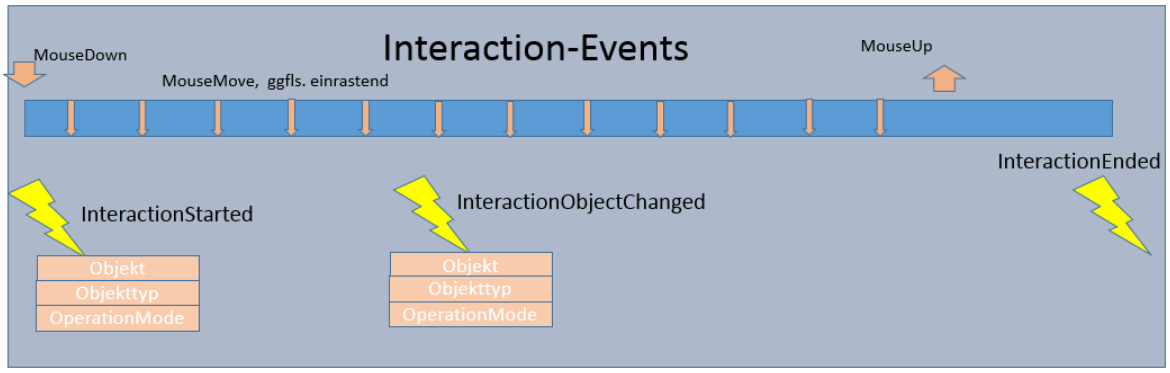
- If a node is moved, its modification, indicated by the real node, will be permanently described by the VcNodeModifying event.
- Modifications of the chart node are indicated by the VcNodeModified event.
- When the interaction is finished, upon releasing the mouse key, the concluding event pair, consisting of the **VcNodeModifying** and the **VcNodeModified** event are provided.

The concerned objects in events that use real nodes are the real objects.

In the last **VcNodeModifying** event, the chart node (as opposed to the previous **VcNodeModifying** events) with the values that were last set during the interaction is provided, i.e. the status at the time of the last small green lightning. **e.OldNode** of the EventArgs describes the status at the beginning of the action. This way, the start and end status of the interaction can be compared.

As always, the chart node is available in the last VcNodeModified event and all internal processes are finished.

Interaction Events



As described above, the object events are now thrown during and at the end of an interaction. The signature of the event handler, e.g. of the **VcNodeModifying** event don't differ there. But how to recognize whether the event has been thrown during or at the end of an interaction?

This could be important, because not every modification resulting from a mouse movement, for instance, is to be stored to a data base: This would

cause too much time-consuming effort. Of course, the data shall only be stored after the action was finished.

This problem can be solved now by some new events that accompany and describe the interaction and can be evaluated in the object events during the interaction.

As soon as the left mouse key is pressed, the **VcInteractionStarted** event delivers information on the object the mouse key is standing on (object and object type) and on what is happening with the object. Everything that is needed for the interaction can be prepared.

Tip: The update behavior can also be switched object- and context-specific here. In an extreme case, one could have one node react completely dynamical and another one with a blue phantom frame. Moreover, an according setting (**InInteractionEventsEnabled**) allows for an individual decision about whether the object events are to come also during the interaction or not.

Example: Node

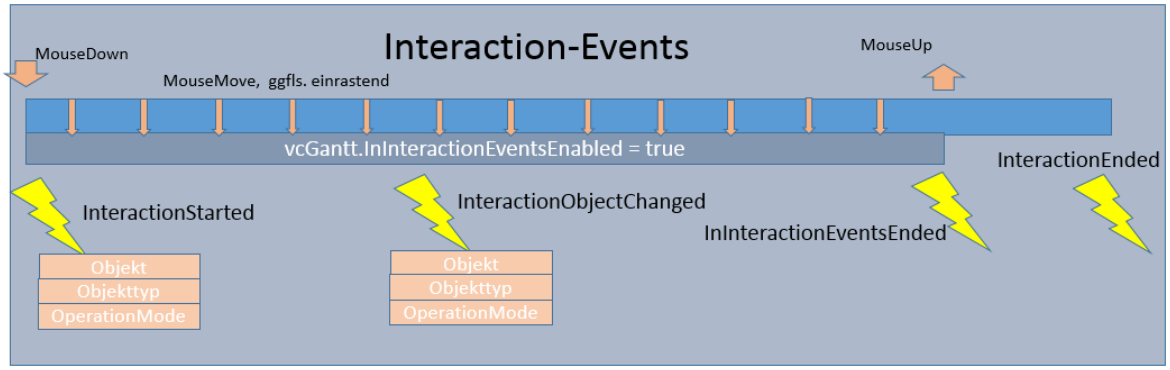
By

- Object: NodeObject
- Type: vcObjTypeNodeInDiagram
- OperationMode: vcIIMMoveNode
- upon pressing the left mouse key, the **VcInteractionStarted** event shows that the moving of a node in the chart has started.
- Information or elements that ought to accompany the interaction can be initialized here.

- **Creating Objects**
- In some interactions, there's no object available initially, e.g. when creating nodes or boxes. In this case, the event **VcInteractionObjectChanged** comes as soon as the object was created internally, being the real chart node where nodes are concerned.
- The end of the action is indicated by the **VcInteractionEnded** event. Every additional element having been used during the interaction can be removed here.
- When new objects are created with Interaction events, the process is as follows:
- VcInteractionStarted
- VcInteractionObjectChanged

- Modifying/Modified Events, showing modifications when creating an element
- Creating und Created Events
- VcInteractionEnded.

> InInteraction Events activated during the interaction



When the Interaction events are also enabled during the interaction (**vcGantt.InInteractionEventsEnabled = true**), there will be an additional event indicating the end of these events upon releasing the mouse key: **VcInInteractionEventsEnded**.

This makes it easy to differentiate the object events being thrown during the interaction from those that are thrown at the end of the interaction. If this event is thrown, the next object event will be the concluding event.

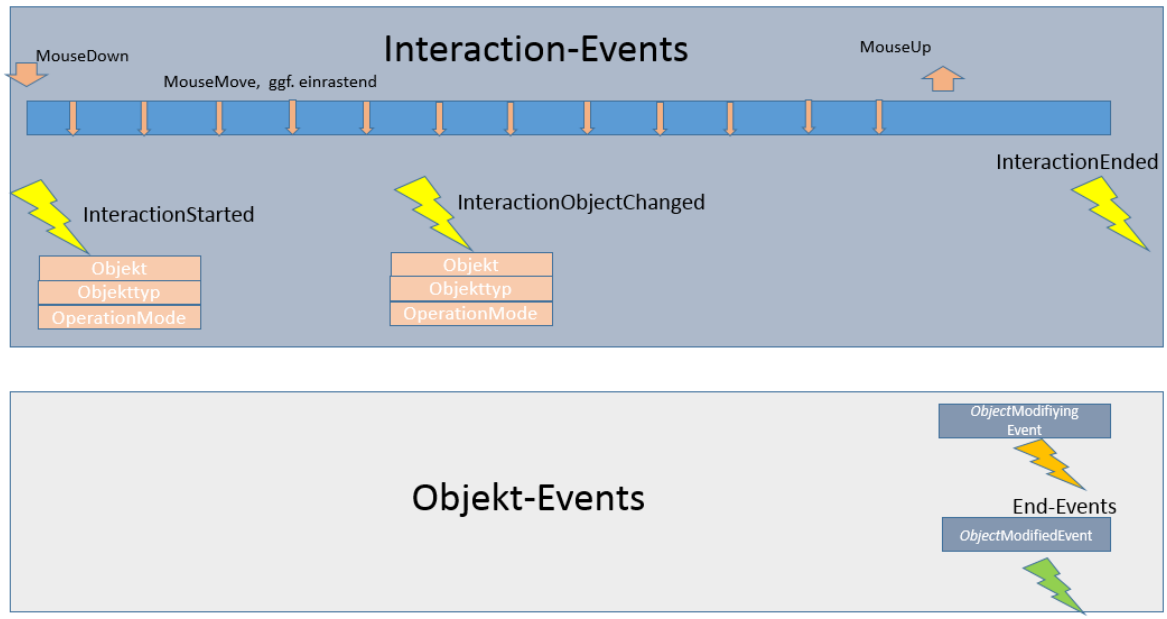
> Possible Scenarios

In other words, there are two possible conditions when using Interaction events.

Controlling an interaction with:

- InInteraction Events being switched off
- InInteraction Events being switched on

> **Cooperation with the events of the involved objects while the InInteraction events are deactivated**



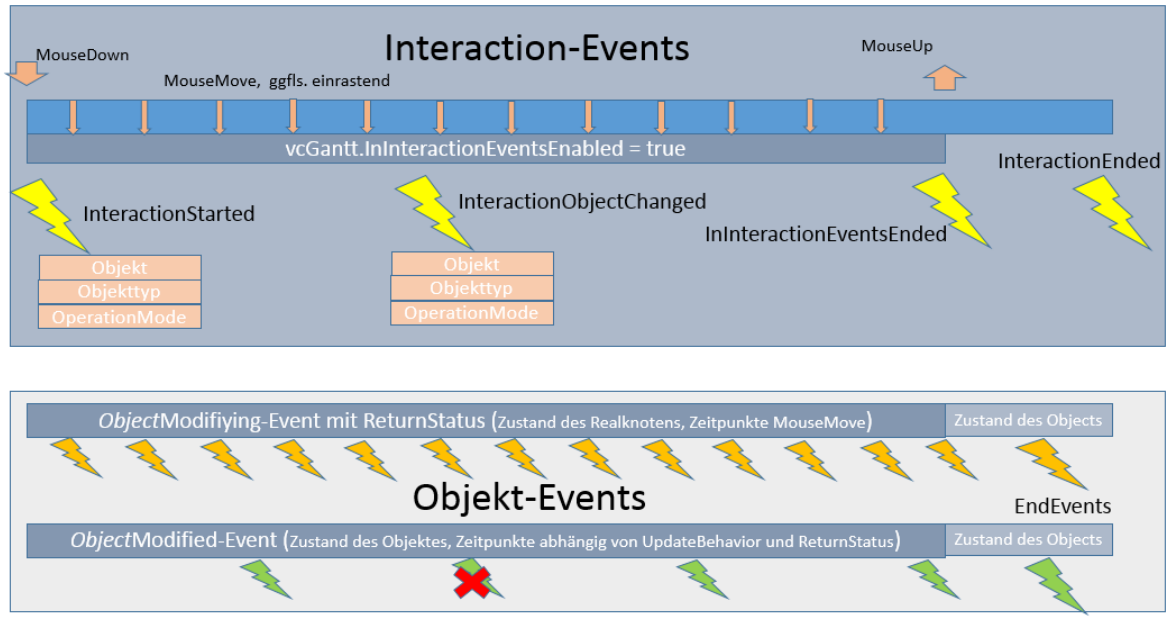
The screenshot shows how the Interaction (yellow lightnings) and the object events (ochre and green lightning) cooperate when InInteraction events are switched off (**vcGantt.InInteractionEventsEnabled = false**):

The interaction is started which is indicated by the **InteractionStarted** event.

When releasing the mouse key, the object events appear first, e.g. **VcNodeModifying** and **VcNodeModified** with a node. In other words this is the old behavior regarding object events so that existing code in the object events doesn't have to be modified if the InInteraction events are not used.

The end of the interaction is indicated by the **VcInteractionEnded** event.

> Cooperation with the events of the involved objects while the InInteraction events are activated



If the InInteraction events are used, the following events appear:

- **VcInteractionStarted** upon pressing the left mouse key
- Modifying and Modified events while the mouse is moved
- **VcInInteractionEventsEnded** and afterwards the finishing object events when the left mouse key is released
- **VcInteractionEnded** to indicate the end of the interaction.

Example: Moving a node:

The interaction starts when the left mouse key is pressed while the mouse cursor is at a node. The event **VcInteractionStarted** appears.

The events appearing upon moving the mouse indicate the status of the real node (**VcNodeModifying**) and while updating (**VcNodeModified1>) the chart node.**

When the mouse key is released, the VcInInteractionEventsEnded event appears

The object events **VcNodeModifying** and **VcNodeModified** indicate the status of the chart node at the end of the interaction.

The last to appear is the **VcInteractionEnded** event.

> Example: Behavior of the object events when the node update behavior "On mouse move" is set

Since the **VcNodeModifying** event allows for the EventReturnstatus (e.ReturnStatus) to be modified, this can now also be done during the interaction.

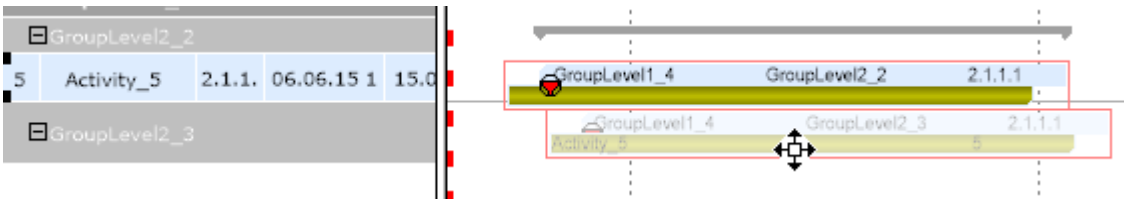
So, if e.ReturnStatus = ReturnStatusFalse indicates that the provided data are not "valid", the object in the chart will not be refreshed with the next possible update and the according **VcNodeModified** event will not be thrown.

This is visualized by the object remaining at its old place and the current position being still indicated by the phantom.

The status of objects visualized by reals (currently only nodes and node boxes) is indicated as follows:

The current position is visualized by a brightened real, the values of which also still being provided in the events.

The last valid status, i.e. the last one not returning ReturnStatusFalse as e.ReturnStatus, is indicated by another real, that quasi "gets stuck" there; this way both pieces of information are being visualized.



At the node, the values of the last valid status, i.e. that of the stuck real, correspond to the **e.OldNode** in the **VcNodeModifying-Event**

If the last **VcNodeModifying** event before the **VcInInteractionEventsEnded** was finished with ReturnStatusFalse, the last valid state will be provided in the End events.

There it can be decided whether to accept this state or not. If in the End event ReturnStatusFalse is set, the original start status will be restored.

Practical Tip: We recommend to create an "accompanying InteractionInfo" object that provide the needed information on the interaction in the events and can be evaluated accordingly.

3.14 Interaction Events

3.15 Layers

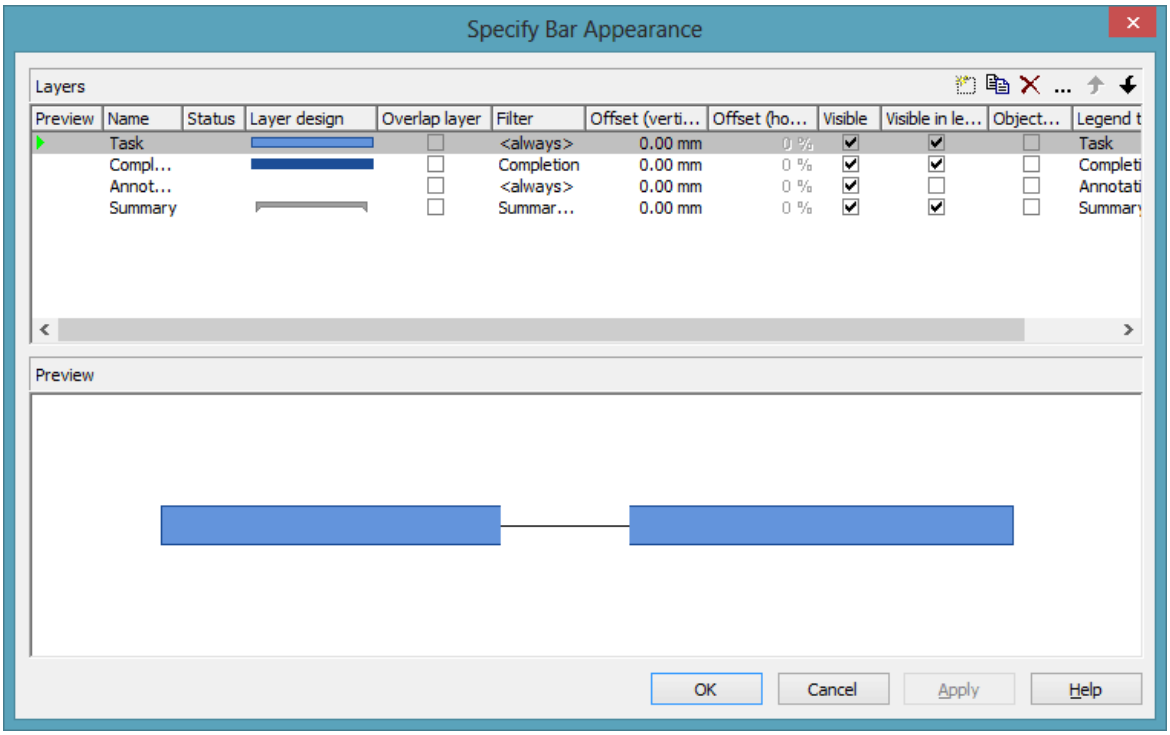
A layer represents a specific point in time (symbol or bitmap layers) or a timespan (rectangle, wedge-shaped or line layers).

Activities are graphically displayed by one or more layers. If an activity comprises several layers, the layers are drawn on top of each other, starting by the layer of lowest priority and finishing by the layer of highest priority.

For each layer a filter is used. By using filters, you can assign a layer to only those activities that fulfill the filter conditions.

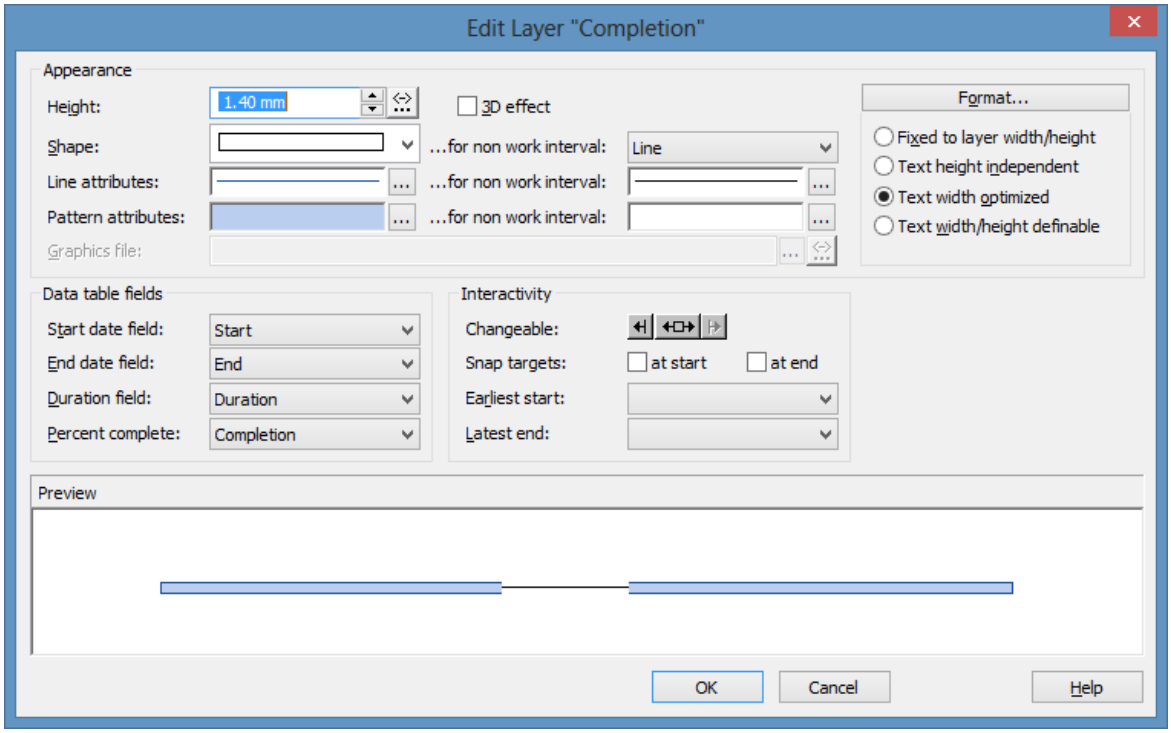
The layers can have different patterns, background pattern colors and/or annotations. In addition, they can be of varying heights and offsets, vertically or horizontally, so all layers that belong to a node have a chance to be visible.

In the **Specify Bar Appearance** dialog box, you can define layers. All layers existing are displayed here, in the order of their drawing priority.



By the buttons in the top right corner of the dialog you can add ( ), copy ( ), delete ( ) or edit a layer ( ).

To edit a layer, please select it from the list, click on the **Edit layer** button ( ) or double-click on the desired layer graphics in the column **Layer design**. The **Edit Layer** dialog box will open where you can edit the graphical attributes of the layer.



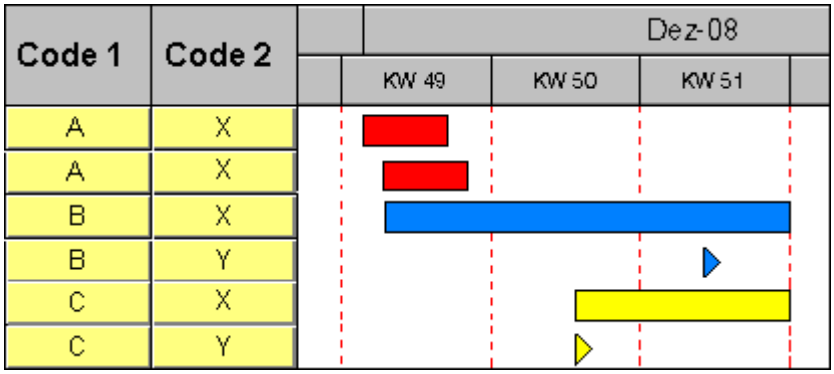
> Applying filters to layers

By using filters, you can have layers assigned to specific nodes only, depending on the data of the layer.

To edit a filter, in the **Specify Bar Appearance** dialog box please click on the **Filter** field. Two buttons will appear. Click on the **Edit** button to open the **Administer Filters** dialog box. From here you can get to the **Edit Filter** dialog box where you can edit the filter condititions.

(Also see "Important Terms: Filters".)

In the below example "Code2 = X" is defined for a rectangle layer, "Code2 = Y" for a symbol layer. The colors are assigned by mapping Code1.



> Layer shapes

You can choose between rectangle layers, wedge-shaped layers, line layers, symbol layers, bitmap layers and invisible symbol layers.

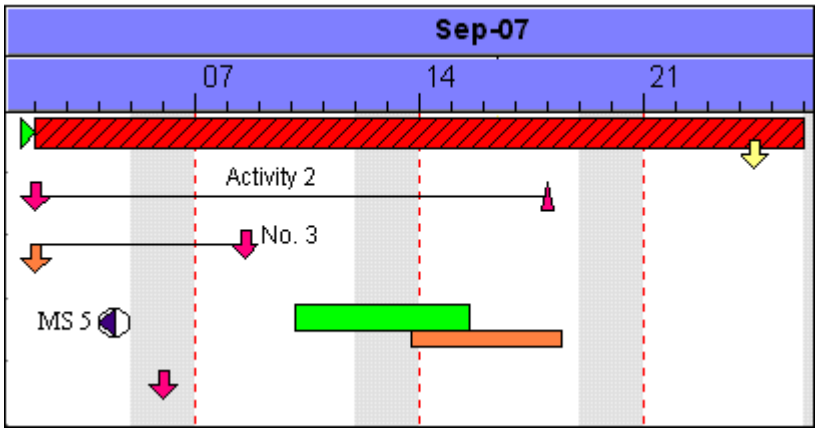
Select the layer shape from the **Shape** select box in the **Edit Layer** dialog box.

Symbol layers represent specific points in time. Some symbol layers were predefined, but you can also define your own symbol layers (for example company logos). You can select a bitmap file by the **Graphics file** field.

Timespans can be visualized by rectangle layers, wedge-shaped layers or line layers. Wedge-shaped layers are useful for visualising increasing and decreasing activities, e.g. during periods of starting or phasing out.

The layer type **invisible symbol** is invisible, except for its annotation; beside, it is not displayed in the legend. So you can use it for additional annotations in activities.

By combining layer shapes, patterns, colors and filters, a large number of different layers can be defined. The below picture displays some examples:



> Degree of completion

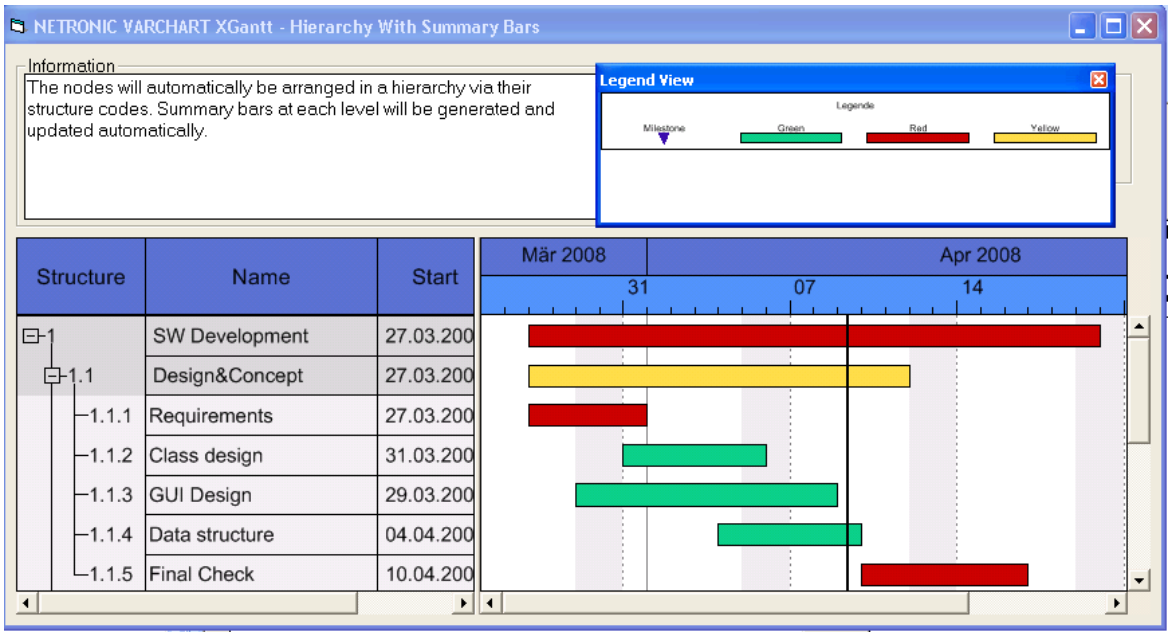
VARCHART ActiveX allows to recognize the degree of completion of an activity at a glance. To display the degree of completion, please proceed as described below:

Create a layer **Completed** and edit it by the **Edit Layer** dialog box. For wedge-shaped and rectangle layers you can select a data field that contains the degree of completion (indicated as %) of the selected layer. For example, select for the layer **Completed** the data field **% completed**. Now specify the graphical attributes (color, pattern etc.) so that the **Completed** layer can easily be recognized.

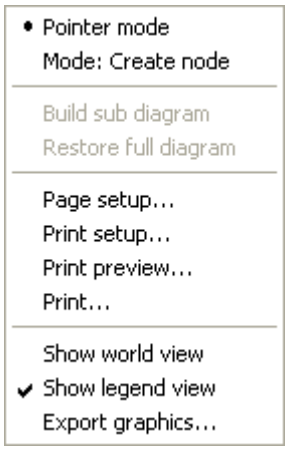
Degree of completion: 90 %

3.16 Legend View

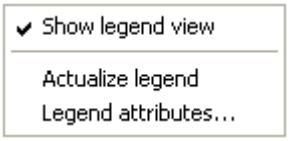
The legend view is an additional window that lets you display a legend on the screen. The layout of the legend can be specified with the legend attributes of **VcBorderBox** or in the dialog **Legend attributes** which can be reached from the **Border area** property page.



At runtime, you can switch on and off the legend view in the default context menu by the menu item **Show legend view**.



Moreover, you can switch on or off the legend view in the legend's context menu.



The context menu offers two more items: **Actualize legend** and **Legend attributes**. By selecting the latter you call the corresponding dialog.

The refreshing of the legend is needed after modifications in the chart, such as adding or deleting nodes, because they are not displayed automatically. The refreshing can also be carried out by switching off and on the legend view. This concerns the loading of nodes as well. If on the property page **Additional views** the attribute **Initially visible** was selected for the legend view and no nodes have been loaded when running the program, the legend stays empty until it was refreshed.

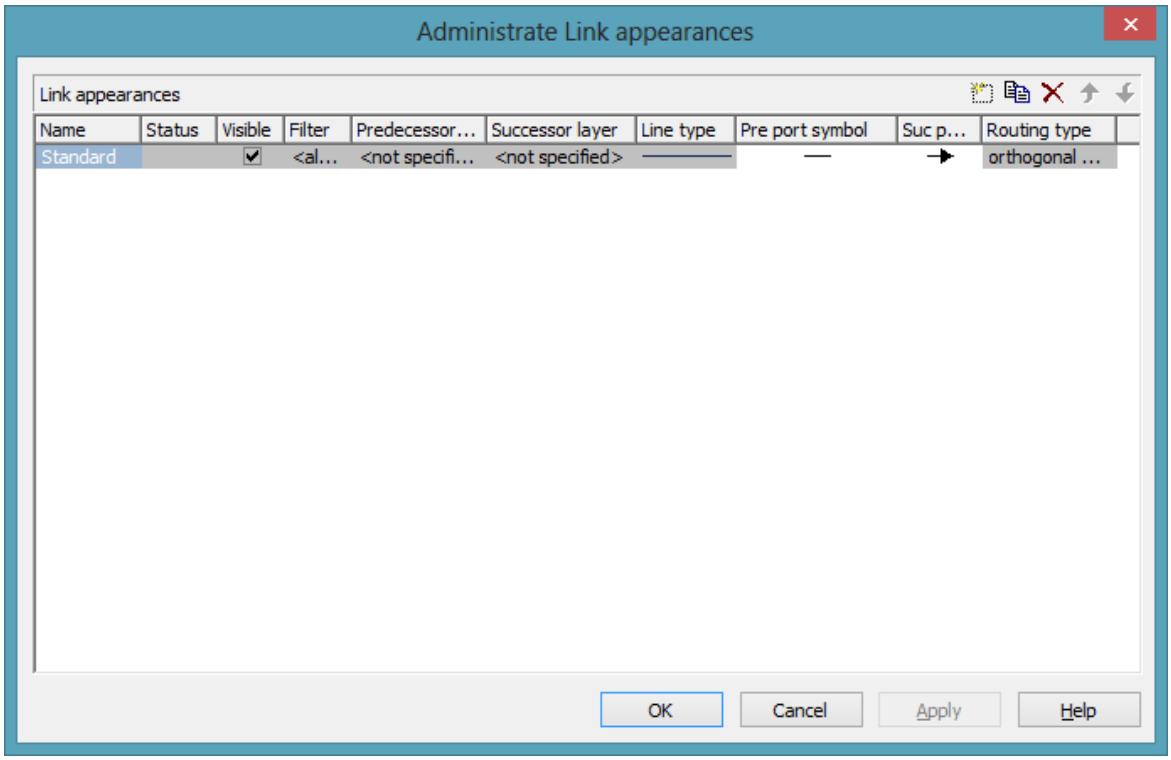
On the **Additional Views** property page you can set the properties of the Legend View. For details please see **The Additional Views Property Page** in the chapter **Property Pages and Dialog Boxes** .

The properties of the Legend View can also be set by the API property **VcGantt.VcLegendView**.

3.17 Link Appearance

You can define different link appearances in the **Administrate Link appearances** dialog. The link appearances will be assigned to the links dynamically by filters.

> Defining a Link Appearance



3.18 Links

A link is defined by a record of the data table which contains the link data. Link data is automatically and simultaneously generated on the generation of nodes. Link data can be loaded from a file by API calls or can be generated interactively by the user.

> Generating Links

At run time, you can use the mouse to draw links between two activities after **Mode: Create Link** was activated in the context menu.

The link is drawn from the first layer of the predecessor activity to the first layer of the successor activity. If a link is created interactively, the application is notified by the **OnLinkCreate** event. Alternatively, you can create links by the API method **InsertLinkRecord**.

> Deleting Links

You can delete a link by clicking on it using the right mouse button to pop up the context menu and by selecting the menu item **Delete**. Beside, you can delete links by the VARCHART ActiveX method **DeleteLinkRecord** or by the method **VcLink.DeleteLink**.

> Events

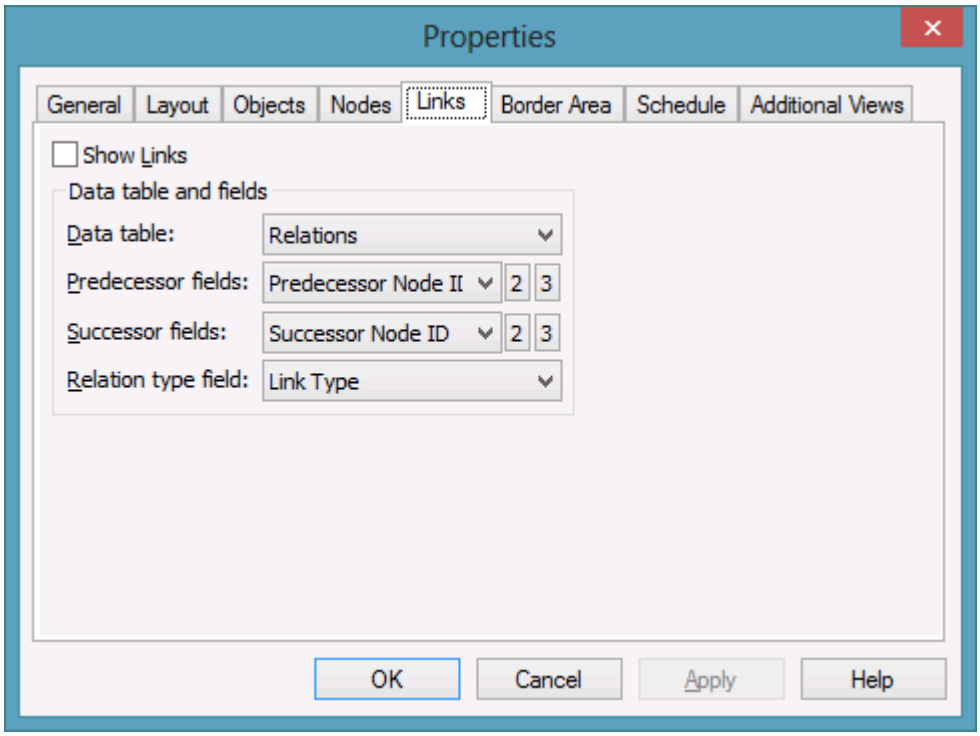
You can react to the below events:

- **OnLinkCreate**
- **OnLinkCreateComplete**
- **OnLinkDelete**
- **OnLinkDeleteComplete**
- **OnLinkLClickCltn**
- **OnLinkLDblClickCltn**
- **OnLinkRClickCltn**

> Specifying Links

On the **Link** property page you can choose whether the links are to be displayed and you can define or modify links.

You can specify the data fields in which the identifications of the predecessor / successor nodes and the relation types are to be stored. If the identification of a predecessor or successor node consists of more than one field, the corresponding link has to match this identification. That means that according to the ID of the respective node, a second or third field has to be selected if necessary. The first field is displayed by default. For setting a second or third field, click on the corresponding button and select the desired field from the drop-down list.



Furthermore you can define link appearances in the dialog **Administrate Link Appearances**. For each one you can select a filter, set the predecessor / successor layer, choose a line type, the predecessor / successor port symbols and the routing type.

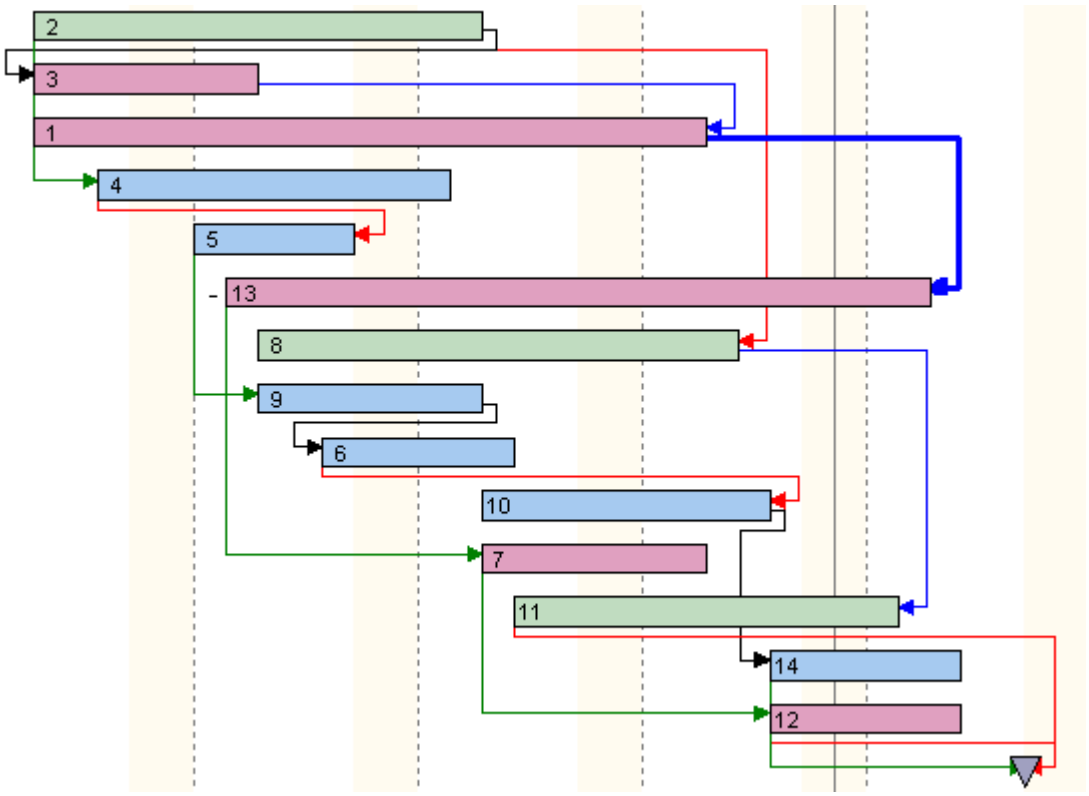
> Types of Links

On the **Link** property page you can select a data field in the combo box **Relation type field** from which the link type is to be loaded.

Link types:

- FF: Finish-Finish
- FS: Finish-Start
- SF: Start-Finish
- SS: Start-Start

The above data field allows to display the link type by the corresponding line routing.



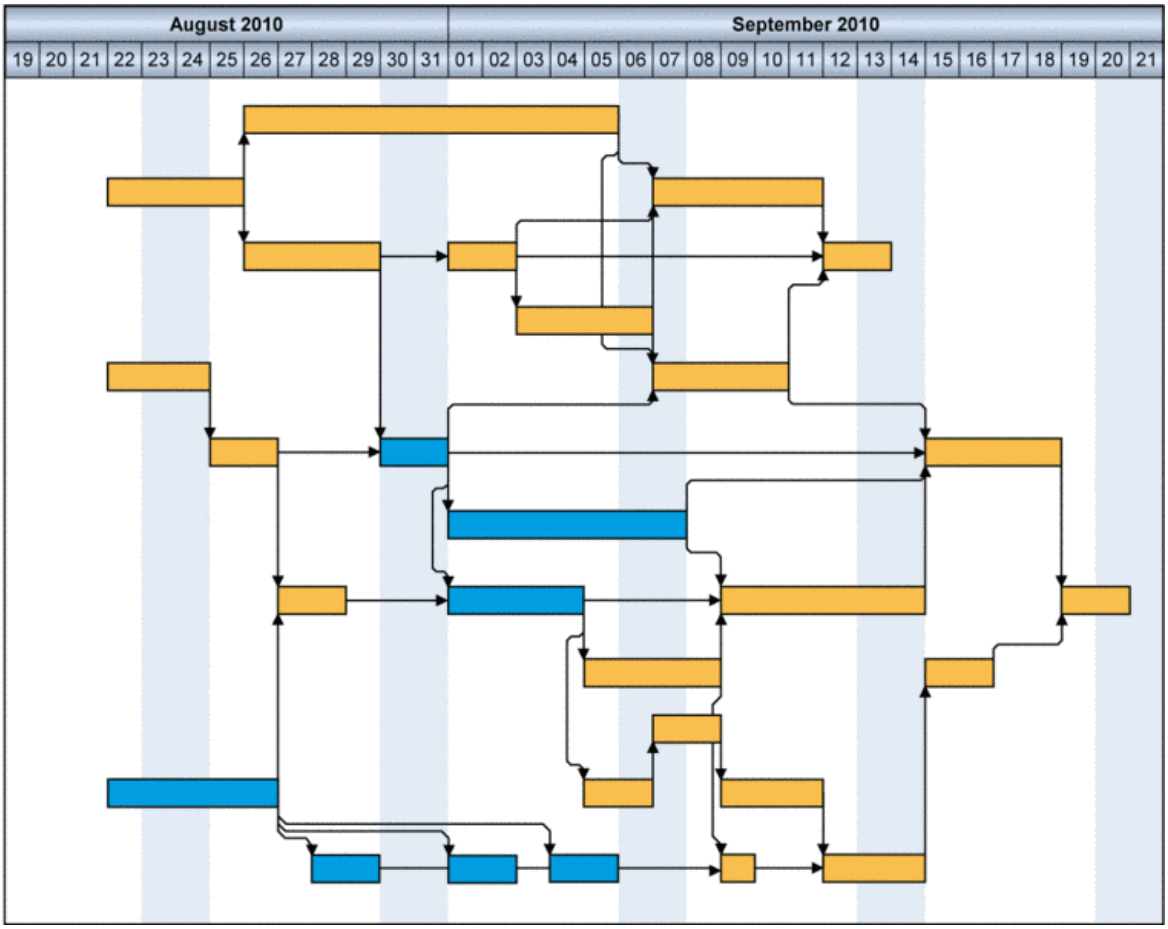
Examples of different link types

> Automated Layout

For the link routing a layouter is available to automatically display links in their optimum position. It can nest elbows so that line cross-overs are reduced to a minimum. The link routing is always unambiguous and allows the user to clearly distinguish where a link comes from and where it leads to.

The row heights in Gantt charts automatically adapt in order to create the required space to display all parallel horizontal link sections in a row.

Little slants are drawn in each elbow to indicate the direction into which the link is going.

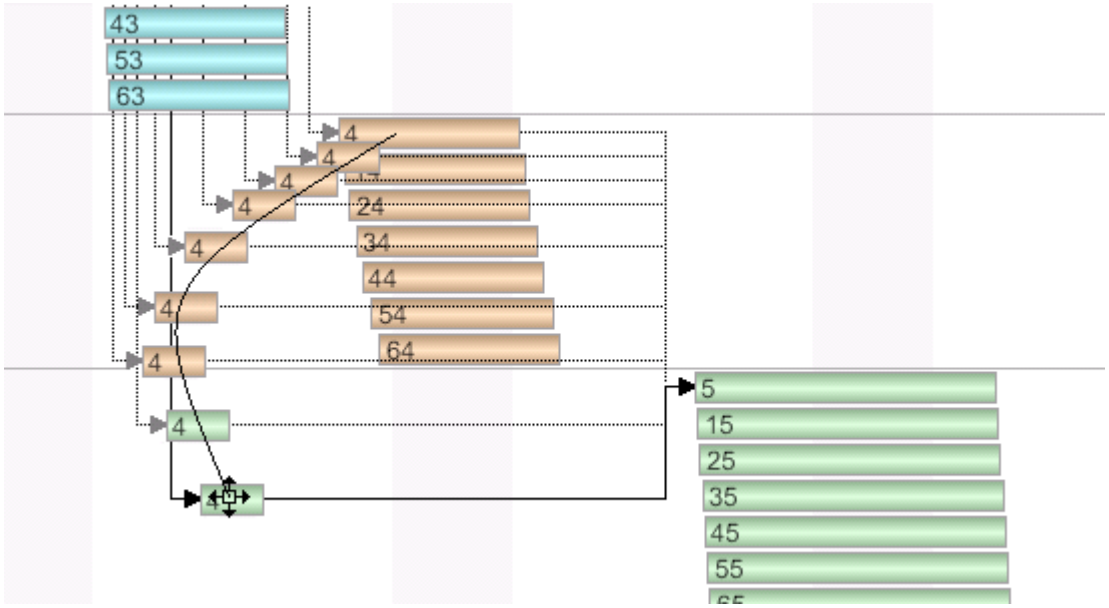

August 2010
September 2010
19 20 21 22 23 24 25 26 27 28 29 30 31 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21

3.19 Live Update

What is Live Update?

With the Live Update, being available from XGantt Version 5 onward, the consequences of a mouse interaction are visualized immediately during the action and not only after ending it.

Up to version 5, VARCHART XGantt used phantoms and the consequences for the overall planning were indicated by the Gantt graph as soon as the dragging action was finished by releasing the mouse key. The live update function, however, lets the planner recognize the results of the mouse action while interacting, since every mouse movement results in updating the node, meaning that the modifications are repeated constantly on the object thus resulting in a live update of the object and the chart. At any point during dragging a visualization of the node matching the respective cursor position with the appending links is shown.



Two ways of modyfiying data

There a two ways of changing and evaluating data:

- Modifications only relating to the particular object such as simple data changes, called **individual** changes in the following. Individual changes occur during each interaction.
- Modifications that do not only affect the particular object but also result in changing complete structures, such as grouping or optimizing, called **structural** changes in the following.

Structural changes can currently only occur while shifting nodes or groups, since only these can be summed up and arranged in structures.

Structural changes are carried out timer-driven (see also below: **Timer-driven Live Update**). **OldNode** and. **PreviewNode** are not planned.

After a structural change, the cursor is automatically scrolled under the cursor again (node tracking).

Interactions affected by Live Update

The interactions affected by live update are: shifting of nodes and groups and interactively creating nodes and links.

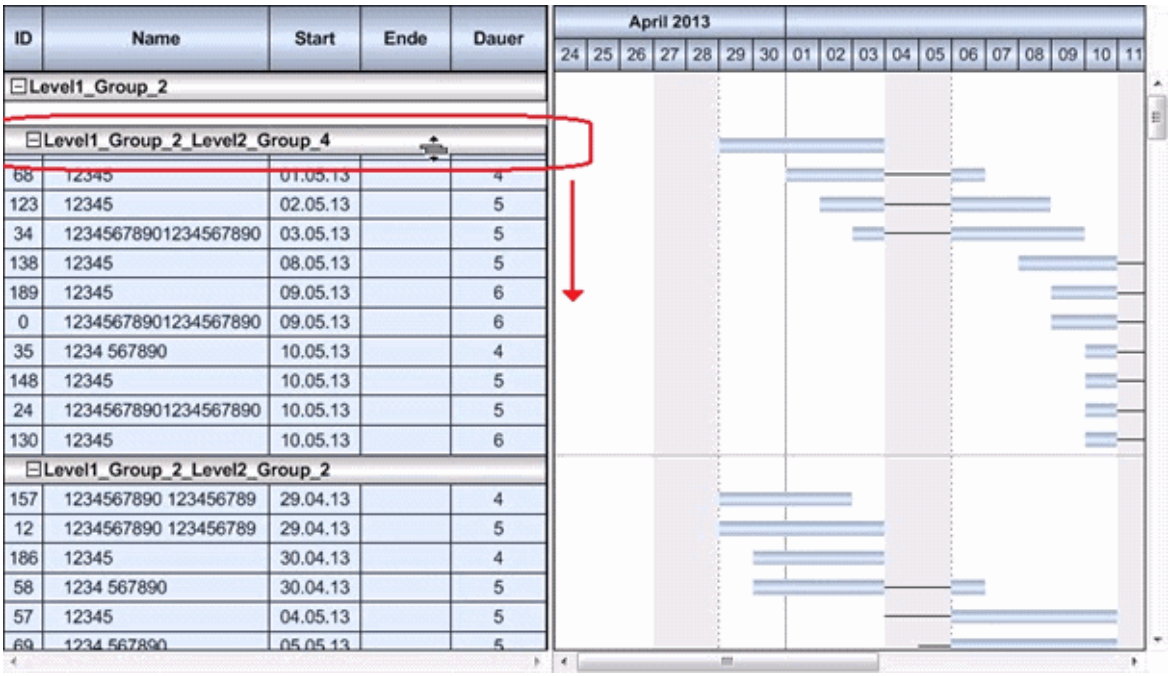
> Shifting of nodes and links in the diagram

Nodes and links can be freely moved in terms of optic, the horizontal and vertical position of the node being always adjusted to the cursor position, thus being always under the mouse cursor. Appending links, being drawn with linkrouting <orthogonal> or <straight> are dragged along accordingly. The linkrouting <distinguish> doesn't work in this case, so <orthogonal> is used. While changing the positon, the visualiziation of the nodes and links is also constantyl updated, meaning that filters and mapping are applied to the complete construct. An empty area will remain at the former node position, reinforcing the dragging effect. The node is dragged away from his former position. For this, the node with ist links is set to **VC_VISIBILITY=VC_NO** and copies of nodes and links are made and updated while dragging.

> Shifting of Groups

In VARCHART XGantt groups can be moved interactively within their levels. This is done by either shifting the summary bar or the group node vertically in the diagram or by vertically moving the respective table format in the table. This structure modification equals a manual sorting, having no equivalent in terms of data, hence no data are modified. After the modification will be done, the shifted summary bar/group node or the shifted table format respectively will be scrolled back under the cursor again automatically, this scroll behavior being called group tracking here.

In the diagram area, a VARCHART node phantom with real presentation of the summary bar/group node is used and in the table area a VARCHART node phantom with real representation of the table box. The real representation will remain unchanged since there will be no data modification during the dragging interaction.



Timer-Driven Live Update

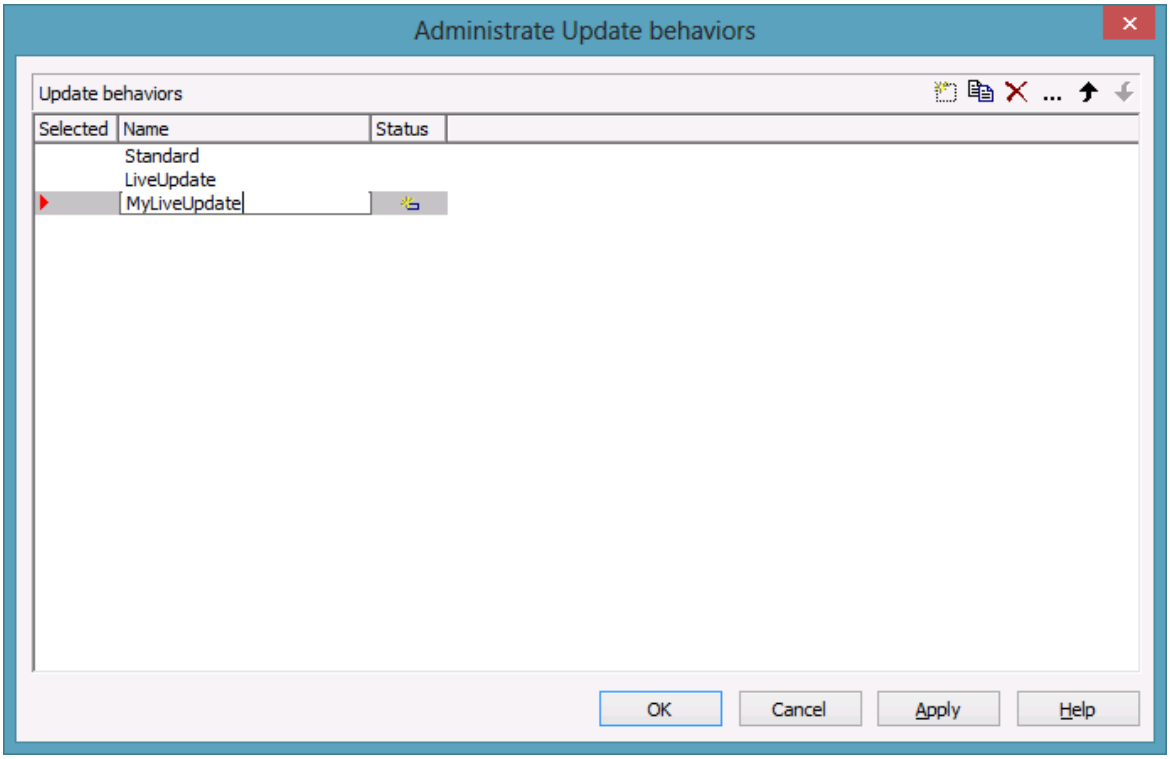
The whole chart gets quite unsteady by the constant (sometimes comprehensive) visual changes and the immediate changing of status without animation options could be confusing if not disturbing so that an alternative for the immediate change of status is called for. Updating caused by structural changes should not be constant but timer-driven. If the user shortly pauses during the mouse interaction, the structural modification will be only carried out after a short, but sigificant waiting time and the chart be updated. The graphic shown always matches the respective cursor position. Now the user can continue interacting since he is still moving the mouse while holding the key pressed. The structural changes are again impended until the user pauses again and again they will be only carried out and the chart be updated after a short, but sigificant waiting time. This is repeated until the interaction ends (releasing the mouse key). This technique ensures that the chart will remain rather steady.

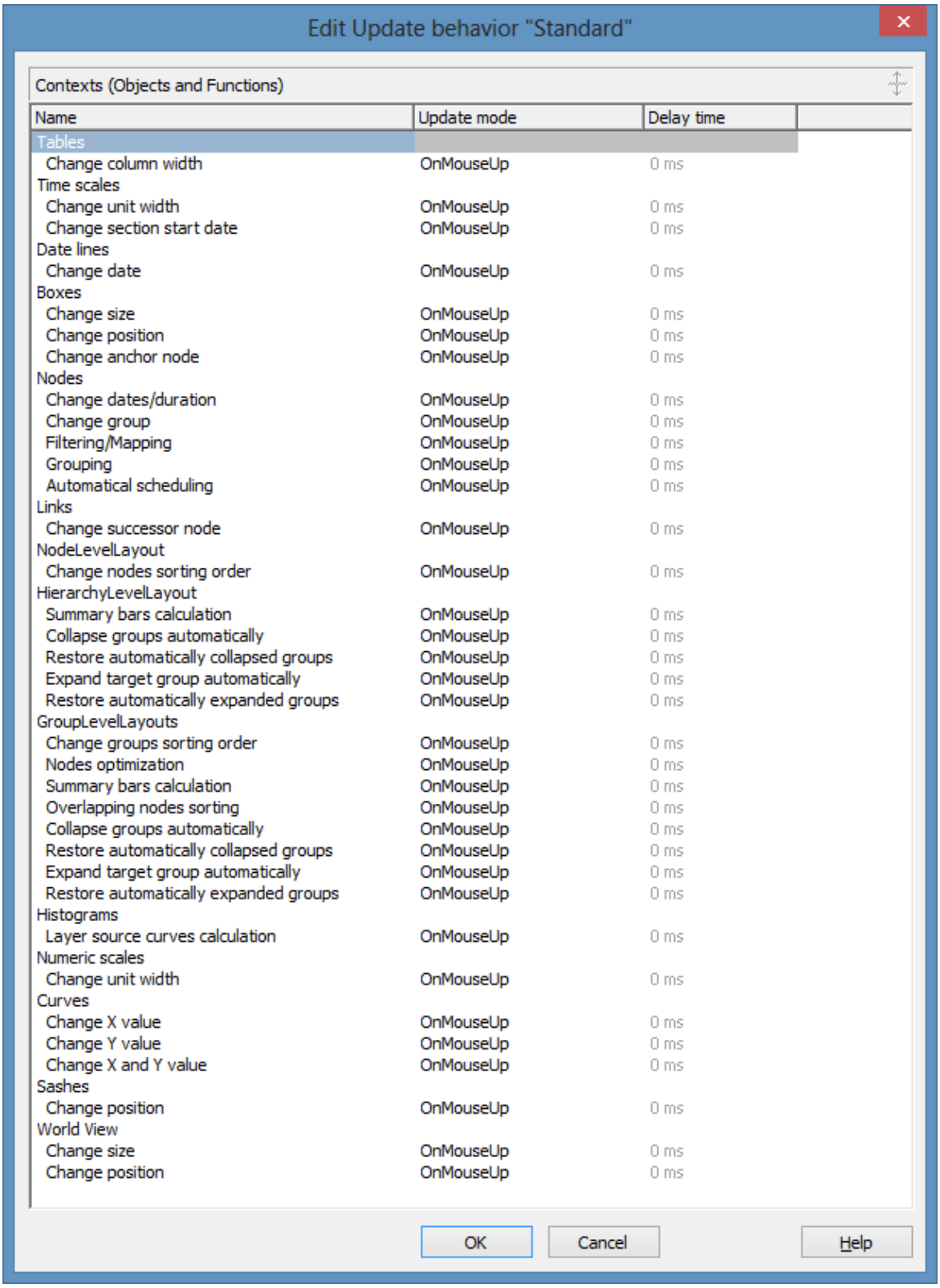
Setting up Live Update in VARCHART XGantt

> At Design time

The live update settings can be made in the **Administrate Update Behavior** and the **Edit Update Behavior** dialogs at design time. VARCHART XGantt comes alread with the update behaviors **Standard** and **Live Update** the settings of which can **not** be customized by the user.

The user can, however, create individual update behaviors that can be customized at will in the two dialogs shown below.





Name	Update mode	Delay time
Tables		
Change column width	OnMouseUp	0 ms
Time scales		
Change unit width	OnMouseUp	0 ms
Change section start date	OnMouseUp	0 ms
Date lines		
Change date	OnMouseUp	0 ms
Boxes		
Change size	OnMouseUp	0 ms
Change position	OnMouseUp	0 ms
Change anchor node	OnMouseUp	0 ms
Nodes		
Change dates/duration	OnMouseUp	0 ms
Change group	OnMouseUp	0 ms
Filtering/Mapping	OnMouseUp	0 ms
Grouping	OnMouseUp	0 ms
Automatical scheduling	OnMouseUp	0 ms
Links		
Change successor node	OnMouseUp	0 ms
NodeLevelLayout		
Change nodes sorting order	OnMouseUp	0 ms
HierarchyLevelLayout		
Summary bars calculation	OnMouseUp	0 ms
Collapse groups automatically	OnMouseUp	0 ms
Restore automatically collapsed groups	OnMouseUp	0 ms
Expand target group automatically	OnMouseUp	0 ms
Restore automatically expanded groups	OnMouseUp	0 ms
GroupLevelLayouts		
Change groups sorting order	OnMouseUp	0 ms
Nodes optimization	OnMouseUp	0 ms
Summary bars calculation	OnMouseUp	0 ms
Overlapping nodes sorting	OnMouseUp	0 ms
Collapse groups automatically	OnMouseUp	0 ms
Restore automatically collapsed groups	OnMouseUp	0 ms
Expand target group automatically	OnMouseUp	0 ms
Restore automatically expanded groups	OnMouseUp	0 ms
Histograms		
Layer source curves calculation	OnMouseUp	0 ms
Numeric scales		
Change unit width	OnMouseUp	0 ms
Curves		
Change X value	OnMouseUp	0 ms
Change Y value	OnMouseUp	0 ms
Change X and Y value	OnMouseUp	0 ms
Sashes		
Change position	OnMouseUp	0 ms
World View		
Change size	OnMouseUp	0 ms
Change position	OnMouseUp	0 ms

Note: Please note that individual update behaviors for data driven objects (nodes, links and groups) can **only** be assigned by API.

> At runtime

The settings are made in the following objects:

- VcBox
- VcCurve
- VcDateLine
- VcGantt
- VcGroup
- VcLinks
- VcNode
- VcNumericScale
- VcTable
- VcTimeScale
- VcUpdateBehavior
- VcUpdateBehaviorCollection
- VcUpdateBehaviorContext
- VcWorldView

For further information see the API reference of this manual.

3.20 Localization of Text Output

By the event **OnSupplyTextEntry** you can edit texts of context menus, dialog boxes, info boxes, error messages and the names of months and days that appear during runtime, for example in order to translate them into different languages.

To do so, activate the check box **OnSupplyTextEntry events** on the **General** property page. Or set the property **EnableSupplyTextEntryEvent** to **True** to activate the event.

Example Code

```
VcGantt1.EnableSupplyTextEntryEvent = True
```

Then capture the **OnSupplyTextEntry** event and specify the text that you want to appear.

Example Code

```
Private Sub VcGantt1_OnSupplyTextEntry(ByVal controlIndex As _
                                VcGanttLib.TextEntryIndexEnum, _
                                TextEntry As String, _
                                returnStatus As Variant)
    Select Case controlIndex
        Case vcTXERibCW
            TextEntry = "Semaine"
        Case vcTXERibDay0
            TextEntry = "Lundi"
        Case vcTXERibMon8
            TextEntry = "Septembre"
        Case vcTXERibQuar2
            TextEntry = "2. Quart."
        End Select
End Sub
```

3.21 Maps

Maps are used to set certain properties in dependence on data, thus avoiding to define large numbers of filters.

By using maps you can for example assign background colors, patterns, pattern colors and more properties to layers in dependence on their data.

Maps consist of a list of mappings. Each mapping consists of a key and a value. Depending of the map type, the value can be a graphics file name, a pattern etc. The key is a possible entry in a data field. At runtime, the keys are compared to the actual contents of the adressed data field and if they match, the value for the adressed graphic property is applied.

If there are more than two columns, more than one value is assigned to one key.

Example: In the map, the key "A" is assigned to the value "green". If the map is applied and some node field contains the value "A", the color green is assigned to this node (as background color of its layer, for instance). As a second value, a legend text could be assigned saying "finishes in time".

So, as a basic principle, the field values are compared to the keys of the map. If they match, the map value(s) are used.

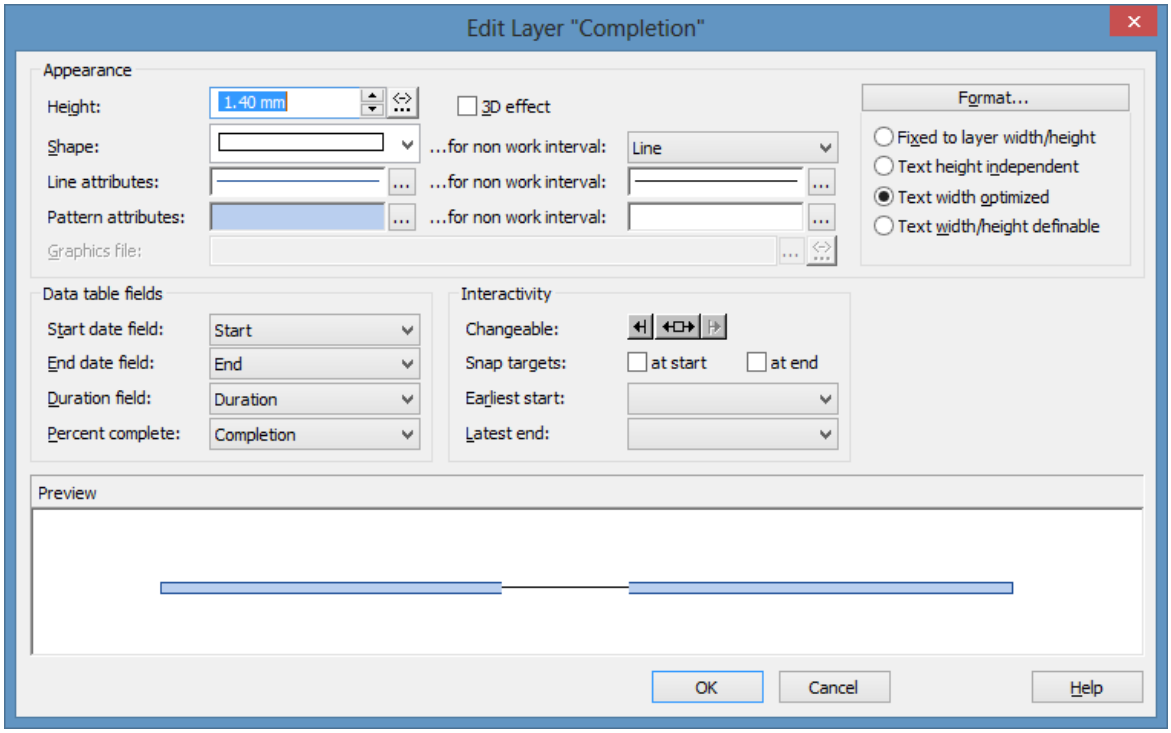
By using filters instead of keys you can specify more complex mappings. Basically, the concrete keys are interpreted first and only if they do not apply, the filters are interpreted.

> Example: Background color of layers

In the following example, the background color is assigned in

dependence on the node data by using a map.

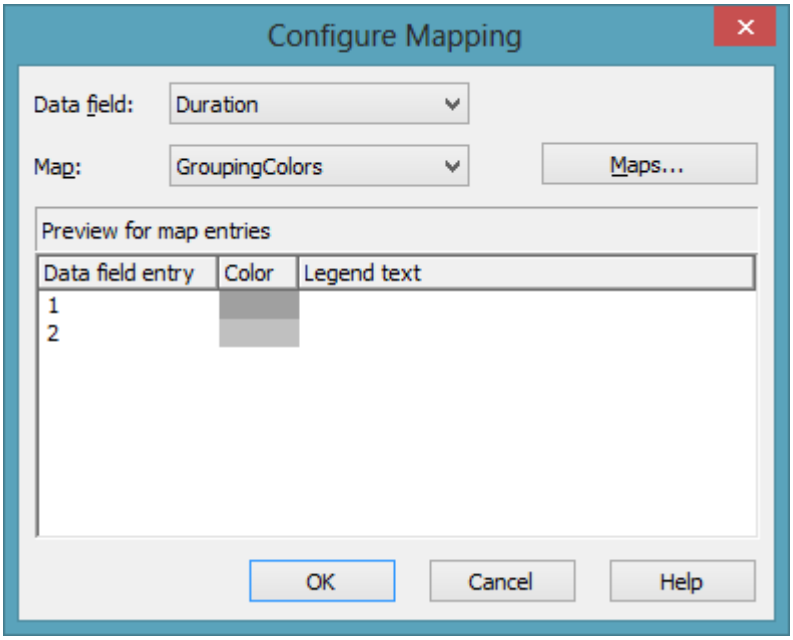
In the **Edit Layer** dialog box, click on the button next to the **Background color** field ().



You will get to the dialog box **Configure Mapping**.

> Configure Mapping

The **Configure Mapping** dialog lets you assign a data field of a node to a map, so that the value in the data field can be compared to the keys of the map. Thus the desired property, in our example the background color of the layer, is specified data- dependent. If the attribute shall not be dependent on only one single value but on a range of values or even more complex criteria, you can create a filter which can be selected in the **Edit Map** dialog instead of a single value. This filter will then be displayed in the **Configure Mapping** dialog in the list of the data field entries.

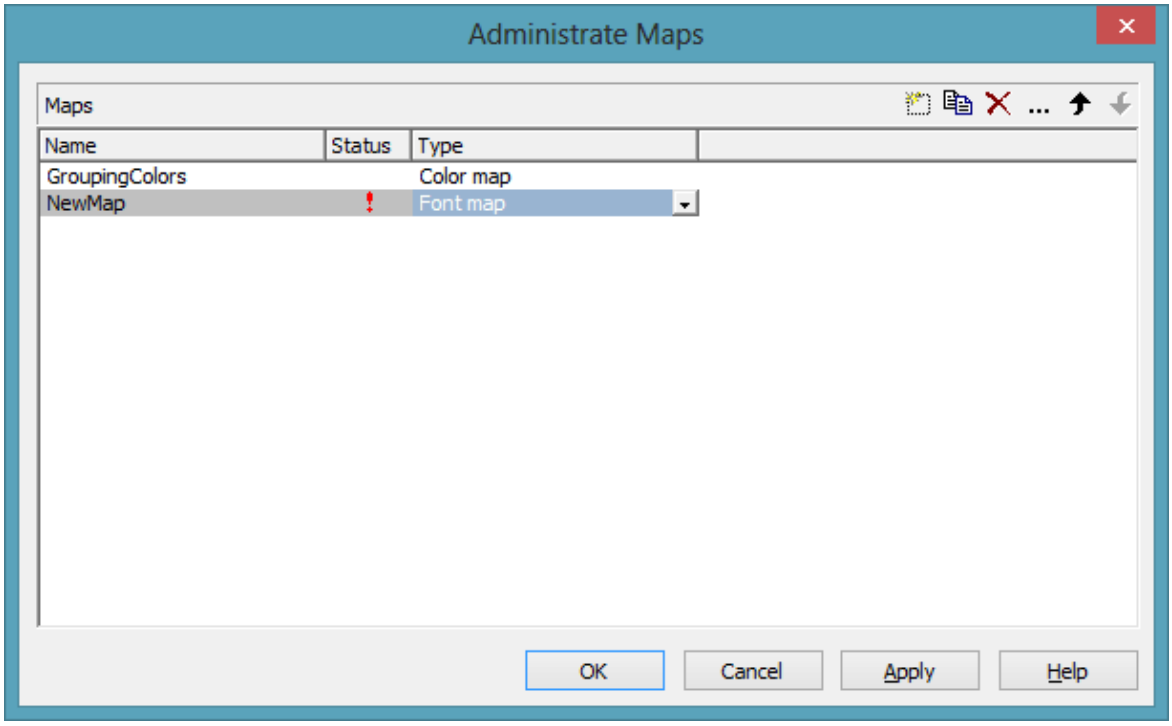


To configure a mapping, please select a node **Data field** at the top of the dialog, the values of which shall be compared to the key values of the map. From the field below, select an appropriate **Map**. (Only those maps are available which match the attribute selected in the **Edit Layer** dialog. Because in our example you have selected the background color, only maps of the type "Color map" are displayed). After having selected the map, ist contents becomes visible in the preview of the dialog. If there isn't a map to select, please create one as described the chapter below.

> Administration of Maps

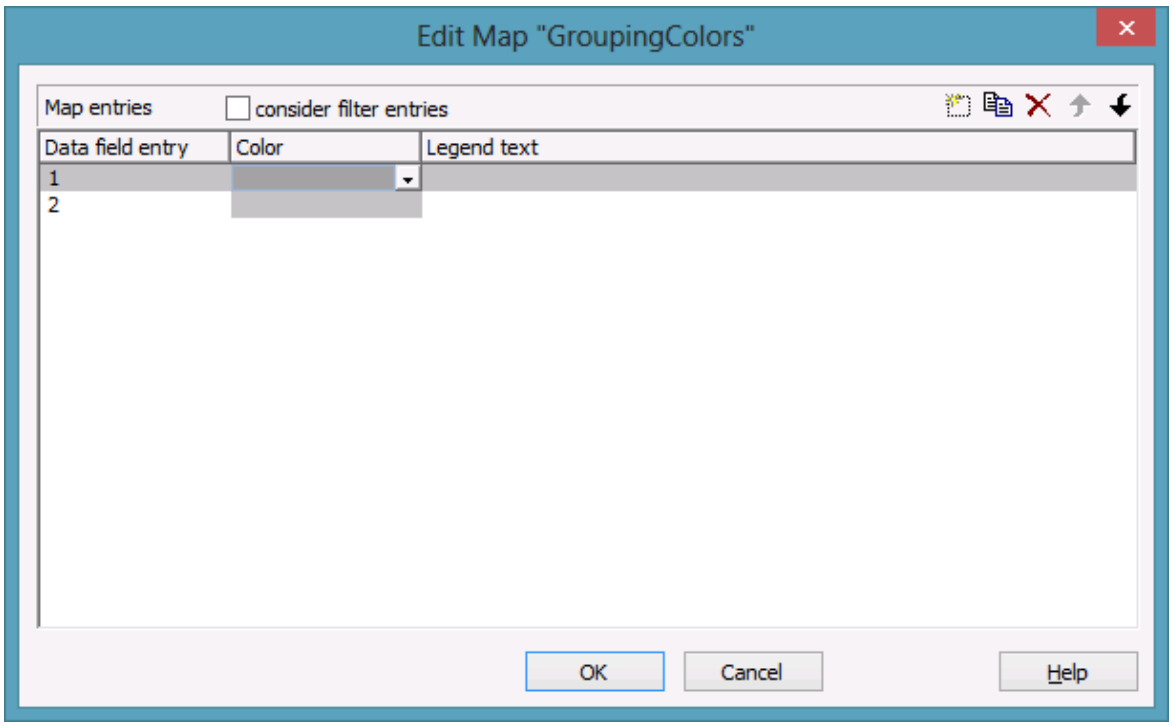
In the **Administrate Maps** dialog which can be invoked by clicking the **Maps** button or by clicking the **Maps** button of the **Objects** property page, you can modify the name and the type of a map by directly entering the corresponding data fields. By clicking the corresponding buttons on the right at the top of the window, you can also create, copy, edit or delete maps.

You can choose between different types of maps, according to whether colors, patterns, graphic files, fonts, lengths or numbers are to be allocated to data field contents.



> Editing Maps

To edit a map, mark it in the table and click on the button ... above the table. The **Edit Map** dialog box will open.



Of each key (=data field entry), the table shows its corresponding values, which, depending on the map type, in our example are the color and the legend text assigned.

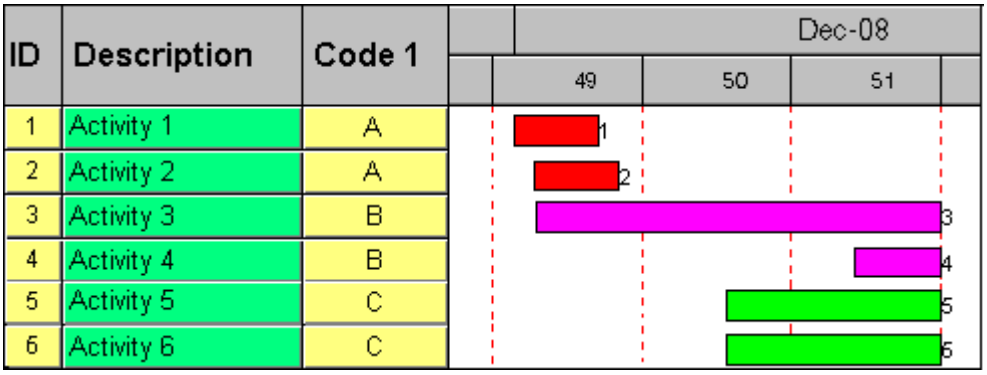
By the buttons right-hand at the top you can create, copy or delete keys (map entries) or modify their position in the table.

If you have ticked the check box **consider filter entries** not only the single values from the list of data field entries are considered as keys but also the filters which can be selected from the drop down list. Thus you can not only specify a single value as key but also more complex criteria.

In a map you can create 150 map entries at maximum. If you need more map entries, please create a new map, e. g. as a copy of the one being edited.

> Example

The below example shows a layer where activities that have a map entry = "A" are displayed in red, activities of map entry = "B" are displayed in pink, etc. The default background color is gray. It is used for activities with no data field entry or with a data field entry that is not defined in the map.



For further details please read the chapters "Property Pages and Dialog Boxes".

> Adjusting the Map during Runtime

You can modify maps even at runtime by using the **VcMap** methods. This way you can enable the user to modify your default settings by a dialog generated by your own code.

3.22 MultiState Fields

What are MultiState Fields?

It is possible in the table section to display different contents of data fields as different graphics by using maps and graphical fields. MultiState fields are an enhancement of this principle, where a click on a picture results in a change of state of the associated data field. MultiState fields are a comfortable way to edit data fields that can adopt a final number of different states. This is why multState fields can only work if the module **Data Editing** was licensed.

> The Way they Work

A click on the field triggers the search for the next picture in the map that differs from the present one. The corresponding value (i.e. the key in the map) will be assigned to the data field. If, apart from the map, another graphics file was set as a default, it will also be considered when the map is searched through. If the default picture appears, an empty string will be set to the data feld. In other respects the default picture will appear, if in the data field a value occurs that does not equal a key in map.

A most simple application of multiState fields are boolean data fields, which, for example, display the values **true** and **false** by check boxes that show or or do not show a check. When clicking on the present state, the picture will change to the opposite state and the value of the corresponding data field will turn from **true** to **false** (or vice versa).

> Instructions for Programming

- Keys in the map that point to the same graphics file should be placed consecutively. This is the only way to have the same graphics file displayed just once when the map is searched through. This is because on a click, the next picture file will be selected which is different to the picture presently displayed. For example, you can link the keys **true**, **t** and **True** to the same graphics file. If the file is displayed, a different file will be displayed on the subsequent click. So displaying the same graphics file for three times is avoided.
- For the same reason, you should put all keys at the beginning of the map, that point to a graphics file equal to the default graphics file.
- If the same graphics file consecutively appears in the map, the value written to the data field will always be the first key. If **true**, **t** and **True**

were put consecutively in the map (pointing to the same graphics file), always **true** will be stored to the data field, but never **t** or **True**.

- MultiState fields only change their state if editing is allowed (see the corresponding VcGantt properties <**InPlaceEditingOnGroupsInDiagramEnabled, InPlaceEditingOnGroupsInTableEnabled, InPlaceEditingOnNodesInDiagramEnabled, InPlaceEditingOnNodesInTableEnabled**).
- To avoid the pictures to be displayed in different sizes, the height of a graphics field should be set to a value unequal to 0 mm (see dialog **Edit table format** in the VARCHART XGantt property pages).

For more information on graphics files and maps please read the chapters **The "Edit Table Format" Dialog Box** and **Maps** in the User's Guide and the documentation of the VcGantt property **FilePath** in the Reference Manual.

3.23 Node (Activity)

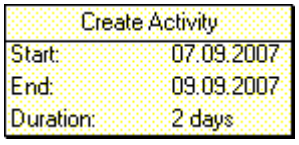
A node (activity) represents a record of the Maindata table. Nodes can be loaded by the programming interface or interactively created by a user.

> Creating Nodes

On the **Nodes** property page you can specify whether the user should be able to

- create new nodes by dragging the mouse (in the **Mode: Create Node**) (**Allow new nodes**)
- create new nodes by double-clicking (**New nodes via double-click**),
- directly edit new nodes via the **Edit Data** dialog box (**Edit new node**).

If a new node is created by dragging the mouse at run time, the **Create Activity** box will appear that indicates the start and end dates and the duration of the new node.

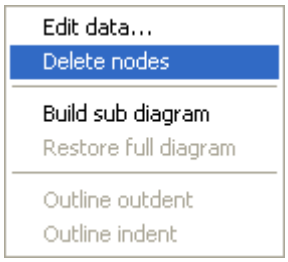


As soon as the mouse button is released, the **Edit Data** dialog box will appear, if the check box **Edit new node** on the **Node** property page was ticked before. The box displays the data of the new node, which can be edited. When a node is created interactively, the application is notified by the events **OnNodeCreate** and **OnNodeCreateCompleteEx**.

Nodes can alternatively be created by the API method **InsertNodeRecord**.

> Deleting Nodes

To delete a node in run mode, position the cursor on the node to be deleted and press the right mouse button. The below context menu will appear:



Select the **Delete nodes** option.

When a node is deleted interactively, the application is notified by the events **OnNodeDelete** and **OnNodeDeleteCompleteEx**.

Nodes can alternatively be deleted by the API method **DeleteNodeRecord**.

> Further Settings to Nodes

Beside, on the **Nodes** property page you can set:

- The data fields in which the data of start, finish, and duration of interactively created nodes is to be stored.
- Whether workfree periods are to be highlighted. In rectangle layers this will be indicated by a solid line.
- Whether calendars are to be assigned to the nodes. The influence of calendars becomes visible when nodes are moved and when durations are calculated. When moving activities, their start and finish dates will not be placed on workfree days. When calculating durations, workfree periods will be taken into account. By default, a five days' calendar ("WeekCalendar") is defined.
- If an individual calendar is required for a node, you can define a data field to store the name of the calendar.
- Whether a user is allowed to move more than one marked node at a time.
- Whether a marked node can be moved as a whole, i.e. including all its layers.

> Events

You can react to the following events:

- OnNodeCreate
- OnNodeCreateCompleteEx
- OnNodeDelete
- OnNodeLClick
- OnNodeLDblClick
- OnNodeModify
- OnNodeModifyComplete
- OnNodeRClick
- OnNodesMarkEx

3.24 OLE Drag & Drop

OLE Drag & Drop operations in VARCHART ActiveX are compatible to the ones in Visual Basic. Methods, properties and events show identical names and results as the default objects of Visual Basic.

Via OLE Drag & Drop activities or subdiagrams can be moved. The drag & drop mode is either started automatically or can be started manually by the VcGantt method **OLEDrag**.

> OLE Drag Mode

The OLE drag mode allows you to drag a node beyond the limits of the current VARCHART ActiveX control. There are two options:

- **Manual:** In this mode you need to invoke the method **OLEDrag** to trigger dragging the node.
- **Automatic:** In this mode dragging a node beyond control limits will be started automatically.

When starting the OLE Drag & Drop operation, the **DataObject** is provided with the source component's data and the **effects** parameter is set in order to trigger the **OLEStartDrag** event, as well as other events of the source. This allows you to control the operation e.g. to add other data formats.

VARCHART ActiveX by default uses the clipboard formats CF_TEXT (corresponding to the vbCFText format in Visual Basic) and CF_UNICODETEXT(for Windows NT 4.0/2000/XP; Visual Basic: 13) which both can be retrieved easily. It is the same data format as used by CSV files .

While dragging, the user can decide whether to move or to copy the object by using or not using the **<Ctrl>** key.

> OLE Drop Mode

Via the OLE drop mode you can enable a node of a different VARCHART ActiveX control to be dropped on an active control.

There are three options:

- **None:** Nodes of a different component cannot be dropped on the active component.
- **Manual:** When dropping a node of a different component, you will receive the **OLEDragDrop** event that enables you to process the data received by the object dropped, e.g. to generate a node or to load a file. If the source and the target component are identical, you will receive either

the event **OnNodeModifyEx** or **OnNodeCreate** as with OLE Drag&Drop switched off.

- **Automatic:** The dropping will automatically be processed by the control, displaying a node in the place of the dropping operation, if possible.

> Events

If you do not wish to have the drag&drop operation performed automatically by the VARCHART ActiveX components, this is how you can interact with it:

After starting the OLE Drag & Drop operation the event **OLEStartDrag** is released by the source control. By this event you can add data formats to the passed **DataObject** and define the permitted drop effects (i.e. copy and/or move). After moving the object, in the target control an **OLEDragOver** event will be triggered, that allows to set the drop effect to **copy, shift** or **prohibited**.

Each **OLEDragOver** event in the target control will trigger an **OLEGiveFeedback** event in the source control, that allows to set the mouse cursor. If in the target control the **OLEDropMode** was set to **Automatic**, the **OLEDragDrop** event will be invoked when the user drops the object. If in the target control the **OLEDropMode** was set to **Manual** and the source and target component are not identical, it is your job to produce a result that corresponds to the drop effect. After the operation in the source control the **OLECompleteDrag** event is triggered. In case you changed the mouse cursor in the **OLEGiveFeedback** event manually you should reset it now.

Note: The source and the target control may be the same control. It is also possible that they are controls other than VARCHART ActiveX or do not even belong to your application at all. If you want to make sure that the source and target controls belong to your application, you can set a format by the **DataObject** method **SetData**. The format needs to be registered by the Windows API call **RegisterClipboardFormat** before it can be used. You can verify the existence of the format by the **DataObject** method **GetFormat** on the **OLEDragOver** and **OLEDragDrop** event of the target control.

If you want to provide the data in several data formats and if you want to want to avoid the effort of specifying all formats for the **DataObject**now, you can use the key word **Empty** for **SetData**:

dataObject.SetData Empty, myClipFormat

On a request for the existence of a format using **dataObject.GetFormat** the target application will answer **True**. A **DataObject.GetData** call to the

source control will trigger the **OLESetData** event which then allows to pass the desired formats.

When you want to drag & drop file names, the **DataObjectFiles** object becomes interesting. To drag a file name, you first have to define the file format **vbCFFiles** (resp. **CF_HDROP**) in the **OLEStartDrag** event using **dataObject.SetData Empty, vbCFFiles**. Now you can add files using the **DataObject.Files.Add** method. To drop a file name (e.g. from the Windows Explorer), first check the existence of the the **vbCFFiles** format using **DataObject.GetFormat**, then read the file names e.g. **DataObject.Files(i)**.

3.25 Resource Scheduler

The ResourceScheduler2 is a substantial enhancement of Resource-Scheduler1 (version 3.1). The different object types required for resource scheduling are now anticipated in data tables of their own, which was facilitated by version 4.0 of VARCHART XGantt. In contrast, ResourceScheduler1 merely allowed the different objects like tasks, operations, assignments and resources to be implicitly defined in the maindata table.

The below object types exist in ResourceScheduler2 and need to be defined in data tables of their own; resources may even be defined in up to 25 different tables:

- **Tasks**: These objects are composed by operations (see below) and hold basic properties such as the release date, the due date, priority and quantity.
- **Operations**: These objects can be assigned to resources (see below) by assignments (see below) and will receive the start and end dates of the processing time as a result of scheduling. Operations have a defined position within a sequence of their task and can be marked as "started". Beside, several different sequences of operations can be defined that represent mutually exclusive "routes" of processing. All operations of a route selected by the scheduling procedure will be scheduled.
- **Resources**: As their main features, these objects are part of a capacity curve and after scheduling, they also are part of a workload curve. Beside, they time the operations that they have received (timing resource). Therefore, in order to be scheduled, an operation needs to be assigned to a resource. Beside a timing resource, also work and material resources can be assigned to an operation. Another essential feature of a timing resource is its ability to be grouped on multiple levels. A timing resource may belong to different groups at one time.
- **Assignments**: These objects are the links between operations and resources, that allow to specify a factor for the quantity to be multiplied or divided. When groups of timing resources are scheduled, the assignments are marked correspondingly and additional assignments are generated for each single resource, so that they can be scheduled and displayed in VARCHART XGantt.
- **Links**: These objects describe the sequence of tasks, i.e., preceding tasks have to be finished before the succeeding ones can start.

Survey of the Objects and Their Properties

Task Table	
TaskDataTableName	Name of the task table
TaskDueDateFieldIndex	Date, up to which a task has to be finished
TaskPlanningStrategyFieldIndex	Planning strategy: ASAP or JIT for single tasks
TaskPriorityFieldIndex	By assessing the importance of a job, the priority will bring forward a job or put it on hold.
TaskQuantityFieldIndex	Quantity to be produced by the task.
TaskReleaseDateFieldIndex	Date from which onward a task is allowed to be scheduled.
TaskResultEndDateFieldIndex	Scheduled date of finish
TaskResultPostEndDateFieldIndex	Scheduled date of post time finish
TaskResultPreparationStartDateFieldIndex	Scheduled date of preparation time start
TaskResultProcessingStepFieldIndex	Scheduled sequence number of the task
TaskResultProcessingTimeFieldIndex	Scheduled planning time of the task
TaskResultRouteFieldIndex	Scheduled route consisting of the resources available that work off the task
TaskResultStartDateFieldIndex	Scheduled start date of the task

Operations Table	
OperationDataTableName	Name of the operation table
OperationMaximumInterruptionTimeFieldIndex	Maximum time for which the operation is allowed to be interrupted while occupying a resource
OperationLoadPerItemFieldIndex	Load of resource per item
OperationOverlapQuantityFieldIndex	Overlapping time with other resources
OperationPostLoadFieldIndex	Post load of the operation
OperationPreparationLoadFieldIndex	Preparation load of the operation
OperationResultPostEndDateFieldIndex	Scheduled finish of the post time
OperationResultProcessingTimeFieldIndex	Scheduled processing time of the operation
OperationResultPreparationStartDateFieldIndex	Scheduled start date of the

Operations Table	
	preparation time
OperationResultSelectedTimingResourceIDFieldIndex	Determined ID of the timing resource
OperationResultStatusFieldIndex	Error or warning state
OperationRouteFieldIndex	Route to which the operation belongs
OperationSequenceNumberFieldIndex	Sequence of the operation within the route
OperationStartLockDateFieldIndex	Fixed start date
OperationTaskIDFieldIndex	Task, to which the operation belongs
OperationWorkInProcessFieldIndex	Degree of completion of the operation

Resourcen Table	
ResourceCalendarNameFieldIndex	Name of the resource calendar
ResourceCapacityType	Finite or infinite capacities for all resources
ResourceCapacityTypeFieldIndex	Finite or infinite capacities for single resources
ResourceConstraintTypeFieldIndex	Condition for work and material resources
ResourceDataTableName	Name of the resource table
ResourceEfficiencyFieldIndex	Efficiency in %
ResourceGroupDataTableName	Name of the table of group resources
ResourceGroupIDFieldIndex	Group identity of the resource
ResourceNameFieldIndex	Name of the resource
ResourceResultLoadCurveNamePrefix	Curve to which the scheduled work load of work and timing resources is to be stored
ResourceResultStockCurveNamePrefix	Curve to which the scheduled stock of material resources is to be stored
ResourceSelectionStrategy	Selection strategy of resources
ResourceSoftConstraintStartDateFieldIndex	Date of status change of a resource from "hard" to "soft"
ResourceType	Type of resource
ResultProcessingStepCount	Number of scheduled tasks

Assignment Table	
AssignmentDataTableName	Name of the assignment table

Assignment Table	
AssignmentIsResultFieldIndex	Was the data record generated by the scheduling procedure?
AssignmentIsVisibleFieldIndex	Should the assignment be visible in the chart?
AssignmentLoadOrConsumptionFieldIndex	Value per item
AssignmentMaximumLoadFieldIndex	Maximum work load limit
AssignmentMinimumLoadFieldIndex	Minimum work load limit
AssignmentOperationIDFieldIndex	Operation assigned
AssignmentResourceSelectionStrategyrFieldIndex	ASAP or JIT for a single resource
AssignmentResourceIDFieldIndex	Resource assigned

Link Table	
LinkDataTableName	Name of the link table
LinkDurationFieldIndex	Minimum time offset
LinkPredecessorTaskIDFieldIndex	Predecessor task of the link
LinkSuccessorTaskIDFieldIndex	Successor task of the link

General Properties	
BaseTimeUnit	Separate time unit for resource scheduling
BaseTimeUnitsPerStep	Coarse or small steps for scheduling?
DataRecordEventsEnabled	Should DataRecord events be enabled?
DefaultOperationMaximumInterruptionTime	Maximum duration of a unique interruption for operations
DefaultResourceCalendarName	Default calendar for scheduling
FullUsageOfPlanningUnitsEnabled	Using up remaining capacities of resources
PlanningEndDate	End of the scheduling time span
PlanningStartDate	Beginning of the scheduling time span
PlanningStrategy	Planning strategy: ASAP or JIT for all tasks
Process	starting the scheduling procedure
ToleranceTimeOnASAPDueDates	Allowance to the due date
ToleranceTimeOnJITReleaseDates	Allowance to the release date
ToleranceTimeOnStartLockDates	Allowance to a locked start date
WorkInProcessType	Unit of the degree of completion
WritingDebugFilesEnabled	Should debug files be written?

After having set the properties of the table, the scheduling procedure can be started by invoking the method **Process**.

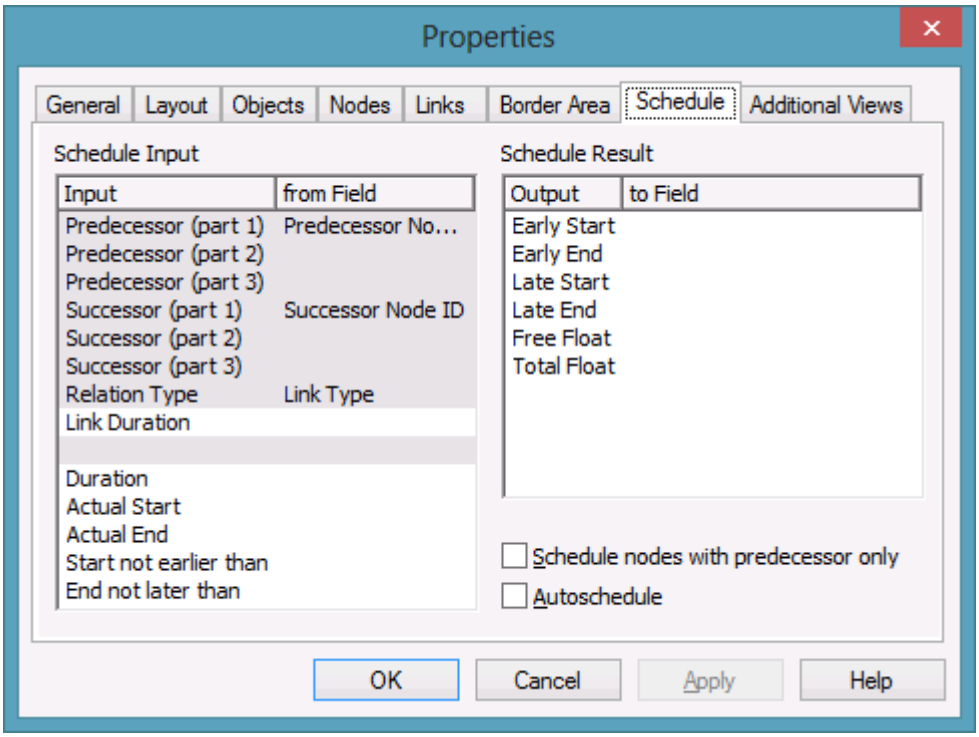
3.26 Schedule

The VARCHART XGantt Scheduler allows to perform simple date calculations. The desired project start and end dates are to be passed as parameters.

By the **Schedule** property page you can adapt the date calculation settings of VARCHART XGantt to your interface by specifying the data fields that you wish to use for input (**Schedule Input**) and for output (**Schedule Result**) of the scheduler.

The scheduler uses data fields of the respective nodes and links tables .

The key data for calculating the dates are the durations of activities, their logical dependencies and the project start. This data is used to calculate the early and late start and end dates as well as the total float and the free float. The **Predecessor** and **Successor** fields cannot be edited in the **Schedule Input** table. They merely display the settings of the **Links** property page.

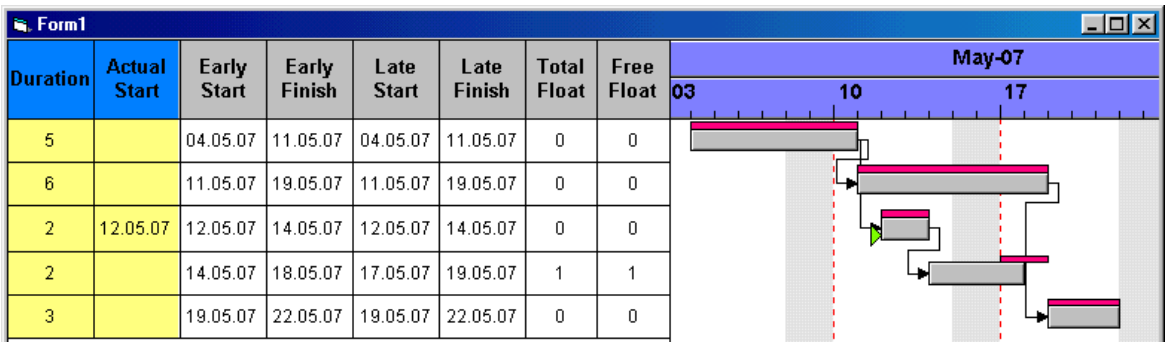


The results are stored to data fields of the interface. Available results are: **Early Start, Early Finish, Late Start, Late Finish, Total Float** and **Free Float**. To each of them you can assign a field from the list of fields specified in the data definition. All of the below examples were calculated for the project start on May 4, 2007, which you can set by the below API code:

Example Code

```
VcGantt1.ScheduleProject "04.05.07, 0"
```

The settings displayed above give the below graphical result:



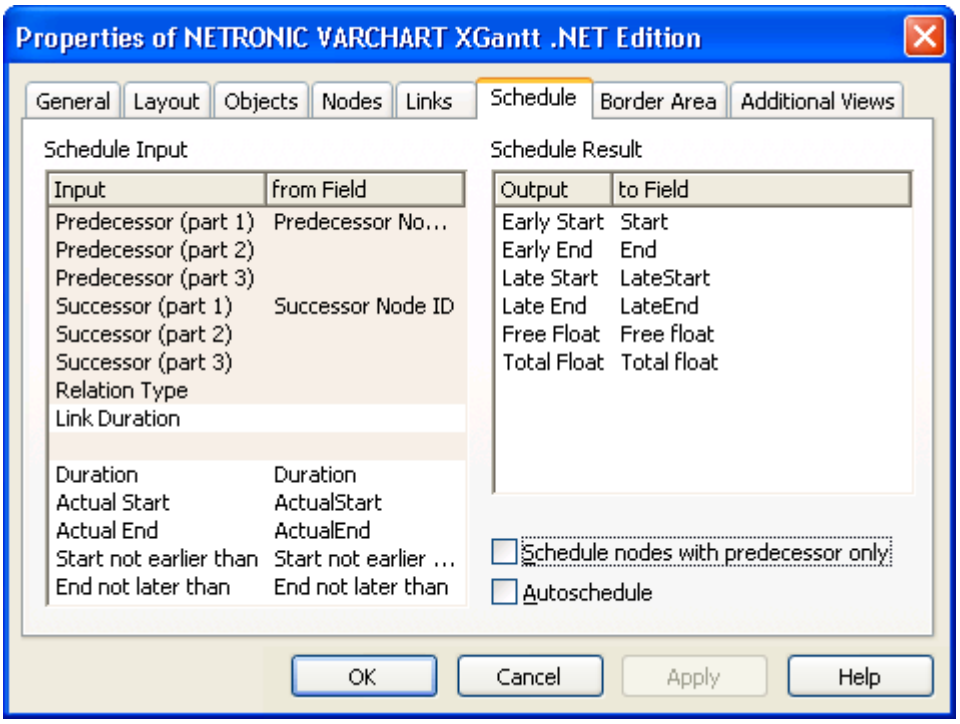
Duration	Actual Start	Early Start	Early Finish	Late Start	Late Finish	Total Float	Free Float
5		04.05.07	11.05.07	04.05.07	11.05.07	0	0
6		11.05.07	19.05.07	11.05.07	19.05.07	0	0
2	12.05.07	12.05.07	14.05.07	12.05.07	14.05.07	0	0
2		14.05.07	18.05.07	17.05.07	19.05.07	1	1
3		19.05.07	22.05.07	19.05.07	22.05.07	0	0

In the above example, the early pair of dates and the late pair of dates are displayed as a layer each.

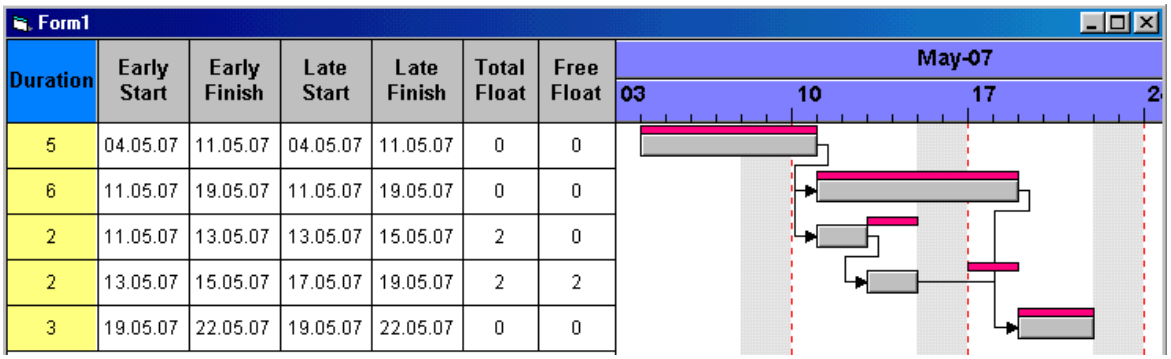
There are more ways to influence the date calculations of the VARCHART XGantt Scheduler.

1. You can set actual start/end dates. This way, the activities cannot be moved.
2. You can specify reference dates for the **Start not before** and **End not later than** conditions by allocating a field from the data definition to each value in the left-hand table on the **Schedule** property page.

The below diagram shows the settings that were made for the example following:

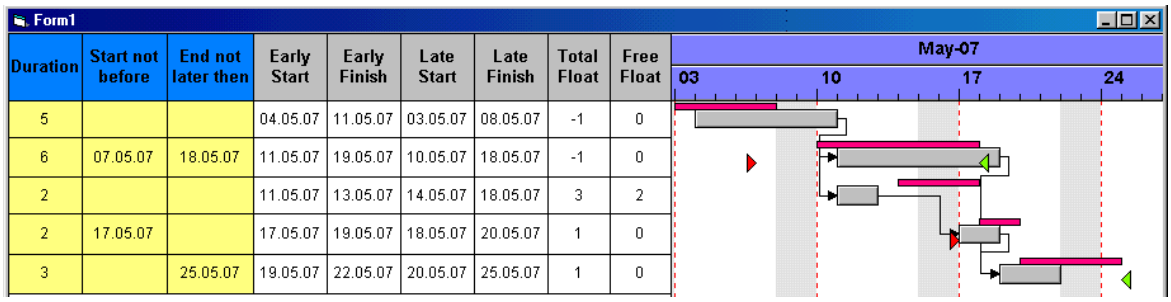


Setting the actual start of an activity will also fix the early and late dates. In the below picture, the actual start date set is marked by a green triangle.



Duration	Early Start	Early Finish	Late Start	Late Finish	Total Float	Free Float
5	04.05.07	11.05.07	04.05.07	11.05.07	0	0
6	11.05.07	19.05.07	11.05.07	19.05.07	0	0
2	11.05.07	13.05.07	13.05.07	15.05.07	2	0
2	13.05.07	15.05.07	17.05.07	19.05.07	2	2
3	19.05.07	22.05.07	19.05.07	22.05.07	0	0

Using the expressions **Start not before** and **End not later than** may or may not have an effect. In the below example, the date limits are marked by red and green triangles. Some do not have an effect on the date calculation. The end date restriction of the second activity entails a negative float for the two initial activities.



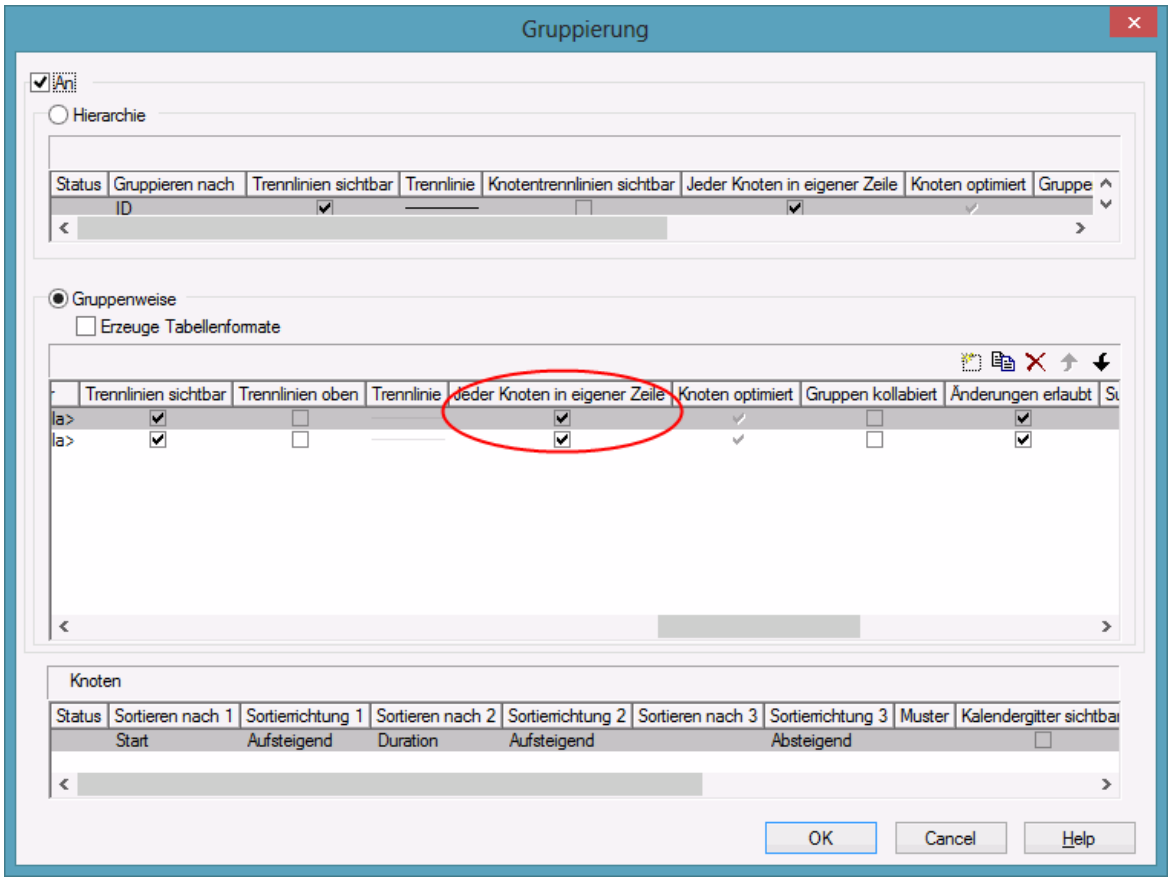
Duration	Start not before	End not later then	Early Start	Early Finish	Late Start	Late Finish	Total Float	Free Float
5			04.05.07	11.05.07	03.05.07	08.05.07	-1	0
6	07.05.07	18.05.07	11.05.07	19.05.07	10.05.07	18.05.07	-1	0
2			11.05.07	13.05.07	14.05.07	18.05.07	3	2
2	17.05.07		17.05.07	19.05.07	18.05.07	20.05.07	1	0
3		25.05.07	19.05.07	22.05.07	20.05.07	25.05.07	1	0

3.27 Sorting

Usually, applications require activities to be sorted according to certain criteria. Only those nodes can be sorted, that do not form part of a hierarchy, i.e. that are base nodes or belong to a group. So you will find setting options in places where you can set properties of group nodes and of base nodes. When sorting nodes, it makes a difference whether nodes are arranged in separate rows or whether several nodes are displayed in a single row.

Arrangement: Nodes in Separate Rows

If you wish the nodes to be arranged in separate rows, please invoke the **Grouping** dialog that you can get to by selecting the **Objects** property page and then **Grouping**:



In the center window, please tick the box **Nodes in separate rows**. Alternatively, you can set this feature by the API property **VcGroupLevelLayout.AllNodesInOneRow**.

In the window below which is called **Nodes** you can specify three data fields by which the activities are to be sorted when the diagram pops up. In

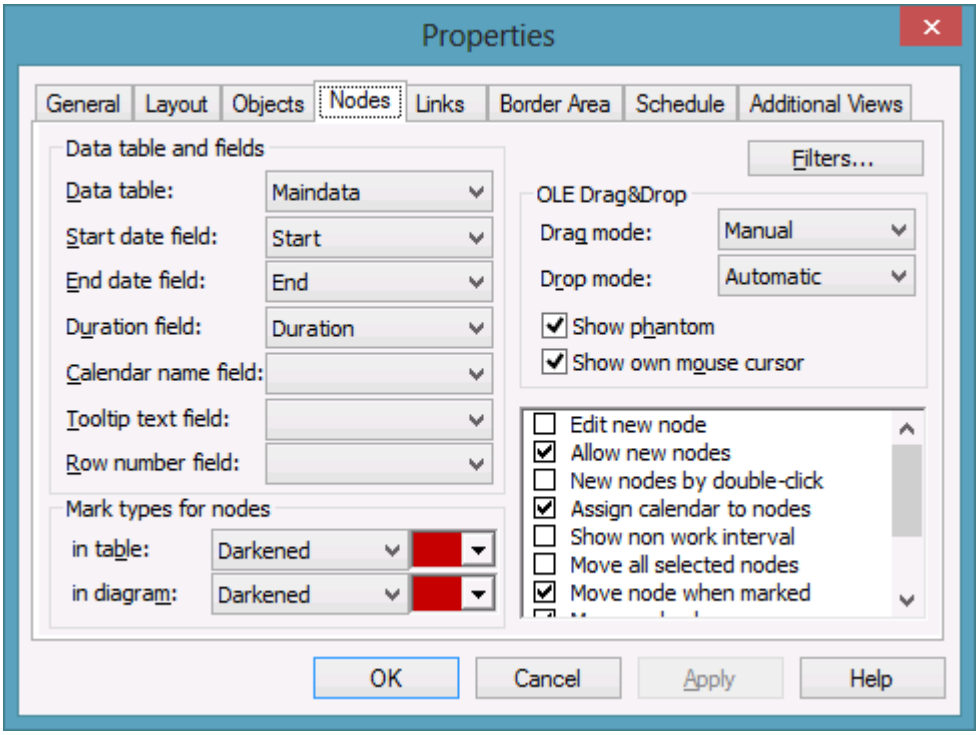
addition, you can select an ascending or a descending sorting order for each of the data fields.

If the activities are grouped, sorting will apply to the nodes of each group.

Beside, the below options for defining the appearance of the node line are available :

- Selection of a **Pattern**
- display, position and style of the **Separation Line**
- specify after how many activities a separating line should be drawn by entering a value in the field **Separation line step size**. If the activities are grouped, the counting will be done separately for each group.

Further sorting options can be set on the **Nodes** property page:



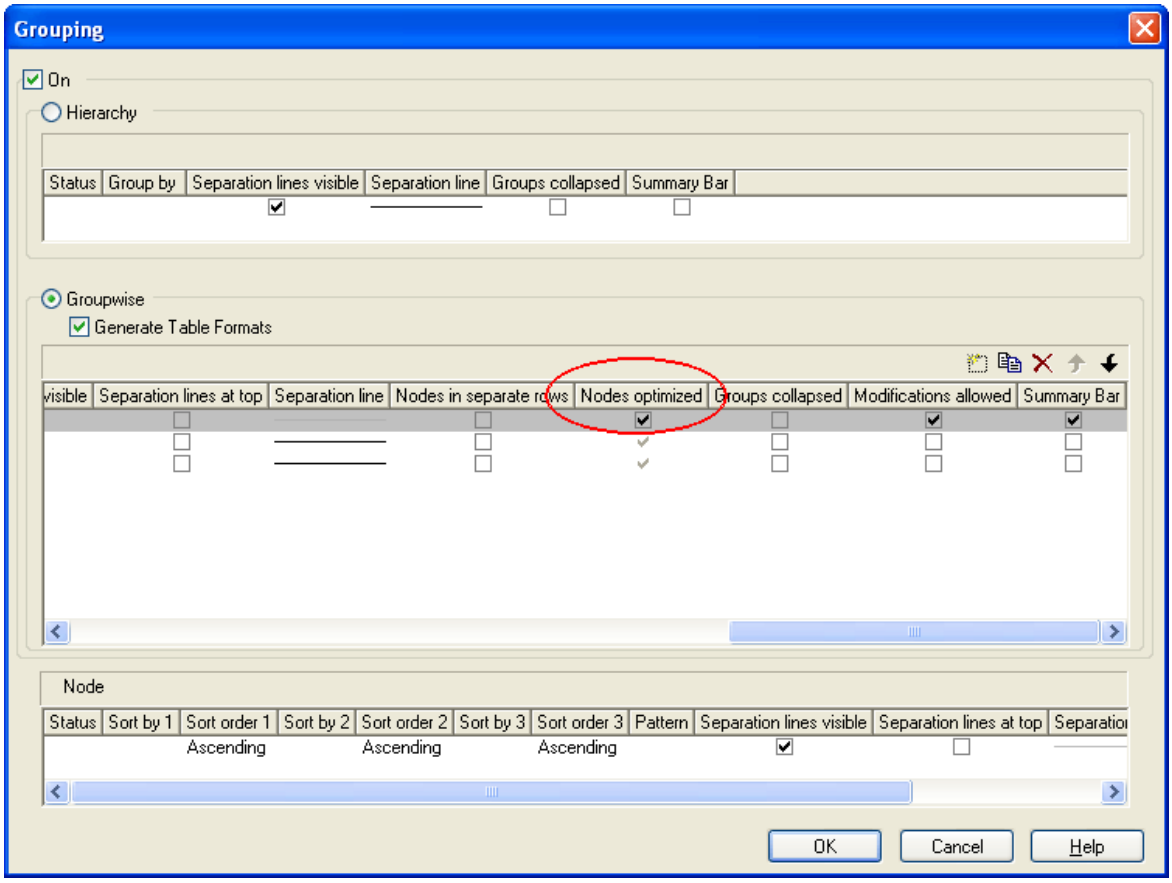
- You can select a data field to which the row numbers of the activities are stored. The **row number field** will not be updated until saving the data by the **Save As** method.
- By ticking **Moving a node vertically via diagram allowed** and/or **Moving a node vertically via table allowed** you can enable the user to modify the order of activities by dragging them to a different row . If an activity is moved to a different group, its grouping code and color will adjust to the new group. If an activity is comprises more than one layer, the **Shift** key has to be pressed in addition.

Note: Please note that the settings in the **Grouping** dialog and on the **Nodes** property page are only used to sort the data when the application is started. If you want to sort the activities later again, please use the method **SortNodes**. So an update of the sorting has to be invoked separately by this call.

Arrangement: Nodes of a Group in One Row

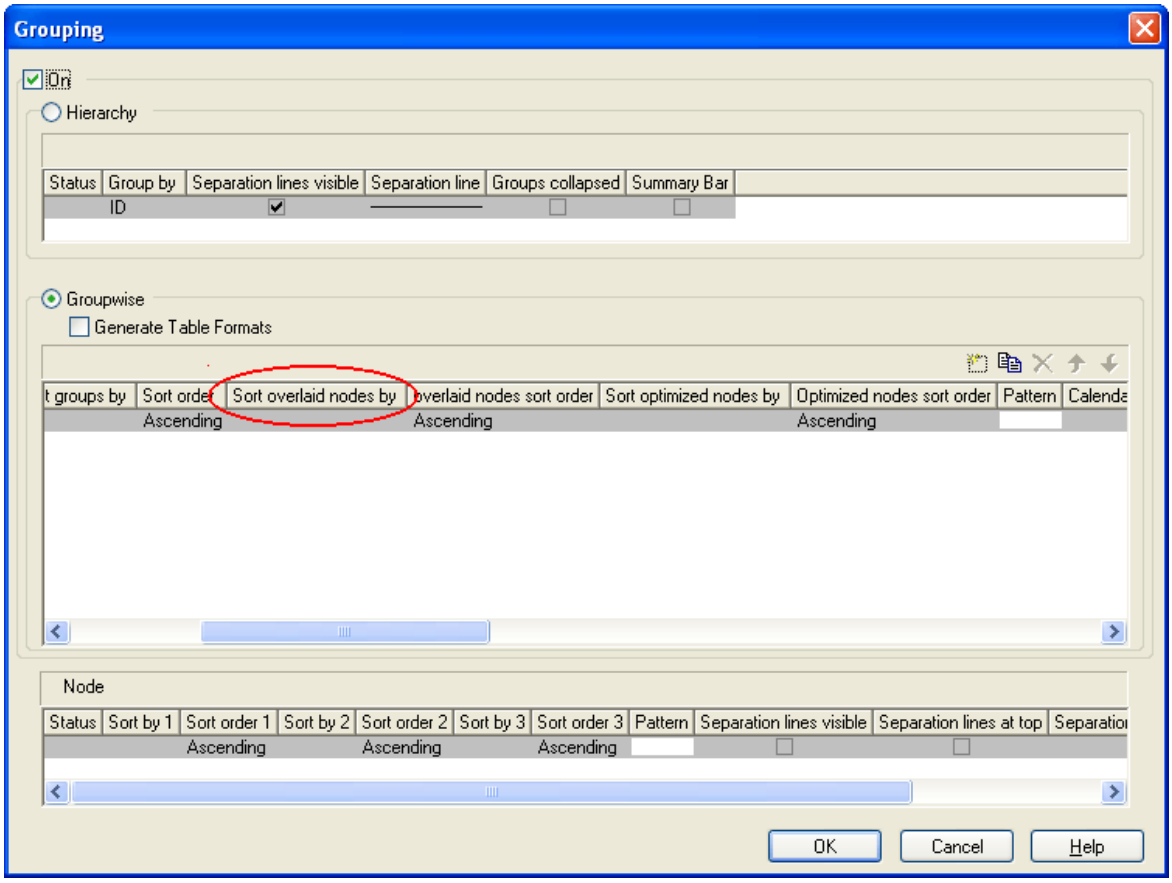
If several nodes (i.e. the nodes of a group) are put in a single row, you can assign a drawing priority (which is also a kind of sorting) to the nodes. Two different types of arrangement exist, the **overlapping** one and the **optimized** one, where the activities of one row either overlap each other or avoid overlapping by widening the row.

You can put several nodes in one row by unticking the box **Nodes in separate rows** in the **Grouping** dialog. By default, the adjacent field **Nodes optimized** will appear activated:



You can deactivate this check box which will entail the nodes of a row being displayed as overlapping. You can alternatively set this feature by the API property **VcGroup.NodesArrangedOptimized**.

The drawing priority of the nodes you can set by the field **Sort overlapping nodes by**:

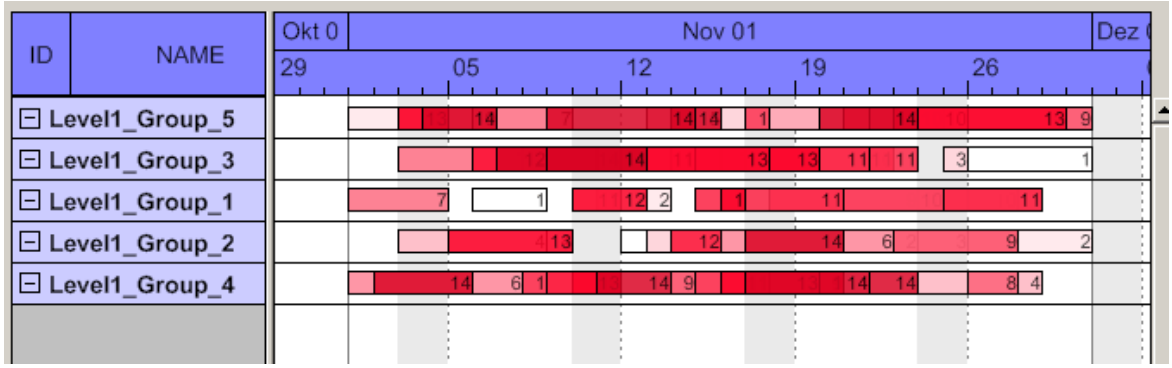


In analogy to overlapping nodes, you can sort optimized nodes by the field **Sort optimized nodes by**.

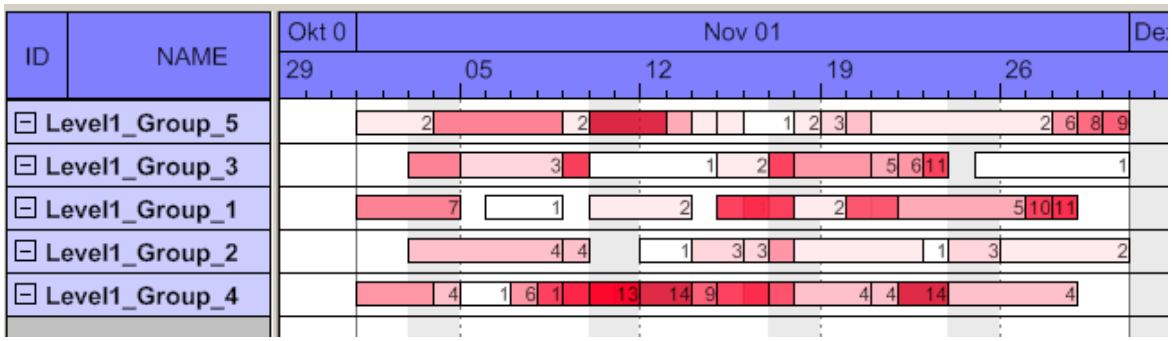
If you do not set a sorting priority, the nodes by default will be displayed in the order of their date and duration, the latest and shortest ones being drawn on top of the earlier and longer ones. The drawing priority can also be set by the API properties **VcLevelLayout.OverlaidNodesSortDataFieldIndex** and **VcLevelLayout.OptimizedNodesSortDataFieldIndex**.

You do not need to update the sorted nodes by a separate call, they will update automatically. Besides, by the adjacent field **Overlapping nodes sort order** you can assign an ascending or descending sort order. The sorting direction can alternatively be set by the API properties **OverlappingNodes-SortOrder** and **OptimizedNodesSortOrder**, respectively.

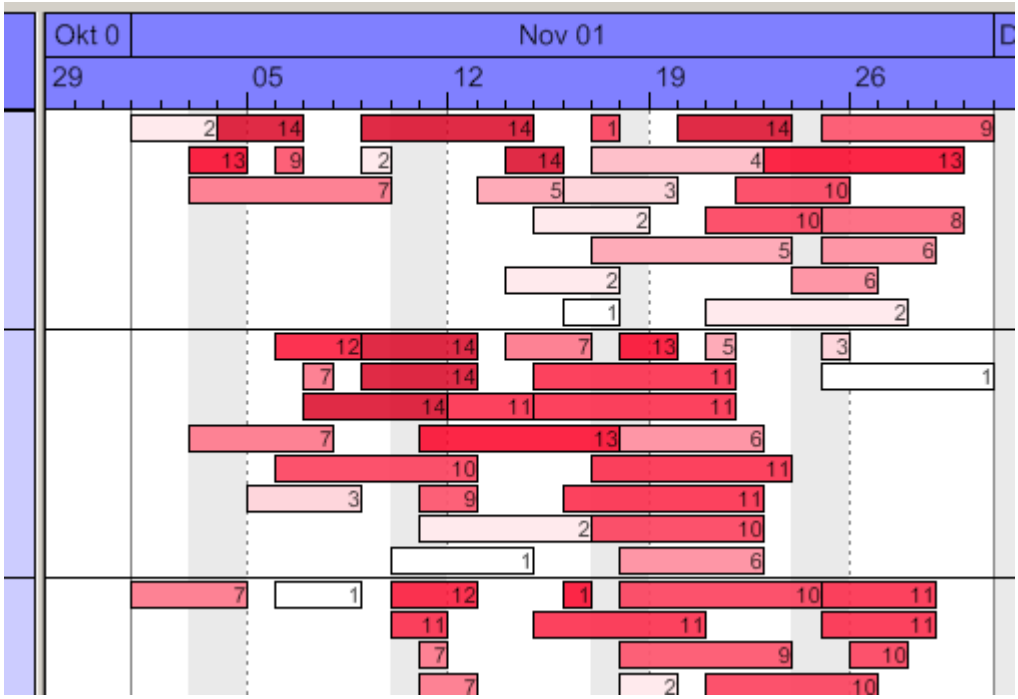
Below, some results of the setting are shown:



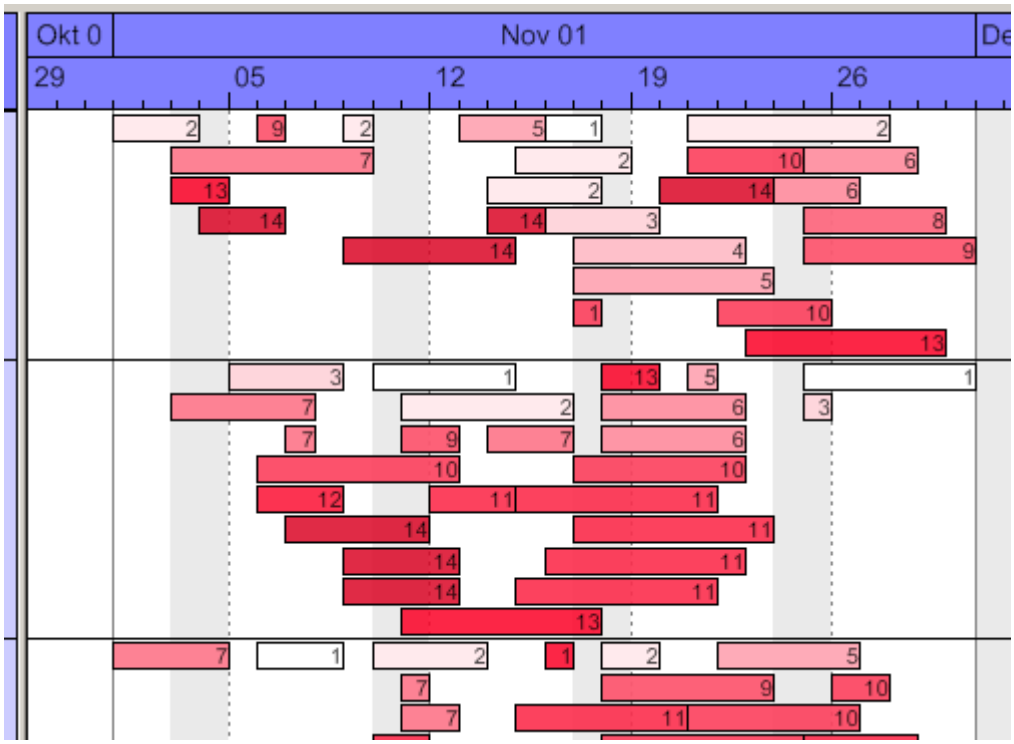
Overlapping node arrangement showing an ascending drawing priority of dark nodes (dark nodes drawn on top of light nodes)



Overlapping node arrangement showing an descending drawing priority of dark nodes (light nodes drawn on top of dark nodes)



Optimized node arrangement showing an ascending drawing priority of dark nodes (dark nodes drawn in the upper section of the row)



Optimized node arrangement showing an descending drawing priority of dark nodes (light nodes drawn in the upper section of the row)

3.28 Table

The properties of the table can be set by three different dialogs, that can be reached by the property page **Objects** and the button **Table**. The dialogs of the actual table features are named **Specify Table**, **Edit Table** and **Edit Table Format**. You can create several tables in the **Specify Table** dialog.

The table consists of six columns (default) that are only visible if they are assigned a width greater than 0. The rows in the table are defined by table formats. For each table format you can specify the font style, font color, background color, alignment and margins. Each format is applied in certain conditions:

- **StandardListCaption** for the table header
- **StandardList** for activities/rows.

In addition to the default table formats you can create table formats for which you can specify names and filters individually.

Table formats for a hierarchical arrangement:

The hierarchical arrangement can be set on the property page **Objects** by clicking on the button **Grouping**.

- **Hierarchy:** Format for hierarchical levels when expanded; the second field (usually the activity name) will be indented to display a lower level. A "-" indicates that the level can be collapsed.
- **HierarchyCollapsed:** Format for collapsed hierarchy levels. A "+" indicates that the level can be expanded.

ID	NAME	START
1	⊟ **SW Development**	**02.09.98**
1.2	⊞ ***Design&Concept***	***02.09.98***
1.3	⊟ **Coding**	**09.09.98**
1.3.1	Phase A (DB)	09.09.98
1.3.2	Phase B (GUI)	15.09.98
1.4	⊞ ***Testing***	***17.09.98***
1.5	Sales & Marketing	05.09.98
1.6	Delivery	24.09.98
1.7	Final Party	

*Picture above: The format **HierarchyCollapsed** is displayed in the row **Design&Concept** indicating a collapsed hierarchy level; the format **Hierarchy** is displayed in the row **Coding**, indicating an expanded hierarchy level.*

Table formats for a grouped arrangement:

A grouped arrangement can be set on the property page **Objects** by clicking on the button **Grouping**.

- **Subtitle:** for the headers of non-collapsed groups. The header consists of a single field that fills the width of the table completely. A "-" indicates that a group can be collapsed.
- **Collapsed:** Format for the headers of collapsed groups. A "+" indicates that a group can be expanded.

ID	NAME	START
⊟ **A**		
1	SW Development	02.09.08
3	Requirements	02.09.08
7	Final Check	16.09.08
12	QA Requirement Check	23.09.08
⊞ ***Group C***		
⊞ ***Group B***		
⊟ **E**		
15	Final Party	30.09.08

*Picture above: The format **Subtitle** is displayed in the rows **GroupC** and **GroupB** indicating a collapsed group level; the format **Subtitle Collapsed** is displayed in the rows **A** and **E**, indicating an expanded group level*

3.29 Time Scale

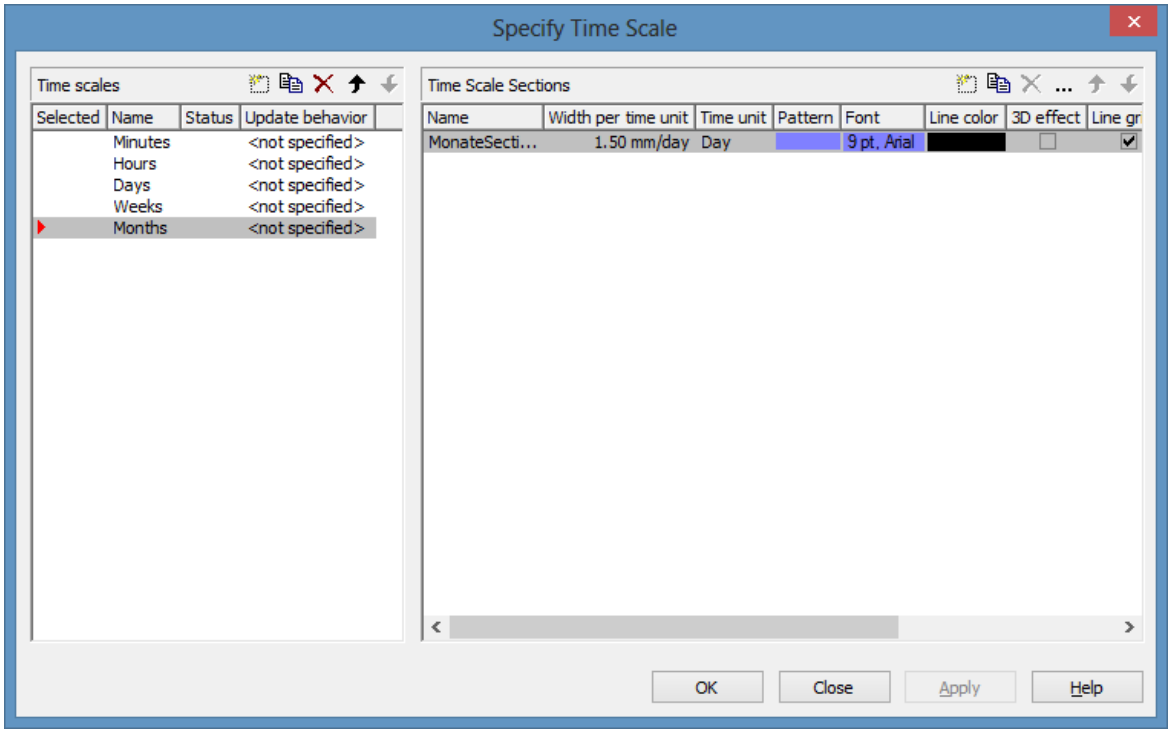
In a diagram, a time scale is displayed at the top of a Gantt graph. Another time scale can be displayed below the Gantt graph (see the dialog boxes **Edit Time Scale Section**, **Ribbons** and **Position**). An appropriate timescale for the time period displayed can be selected.

You can partition the time scale into sections by specifying their number, ranges and scales. Project periods that you want to show in particular detail can be displayed in magnification. Perhaps you wish to present your project plans for the immediate future in more detail than your plans for the distant future or for the past, allowing to concentrate on the project phases that currently are of high interest. You can shift the focus as your project proceeds. Or you can start with a general project overview and continue your planning in increasing detail.

June 2011					July 2011				
Cw 23	Cw 24	Cw 25	Cw 26		Cw 27	Cw 28	Cw 29	Cw 30	Cw 31

There is a whole range of options for designing the time scale, the sections and the grids. For each individual object, you can specify the scales, notations, font attributes, text alignment, colors, line thicknesses, line types, and so on. To keep your planning well-structured, for each section you can define grids, e.g. a day or week grid.

In the **Specify Time Scale** dialog you can select (**Selected**) a time scale for your diagram from a set of preset timescales. The time scales offered differ by their width of time units and by their ribbons.



It is possible to modify the selection at run time.

> Specifying start and end dates of the time scale

The default start and end dates of the time scale you can set on the **General** property page (**Project Start** and **Project End**). At run time, the start date can be adapted to the current date by the property **TimeScaleStart** or both, start and end date by the method **OptimizeTimeScaleStartEnd**. The date format is "DD.MM.YYYY;hh:mm:ss".

Note: The end date is not included. If you set **TimeScaleEnd** = "31.12.2011" for example, the last day displayed will be the 30.12.2011.

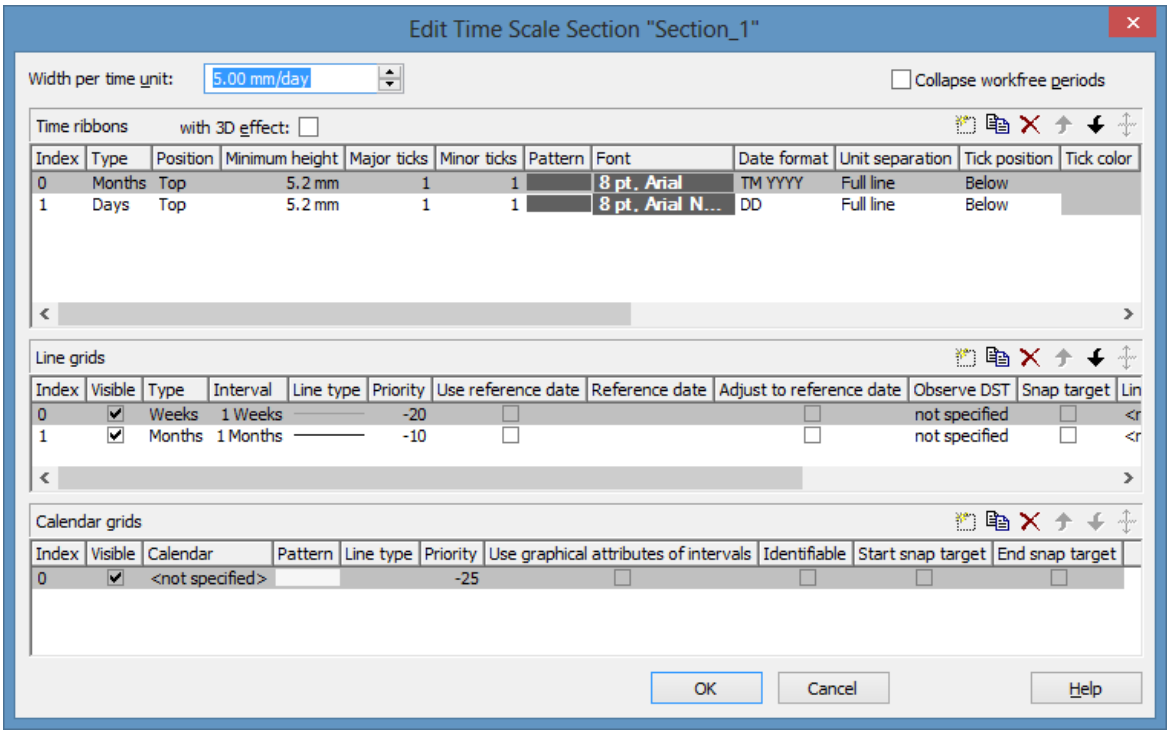
> Sections

You can split the time scale into sections to highlight certain planning periods and specify different ribbons for each section. In the **Specify Time Scale** dialog you can set the **Unit** and the **Width per unit** individually for each section. Also, you can define a separate color, a font, a 3D effect, a line grid and a calendar grid; and you can specify whether workfree periods should be suppressed.

When setting a line grid to a section, vertical grid lines which can be configured are displayed.

When using a calendar grid, workfree periods are marked by colored vertical areas.

From the **Specify Time Scale** dialog you can get to the **Edit Time Scale Section** dialog box where you can edit the ribbons and grids of a section.



> Unit width

A unit is the basic value by which a time scale can be divided. Possible units are: second, minute, hour and day. You can select the unit width in the **Specify Time Scale**.

You can specify the width of a unit by 100ths of millimetres or you can modify the existing value by that size. The minimum width that can be assigned to a time unit is 0.01 mm.

> Ribbons

Ribbons serve to annotate the timescale. A section may consist of more than one ribbon (e.g. one with a monthly and another one with a daily scale). To a ribbon you can assign a **Position**, i. e. whether it is to be displayed at the top or at the bottom of the Gantt graph and whether it is to be displayed at all. Beside, you can specify a type, a minimum height, major and minor ticks, a color, a pattern, a font, a date format, a unit separation, a tick position, a tick color, an alignment, a serial annotation, a reference date, a calendar.

To compose the date you can use the following tokens:

D: first letter of the day of the week (not adjustable)

TD: Day of the Week (adjustable by using the event

OnSupplyTextEntry)

DD: two-digit figure for the day of the month: 01-31

DDD: first three letters of the day of the week (not adjustable)

M: first letter of the name of the month (not adjustable)

TM: name of the month (adjustable by using the event **OnSupplyTextEntry**)

MM: two-digit figure for the month: 01-12

MMM: first three letters of the name of the month (not adjustable)

YY: two-digit figure for the year

YYYY: four-digit figure for the year

WW: two-digit figure for the number of the calendar week: 01-53

TW: text for "calendar week" (adjustable by using the event **OnSupplyTextEntry**)

Q: one-digit figure for the quarter: 1-4

TQ: name of quarter (adjustable by using the event **OnSupplyTextEntry**)

hh: two-digit figure for the hour in 24 hours format: 00-23

HH: two-digit figure for the hour in 12 hours format: 01-12

Th: Text of "o' clock" (adjustable by using the event **OnSupplyTextEntry**)

TH: "am" or "pm" (adjustable by using the event **OnSupplyTextEntry**)

mm two-digit figure for the minute: 00-59

ss: two-digit figure for the second: 00-59

TS: short date format, as defined in the regional settings of the windows control panel

TL: long date format, as defined in the regional settings of the windows control panel

TT: time format, as defined in the regional settings of the windows control panel

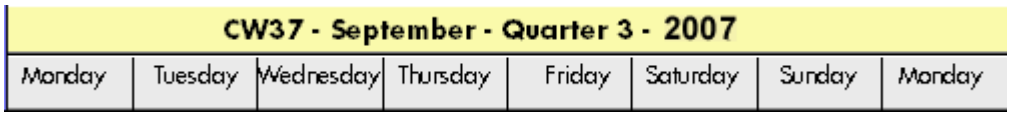
xC/XC: You can set a maximum ten-place, simple upward counting from a reference date onward, for example "15:05:07:16:00", which equals 15 months, 5 days, 7 hours, 16 minutes, 0 seconds. The notation is: **xC44:C33:C22:C11:C00**. In written language: Show at least 2 digits for the counters 4...0 and a preceding "-" symbol if the value

is negative. The separators are variable and can be replaced by other separators symbols. "x" means: Display a preceding "-" symbol if the value is negative, but no "+" symbol if it is positive. "X" means: Display a preceding "-" symbol if the value is negative and a "+" symbol for positive values. In the dialog **Edit Time Scale Section...** the check boxes **Use reference date** and **Adjust major ticks to reference date** need to be ticked, also, the parameter **Serial annotation** has to be set to **No**. In the application the reference date is set at run time by the call **VcRibbon.set ReferenceDate**, overriding any settings in the dialog.

Note: Characters which are not to be interpreted as part of the date should be preceded by a backslash '\'. '\\' for instance results in '\'. The special characters: **':, /, -'** and **blank** don't need '\' as prefix.

> Example for the ribbon annotation

1. ribbon: **TW WW-TM-TQ-YYYY**, 2. ribbon: **TD**



You can replace the predefined texts by our own texts by setting the property **TextEntrySupplyingEventEnabled** to "True". Then you can react to the following values of the ControlIndex:

- vcTXERibDay0 to vcTXERibDay6 (2212 to 2218)
- vcTXERibCW (2223)
- vcTXERibMon0 to vcTXERibMon11 (2200 to 2211)
- vcTXERibQuar0 to vcTXERibQuar2 (2219 to 2222)

Example Code

```
Private Sub VcGantt1_OnSupplyTextEntry(ByVal controlIndex As _
                  VcGanttLib.TextEntryIndexEnum, _
                  TextEntry As String, returnStatus As Variant)
    Select Case controlIndex
        Case vcTXERibCW
            TextEntry = "Semaine"
        Case vcTXERibDay0
            TextEntry = "Lundi"
        ....
        Case vcTXERibMon8
            TextEntry = "Septembre"
        Case vcTXERibQuar2
            TextEntry = "3. Trimestre"
        End Select
```

```
End Sub
```

Semaine 37 - Septembre - 3. Trimestre - 2007							
Lundi	Mardi	Mercredi	Jeudi	Vendredi	Samedi	Dimanche	Lundi

3.30 Tooltips During Runtime

Tooltips allow to display information on the objects that the mouse is hovering over. The events **OnToolTipText** and **OnToolTipTextAsVariant** let you edit tooltips (None, Node) that occur during runtime, in order to, for example, translate them into a different language or to suppress them.

The event **OnToolTipTextAsVariant** is required if you use a Script language that does not allow to return strings, e.g. VBScript. To activate the event, activate the check box **OnTooltipText events** on the **General** property page, or set the property **ShowToolTip** to **True**.

Example Code

```
VcGantt1.ShowToolTip = True
```

Then capture the appropriate one of the events **OnToolTipText** or **OnTool-TipTextAsVariantt** and set the text that you want to appear or whether no tooltip should be displayed in that place.

3.31 Unicode

To display Unicode characters on the property pages at design time, an appropriate font has to be set by following the menu of the operating system through **Start / Settings / Control Panel / Display / Appearance** to the **Window** field.

Besides, only those characters can be displayed that belong to the language set by the menu items **Start / Settings / Control Panel / Regional and Language options** .

All objects in a VARCHART component which contain texts can display Unicode characters if an appropriate font was set in the corresponding property **Font**.

A Unicode font can be assigned to context menus, tooltips and run time dialogs by the property **DialogFont** of the **DummyObject** object.

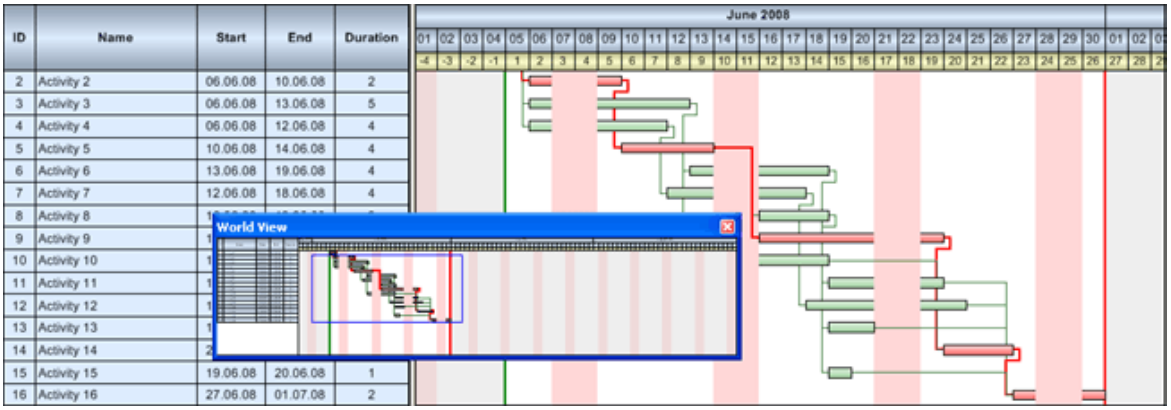
You will find an overview of all available fonts, which contain at least part of all unicode characters in "Wazu Japa's Gallery of Unicode Fonts" (http://www.wazu.jp/index.html). Detailed information on the Unicode standard is also offered on the homepage of the Unicode Consortium (http://www.unicode.org) and on Microsoft's GlobalDev Homepage (http://www.microsoft.com / globaldev / getwr / steps / wrg_unicode.mspx). In Windows 2000 and XP you can find out about the characters contained in the built-in fonts under **Start / Programs / Accessories / System Tools / Character Map**.

When importing CSV files, the method **VcGantt.Load** automatically recognizes whether there is a Unicode or an ANSI file.

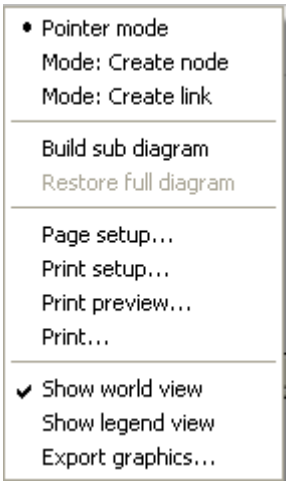
Note: The development environments of Visual Studio 6 are not able to use Unicode characters in source code files. Internally however, the strings of VB6 are displayed in Unicode. If you use Visual C++ combined with MFC you have to set the Defines_UNICODE and UNICODE to use strings in Unicode. The version Visual Studio .NET 2002 and later versions allow to edit source code files in Unicode coding. When saving a file, you need to select the coding type "Unicode".

3.32 World View

The world view is an additional window that displays the diagram completely including the histogram, if present. A frame each indicate the diagram section and the histogram section displayed in their actual size by the main window. If you move one of these frames, the corresponding section in the main window will move proportionally as soon as you release the mouse button. In a similar way, you can enlarge or reduce the display in the main window by zooming the frame in the world view. Vice versa, the position or the size of the frame in the world view will change if you scroll or zoom the section in the main window.



At runtime, you can switch on and off the world view in the default context menu by the menu item **Show world view**.



On the **Additional Views** property page you can specify the properties of the World View. For details please see **The Additional Views Poperty Page** in the chapter **Property Pages and Dialog Boxes**.

The properties of the World View can also be specified by the API property **VcGantt.VcWorldView**.

3.33 Writing PDF Files

Writing PDF files is only possible if an appropriate PDF printing driver is available. The drivers that are free of charge and those that are commercially available differ in their functionality and in the quality of the created PDF files.

Due to the lack of a consistent standard for the controlling of drivers, each printing driver has to be configured individually. The target path for the output file of many PDF printing drivers for instance is preset and can only be modified by altering the Windows registry, by editing INI files or by using driver-specific function APIs or COM objects.

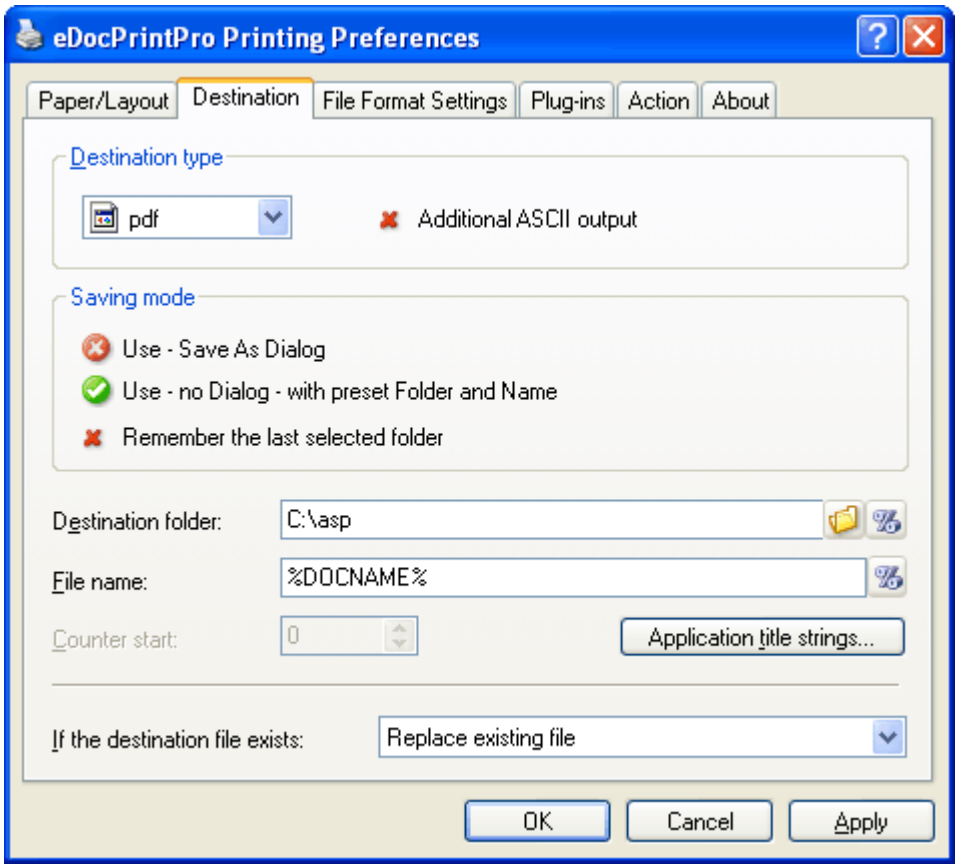
To be suitable a PDF printing driver has to fulfill the below requirements concerning controlling and print quality:

- Depending on the design of the application, it may be necessary that the driver offers the option of switching off all runtime dialogs and message boxes, in particular dialogs for setting file names and paths.
- If file names and paths shall not be set until runtime and if this is only possible by modifying entries of the Windows registry, the permissions of the user account have to be set accordingly.
- For the correct output of texts, Unicode support is needed.
- Fill patterns have to be displayed in sufficient quality. Please note that apart from bitmaps, transparencies cannot be displayed. In bitmaps however, unwanted artifacts may occur.
- The driver has to support vertical text output, otherwise the vertical annotation of date lines in VARCHART XGantt cannot be used.

The aforementioned requirements are fulfilled for instance by the printing driver included in the **Adobe Acrobat Suite** from version 6 onward [www.adobe.com] and the free driver **eDocPrintPro** [www.pdfprinter.at].

Below, please find an outline of the required steps to control the printing driver, using the example of **eDocPrintPro**:

- The dialog **Printing Preferences** can be accessed by the driver's settings in the control panel or by the driver's entry in Start/Programs or by the usual print dialog of an application. If necessary you can in that dialog select that the PDF file should be created without a dialog popping up and that the name of the target file is to be derived from the name of the document for instance. The required settings in **eDocPrintPro** then look as follows:



- In the program, the VcPrinter object of VARCHART XGantt should contain the below settings:

Example Code

```
VcGantt1.Printer.PrinterName = "eDocPrintPro"
VcGantt1.Printer.DocumentName = "abc.pdf"
VcGantt1.PrintEx
```

Very few printing drivers require a different program code:

Example Code

```
VcGantt1.Printer.PrinterName = "Win2PDF"
VcGantt1.PrintToFile "abc.pdf"
```

For further information concerning configuration and usage of **eDocPrintPro** please contact the producer.

3.34 Dragging tools

4 Property Pages and Dialog Boxes

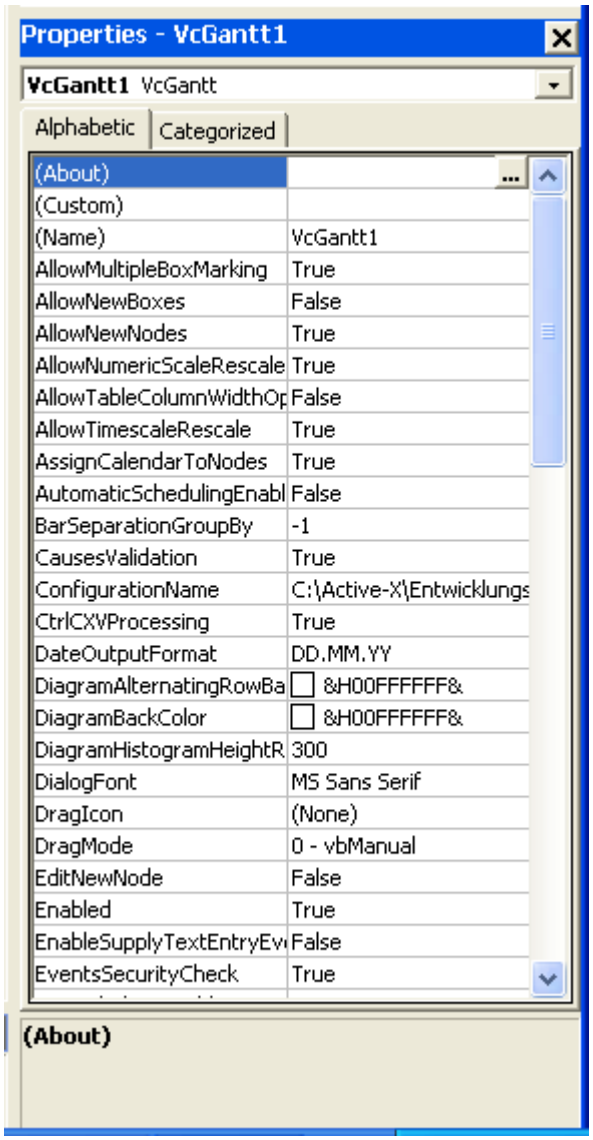
4.1 General Information

Property pages allow to configure VARCHART XGantt already at design time. There are two ways to get to the property pages:

- Press the right mouse button while the mouse pointer is on the control and select **Properties** from the context menu.

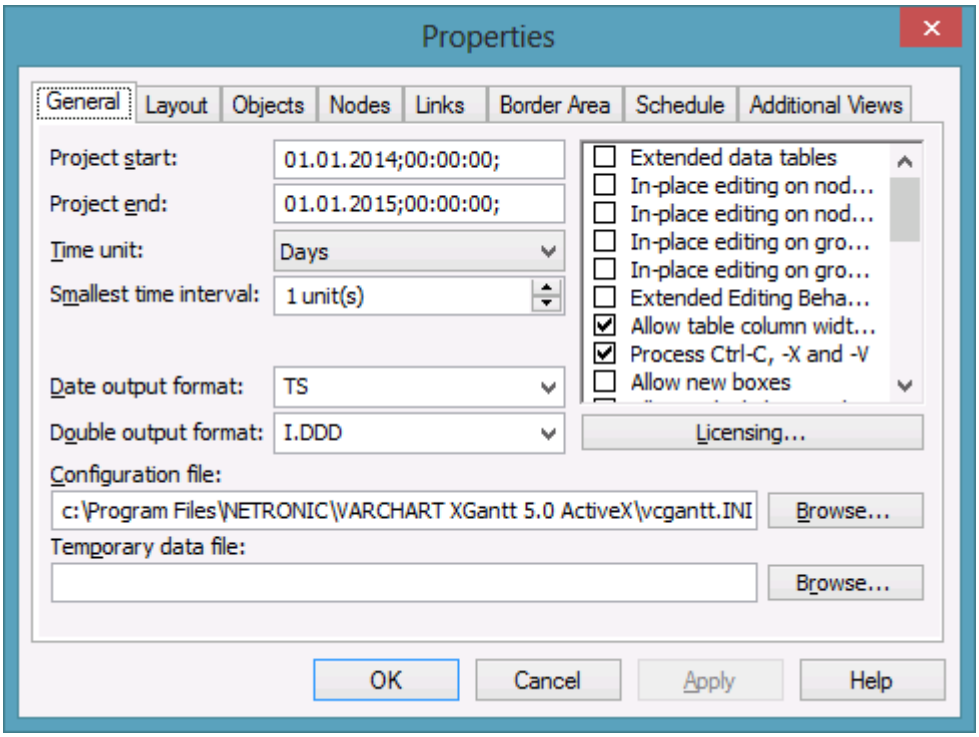
or

- In the **Properties** box of the control (to be invoked by the F4 key) click on the right icon in the icon bar  .



More information about the functions of property pages and dialog boxes you can obtain by either clicking on the **Help** button or by pressing the **F1** key of your keyboard. This will open the corresponding online help file.

4.2 The "General" Property Page



On this property page you can enter general settings of VARCHART XGantt.

Project start

In this field you can set the default start to the time scale. At runtime, you can adapt the start value to the current data by the property **TimeScaleStart** and both, the start and end value by the method **OptimizeTimeScaleStartEnd**. The date format is "DD.MM.YYYY;hh:mm:ss;".

Project end

In this field you can set the default end to the time scale. At runtime, you can adapt the end value to the current data by the property **TimeScaleStart** and both, the start and end value by the method **OptimizeTimeScaleStart-End**.The date format is "DD.MM.YYYY;hh:mm:ss;". The date format is "DD.MM.YYYY;hh:mm:ss;".

Note: The specified day of the end date is not included. If you set **TimeScaleEnd** = "31.12.2011" for example, the last day displayed will be the 30.12.2011.

Time unit

The value entered here will be used to calculate the duration (see Chapter "Important Terms: Layer") and for interactive modification and moving of nodes in the diagram.

Example: If you select the time unit "Days", nodes can only be moved by as many days as specified in the field **Smallest time interval**.

This feature can also be set by the property **VcGantt.timeUnit**.

Smallest time interval

Specify how many time units are equivalent to one step.

Example: If you set the **Time Unit** to "Minutes" and the **Smallest time interval** to "30", the nodes can be moved by steps of half hours.

This feature can also be set by the property **VcGantt.TimeUnitsPerStep** .

Date output format

Select a format from the select box for your date output, or define your own format. The format will also apply to the dialogs of VARCHART ActiveX at runtime.

This feature can also be set by the property **VcGantt.DateOutputFormat**.

To compose the date you can use the following tokens:

D: first letter of the day of the week (not adjustable)

TD: Day of the Week (adjustable by using the event **OnSupplyTextEntry**)

DD: two-digit figure for the day of the month: 01-31

DDD: first three letters of the day of the week (not adjustable)

M: first letter of the name of the month (not adjustable)

TM: name of the month (adjustable by using the event **OnSupplyTextEntry**)

MM: two-digit figure for the month: 01-12

MMM: first three letters of the name of the month (not adjustable)

YY: two-digit figure for the year

YYYY: four-digit figure for the year

WW: two-digit figure for the number of the calendar week: 01-53

TW:	text for "calendar week" (adjustable by using the event **OnSupplyTextEntry**)
Q:	one-digit figure for the quarter: 1-4
TQ:	name of quarter (adjustable by using the event **OnSupplyTextEntry**)
hh	two-digit figure for the hour in 24 hours format: 00-23
HH:	two-digit figure for the hour in 12 hours format: 01-12
Th:	Text of "o' clock" (adjustable by using the event **OnSupplyTextEntry**)
TH:	"am" or "pm" (adjustable by using the event **OnSupplyTextEntry**)
mm	two-digit figure for the minute: 00-59
ss:	two-digit figure for the second: 00-59
TS:	short date format, as defined in the regional settings of the windows control panel
TL:	long date format, as defined in the regional settings of the windows control panel
TT:	time format, as defined in the regional settings of the windows control panel

Note: Characters which are not to be interpreted as part of the date should be preceded by a backslash '\'. '\\' for instance results in "\'. The special characters: **':, /, -**' and **blank** don't need '\' as prefix.

Double output format

From the select box, please select a format for the data type **Double**.You can choose between **I** (whole number), **I.DDD, I.DDDDDD or I,DDD, I,DDDDDD** (3 or 6 decimal places) and **$ I,III.DD** or **I.III,DD €** currency with 2 decimal places.

This feature can also be set by the **VcGantt.DoubleOutputFormat** property.

Configuration file

In this field the configuration file is displayed. By the **Browse** button you can browse for a different file.

The configuration file serves to export the current configuration or to import a stored configuration.

Export: The entry in the field **configuration file** specifies the name of a file to which the current settings are stored. Enter a file name that does not yet exist and click on the **Apply** button. The INI file will be generated and linked to the VARCHART ActiveX instance. All modifications in the property pages will be stored to the file.

Import: The entry in the field **configuration file** specifies the name of a file from which the current settings are loaded. If you enter an existing file name and click on the **Apply** button, the file will be loaded and linked to the VARCHART ActiveX instance. The current modifications in the property pages will be lost.

The settings of the configuration file are loaded only for once. The VARCHART ActiveX control will not read them for a second time from the same file. Instead, the settings will be loaded from internal storings, that are the same as the ones in the configuration file.

So modifying the data of the configuration file from outside will not be effective. If you do want the VARCHART ActiveX control to accept a modified configuration file, you need to rename it and import the renamed file.

Temporary data file

While you are in design mode of your diagram, you can use this option to set a file containing activity data to control the settings for the table section and layers.

By clicking on the **Browse** button you can get to the Windows **Open** dialog box where the file type is preset to "Activity Data (*.bar)".

The setting is only valid during the design time. For runtime, the data file needs to be opened by the method **Open**.

Extended data tables

When activating this box you can create up to 90 data tables, which allows to handle more complex data structures than do the two existing default tables **Main data** and **Relations**. This feature can also be set by the property **VcGantt.ExtendedDataTables**.

In-place editing on nodes in table

Tick this option if in-place editing of node data (if grouping is switched on: of leaf node data) is to be allowed in the table.This feature can also be set by the property **VcGantt.InPlaceEditingOnNodesInTableEnabled**.

If to certain data fields in-place editing shall not be permitted, please don't select the option **editable** in the data definition.

In-place editing on nodes in diagram

Tick this option if in-place editing of node layers (if grouping is switched on: of leaf node layers) is to be allowed in the diagram.This feature can also be set by the property **VcGantt.InPlaceEditingOnNodesInDiagramEnabled**.

If to certain data fields in-place editing shall not be permitted, please don't select the option **editable** in the data definition.

In-place editing on groups in table

Tick this option if in-place editing of group node data is to be allowed in the table. For this, the group data have to use their own data tables. This feature can also be set by the property **VcGantt.InPlaceEditingOnGroupsInTable-Enabled**.

If to certain data fields in-place editing shall not be permitted, please don't select the option **editable** in the data definition.

In-place editing on groups in diagram

Tick this option if in-place editing of group node layers is to be allowed in the diagram.This feature can also be set by the property **VcGantt.InPlace-EditingOnGroups InDiagramEnabled**.

If to certain data fields in-place editing shall not be permitted, please don't select the option **editable** in the data definition.

Extended Editing Behavior

Tick this box to use extended features to edit the table contents and to navigate. This feature can also bet set by the property **VcGantt.Extended-EditingBehavior**.

- **Mark nodes and enter new contents:**

When clicking on a node not only the table row and the corresponding node in the diagram will be marked but you can also directly enter data into a table field.

Please take notice of the following:

When clicking in the **diagram**, the **first** field of the corresponding table line will be marked and will be ready for editing, no matter which field was marked before. By clicking on a different node, the marking will move accordingly and the first field of the corresponding line will be marked.

When clicking in the table area, the field hit will be edited.

For both procedures the following is valid:

You can move the marking by the arrow keys up/down or by the ENTER key and thus mark the previous/next line. If in the table area a field different to the first one should have been marked before, a corresponding selection will appear in the newly marked line. In an already marked table line, the arrow keys right/left will move the marking to the next/previous field, respectively.

Note: By pressing the ESC key, all markings will be undone.

- **Modify field contents**

To modify the contents of a table field you can either click on the field once more or press the F2 key.

There are some data types however which do not require this any more. You can modify date and time fields by clicking on the arrow button. For more information about the usage of the date dialog box please see chapter 4.40 The "Specify Date Lines" Dialog.

The value of numeric data fields may be increased or decreased by clicking on the corresponding arrow buttons.

Note: By pressing the ESC key you can leave the edited fields without saving the modifications.

- **Insert new table lines**

By the INS key you can insert a new row above the current one. If now row was marked, the new line is inserted at the end of the table.

Allow table column width optimization

If you tick this box at run time, double-clicking on a limiting lines of columns will cause the width of the left-hand column to automatically adapt to the length of the texts which it contains. This feature can also be set by the property **VcGantt.AllowTableColumnWidthOptimization**

Process Ctrl-X, -C and -V

If you activate this check box, the key combinations Ctrl+C, Ctrl+X and Ctrl+V will be translated automatically into the clipboard commands **CopyNodesToClipboard**, **CutNodesToClipboard** and **PasteNodesFromClipboard**, respectively. You can revoke this feature by leaving the check box blank, in order to avoid interfering with menu commands in Visual Basic. This feature can also be set by the **VcGantt.CtrlCXVProcessing** property.

Allow new boxes

If you tick this box, the user can create new boxes at run time. To do so, select the **Mode: Create box** or set **InteractionMode** to **VcCreateBox**.

This feature can also be set by the **VcGantt.AllowNewBoxes** property

Allow multiple box marking

By ticking this box, a user can select several boxes at the same time by clicking on them without having to keep the CTRL-key pressed. This option by default is initially disabled.

This feature can also be set by the property **VcGantt.AllowMultipleBoxMarking** .

Show context menu for boxes

Tick this option to enable the context menu for boxes at runtime.

This feature can also be set by the property **VcGantt.ContextMenuForBoxesEnabled** .

Show timescale dialog

Activate this option if the **Edit TimeScale** dialog box is to appear when the user double-clicks on the time scale.

This feature can also be set by the property **VcGantt.ShowTimeScaleDialog** .

Allow time scale rescale

Please activate this option if you want to enable the user to interactively modify the resolution of the time scale.

This feature can also be set by the property **VcGantt.AllowTimeScale-Rescale** .

Allow numeric scale rescale

Specify whether the user should be allowed to rescale the numerical scale of the histogram.

This feature can also be set by the **VcGantt.AllowNumericScaleRescale** property.

Allow zooming by mouse wheel

Tick this option if zooming by mouse wheel is to be allowed. For zooming the user needs to keep the Ctrl key depressed while turning the mouse wheel.

This feature can also be set by the property **VcGantt.ZoomingPerMouse-WheelAllowed** .

OnToolTipText events

Tick this option if the event **OnToolTipText** is to be activated. The event lets you set the text strings to be displayed as tooltip texts with objects.

This feature can also be set by the property **VcGantt.ShowToolTip**.

Scroll events

By ticking this box, you can enable or disable the scroll events. This feature can also be set by the **VcGantt.ScrollEventsEnabled** property.

Note: The scroll events are **disabled** by default.

OnSupplyTextEntry events

By ticking this box you can trigger the **OnSupplyTextEntry** event. The event lets you modify the texts of context menus, dialog boxes and error messages that appear during run time, for example to translate them into different languages.

This feature can also be set by the property VcGantt.**EnableSupplyText-EntryEvent**.

Events Security Check

Tick this option if a confirmation request for the events **OnNodeModify** and **OnNodeModifyEx** is to be carried out. In the above events the **set** calls to the corresponding object types will be suppressed .

This feature can also be set by the property **VcGantt.EventsSecurityCheck**.

Allow reduction of row heights

This option controls the way of calculating the row height in the diagram. If the check box is not ticked, the vertical offsets of the layers are applied by using an imaginary zero line in the vertical center of a node line. To keep the zero line always in the center of the row, it thus may happen that either the top or the bottom row margin will seem rather broad . The layers with a vertical offset of 0, however, stay always vertically centered .

If the check box is ticked, the imaginary zero line is still used but its position is no longer necessarily in the center of the row but so that the row height is as low as possible. Thus it may happen that layers with a vertical offset of 0 are not on the same level as the vertical centered text of the corresponding table row.

This feature can also be set by the property **VcGantt.RowHeightReduction-Enabled**.

Allow font anti-aliasing

This option allows to set anti-aliasing to font characters. If the legibility of certain fonts - in particular non- latin ones - changes for the worse, the option should be switched off.

The anti-aliasing with GDI+ has yet another effect: regardless of the selected zoom factor, texts keep their relative dimension so that the number of characters that fits in a table field will always be the same. If the option is switched off the settings of the operating system are applied instead (the settings can be found in the **Control Panel**, dialog box **Display**, Tab **Appearance: Effects**). Thus, if the option **Smooth edges** is switched on in the **Control Panel**, the texts might still be anti-aliased, notwithstanding the settings of the **General** property page. In this case, at some zoom levels more text could be visible than at others, since the native edge smoothing does not guarantee that the same relative dimension is always kept.

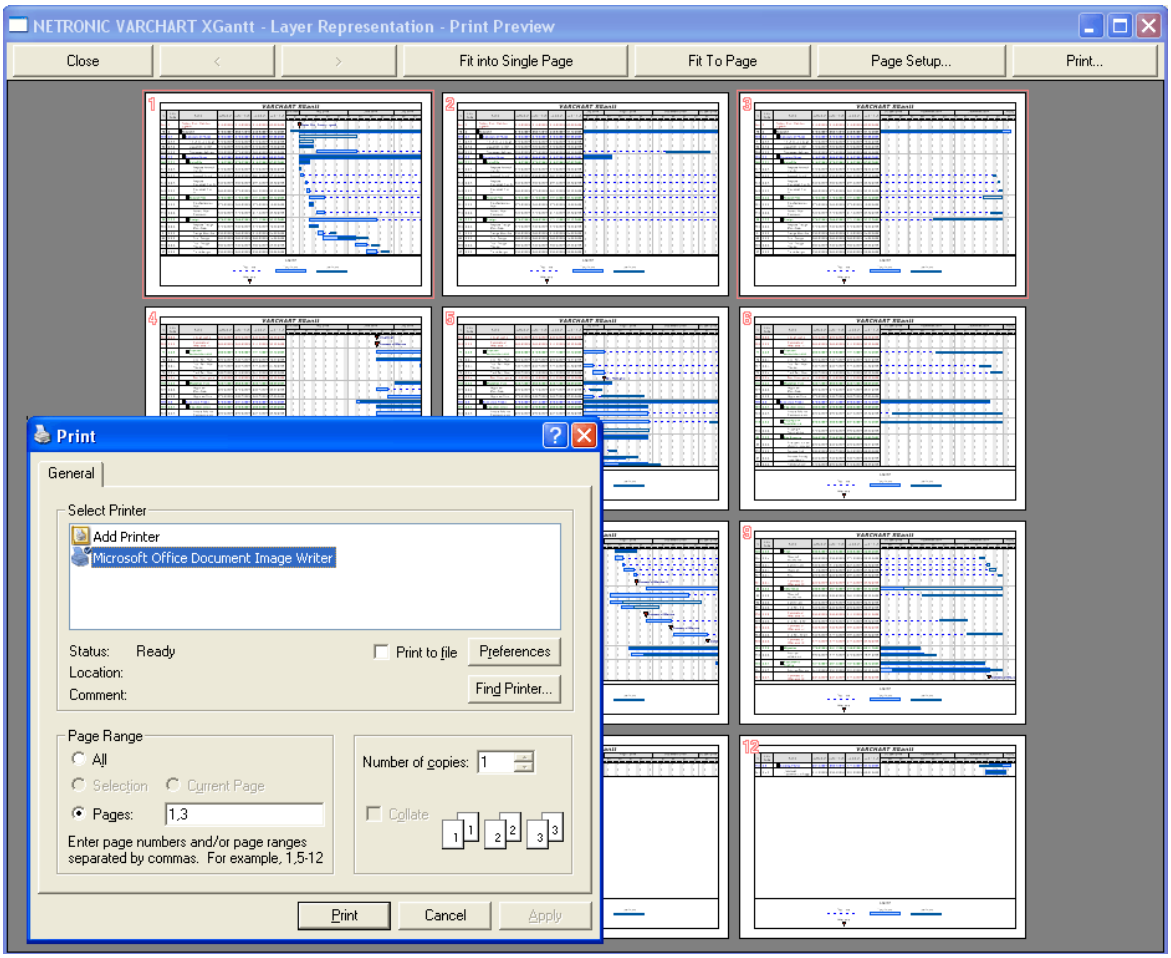
This feature can also be set by the property **VcGantt.FontAntiAliasing-Enabled**.

Use PrintDlgEx dialog

If you tick this check box, the item **Printer setup** will be missing at runtime both in the print preview and in the context menu because the corresponding dialog is now to be found in the (extended) **Print** dialog. If a new project is created, this option is ticked by default whereas in already existing projects it is ticked off for compatibility reasons.

In the print preview you can now select pages by a left click (one page) or by using CTRL + left click (more pages). The selected pages are then preset already as pages to be printed in the **Print** dialog.

If you invoke the **Print** dialog from the print preview, all pages have a page number to make the selection of pages easier.



This feature can **not** be set by an API property.

Enable rounded link slants

If you activate this check box, the slants of links of the routing type **vcLRTOrthogonalDistinguishable** are displayed as quarter circles instead

of straight lines. This feature can also be set at run time by the VcGantt property **RoundedLinkSlantsEnabled** .

Optimization of groups on interactions

If this property is set to true, the nodes of the target group automatically are optimized on interactions such as creating nodes, moving nodes or modifying their start or end date, if they had been in the optimized state of display before. If this property is set to false, on the interactions mentioned the node will be placed at the cursor, if this doesn't cause nodes to overlap. If it does, the node will be placed with other nodes in the next line, if this doesn't cause overlaps. If it does, a new line will be created below the one where the cursor is and the node will be put there.

This feature can also be set by the **VcGantt.GroupOptimizationOn-InteractionsEnabled** property

Consider relation type on node dragging

Tick this box if you want the phantom lines that represent the links to be displayed indicating their type if dragged, and if links are switched on at all. The phantom lines will not start off from the center of the node, but from the left and right side of the node.

This feature can also be set by the **VcGantt.ConsiderLinkRelationTypes-OnNodeDragging** property.

Wait cursor enabled on time-critical operations

Tick this box if you want to set us an internal wait cursor on time-critical operations.

This feature can also be set by the **VcGantt.WaitCursorEnabled** property

Allow panning mode

Tick this box to be able to move certain screen sections at runtime. The contextmenu will then show the additional item **Panning mode**.

Activating the panning mode will apply to **all** view components by default. The **VcGantt.VcViewComponent** property allows to set the panning mode for certain selected components only.

This feature can also be set by the**VcGantt.AllowPanningMode** property.

Allow selection via rubber rect

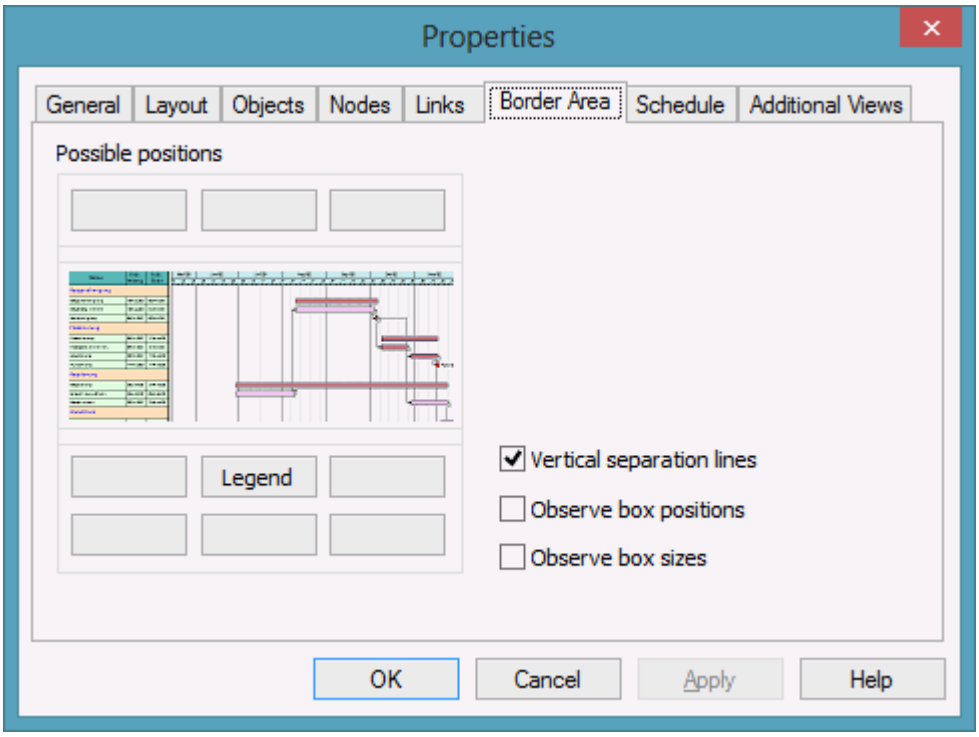
This option allows to enable/disable the selection of nodes by rubber rectangle.

This feature can also be set by the **VcGantt.AllowSelectionViaRubberRect** property.

Licensing

By this button you can get to the **Licensing** dialog box. For more information see chapter **The "Licensing" Dialog**

4.3 The "Border Area" Property Page



Possible positions

There are three areas above and six areas below the diagram which you can utilise for texts, graphics or a legend. These areas are displayed only in the print preview and in the print output. Click on one of the buttons above/below the diagram to reach the **Specification of texts, graphics and legend** dialog box.

Vertical separation lines

Activate this check box, if the areas for texts, graphics or the legend are to be separated by vertical lines.

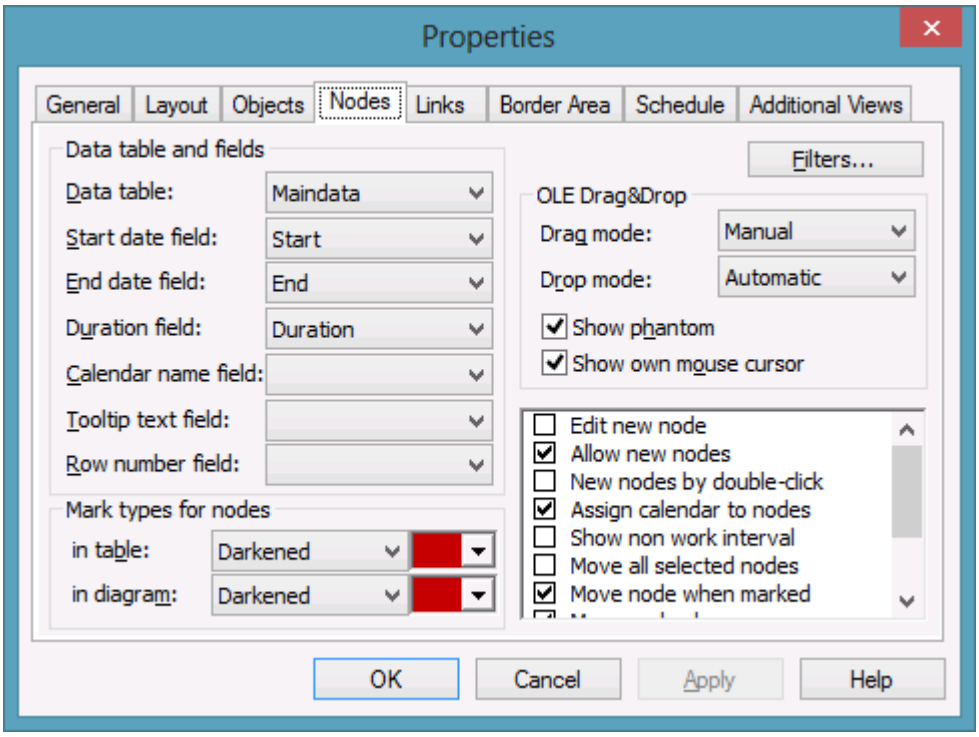
Observe box position

Activate this check box, if the box positions are to be considered as exactly as possible. Otherwise the available space will be divided proportionally between all elements in the row.

Observe box size

Activate this check box, if the box sizes are to be considered as exactly as possible. Possibly the chart will be enlarged and/or the texts in the boxes will be cropped.

4.4 The "Nodes" Property Page



Data table

Select the data table to be used for the visualisation of the nodes.

This feature can also be set by the property **VcGantt.NodesDataTable-Name**.

Start date field

Please select the data field to store the start date of an interactively created node. Date fields only are offered in the select box.

This feature can also be set by the property **VcGantt.NodeStartDateData-FieldIndex**.

End date field

Please select the data field to store the finish of an interactively created node. Date fields only are offered in the select box.

This feature can also be set by the property **VcGantt.NodeEndDateData-FieldIndex**.

Duration field

Please select a data field to store the duration of an interactively created nodes. Only numeric data fields are available.

This feature can also be set by the property **VcGantt.NodeDurationData-FieldIndex**.

Calendar name field

If you wish to use an individual calendar for a node, you can select the data field to store the name of the calendar. For this, the check box **Assign calendar to nodes** needs to be activated. Beside, the calendars have to be created before loading the nodes.

Tooltip text field

The data field specified here is shown as a tooltip if you show a VMF file using the WebViewer and there right-click on a node. No further settings are required.

The VMF (Viewer Metafile) format is a vector format that allows to store a chart independently of pixel resolution. Files of the VMF format can be displayed by the GRANEDA WebViewer on any platform using Java compatible internet browsers.

To show tooltips in your VARCHART ActiveX application, activate the check box **OnToolTipText events** on the **General** property page or set the property **ShowToolTip** to **True** and in the **OnToolTipText** event, specify the data fields to be displayed.

This feature can be also set by the property **VcGantt.NodeToolTipText-Field**.

Row number field

Please select a data field which stores the row number of the node. The modifications only become effective after having carried out an update by using the method **VcGantt.UpdateRowNumberFields**

This feature can also be set by the property **VcGantt.NodeRowNumber-DataFieldIndex**.

Mark type for nodes in table

Use the field on the left to specify whether marking of nodes should be allowed in the table and if so, select the type of marking to be used:

- No Mark
- Surround inside
- Invert
- Darken (by 25%)
- Brighten (by 25%)
- Pickmarks inside

The field to the right lets you select a color for the marking type.

Mark type for nodes in diagram

Use the field on the left to specify whether marking of nodes should be allowed in the table and if so, select the type of marking to be used:

- No Mark
- Surround
- Surround inside
- Invert
- Darken (by 25%)
- Brighten (by 25%)
- Pickmarks
- Pickmarks inside

The field to the right lets you select a color for the marking type.

Filters

This button lets you open the **Administer Filters** dialog box. The filter which serves for preselecting the nodes can only be set at runtime by the property **ActiveNodeFilter** of the object **VcGantt**.

Drag mode

By this property you can set or retrieve, whether dragging a node beyond the limits of the VARCHART XGantt control should be allowed.

- If you select **Manual** you need to invoke the method **OLEDrag** to trigger dragging the node.
- If you select **Automatic**, dragging a node beyond the control limits will be started automatically.

On the start of dragging, the source component will fill the DataObject with the data that it contains and will set the **effects** parameter before initiating the OLEStartDrag event, as well as other source-level OLE Drag & Drop events. This gives you control over the drag/drop operation and allows you to intercede by adding other data formats.

VARCHART XGantt by default uses the clipboard format CF_TEXT (corresponding to the vbCFText format in Visual Basic), that can be retrieved easily.

During dragging, the user can decide whether to shift or to copy the object by using the Ctrl key.

OLE drag & drop operations in VARCHART XGantt are compatible to the ones in Visual Basic. Methods, properties and events have identical names and meanings as the default objects of Visual Basic.

This feature can also be set by the property **VcGantt.OLEDragMode**.

Drop mode

By this property you can set or retrieve, whether a node from a different VARCHART XGantt control can be dropped to the present control.

- Dropping will not be allowed if you select **None**.
- If you select **Manual**, you will receive the event **OLEDragDrop** that enables you to process the data received by the object dropped, e.g. to generate a node or to read a file. If the source and the target component are identical, you will receive either the event **OnNodeModifyEx** or **OnNodeCreate** as with OLE Drag&Drop switched off.
- If you select **Automatic**, the dropping will automatically be processed by the control, generating a node in the place of the dropping, if possible.

This feature can also be set by the property **VcGantt.OLEDropMode**.

Show phantom

This property lets you disable the display of an OLE drag phantom. Disabling the phantom is useful if generating a new object is omitted but merely the attributes of the object in the target control are modified.

This feature can also be set by the property **VcGantt.OLEDragWith-Phantom**.

Show own mouse cursor

This property lets you enable or disable the mouse cursor in the target control during an OLE drag operation. OLE Drag & Drop allows to set the cursor in the source control by the event **OLEGiveFeedback**. If you set it, two competing cursors will exist in the target control, that may appear to flicker. You can avoid the flickering by disabling the target cursor by this check box.

Beside, if the cursor is enabled and the property **OLEDropManual** is set, objects cannot be dropped outside the joining ports of a node. If you disable the cursor, you can drop objects outside the joining ports.

This feature can also be set by the property **VcGantt.OLEDragWithOwn-MouseCursor**.

Edit new node

If you tick this box, the **Edit Data** dialog box will open automatically when the user creates a new node interactively. After having created a node, the **Edit Data** dialog box can be invoked (even if this option is disabled) either by double-clicking this node or by the corresponding item of the context menu.

This feature can also be set by the property **VcGantt.EditNewNode**.

Allow new nodes

This option needs to be activated if you want to enable the user to create new nodes interactively in an open project. New nodes can be created in the **Mode: Create Node** (click in the diagram area, context menu) or, if the **New nodes by double-click** option was ticked, by double-clicking in the appropriate position in the diagram.

This feature can also be set by the property **VcGantt.AllowNewNodes**.

New nodes by double-click

This box lets you specify whether new nodes can be created by a double-click. A new node created this way will be inserted at the current cursor position. Its size (duration) will be a single time unit.

This feature can also be set by the property **VcGantt.NewNodesViaDoubleClick**.

Assign calendar to nodes

Tick this box to assign calendars to the nodes. Assigning calendars to nodes will entail the following: The start and end dates of the activities will not be positioned on workfree days. Workfree periods will be taken into account when calculating the duration of the activities. Currently, the default is a five-day calendar ("WeekCalendar").This feature can also be set by the property **VcGantt.Assign CalendarToNodes**.

If no individual calendar has been assigned per node, the calendar which was defined as active in the CalendarCollection is used.

Show non work interval

Please activate this check box to have workfree intervals highlighted. They will be displayed as was specified in the **Edit layer** dialog.

This feature can also be set by the property **VcGantt.ShowNonWorkInterval**.

Move all selected nodes

Tick this check box to enable all marked nodes to be moved. If you leave it deactivated, only single layers or nodes (depending on whether the **Move node when marked** check box was ticked) can be moved by the mouse, even if several nodes have been marked.

This feature can also be set by the property **VcGantt.MoveAllSelectedNodes**.

Move node when marked

Please tick this check box to move all layers of a marked node in one go. A node can be marked by a mouse click on one of its layers.

If this check box is not ticked, the layers of a marked node can only be moved individually. For moving all layers of the node, please keep the SHIFT key pressed while dragging the node. (For this, the **Move layers as node when shift key pressed** check box needs to be ticked).

This feature can also be set by the property **VcGantt.MoveNodeWhenMarked**.

Move node always

If you tick this check box, all layers of a node can be moved in one go without having to be marked before.

This feature can also be set by the property **VcGantt.MoveNodeAlways**.

Move layers as node when shift key pressed

If this box is ticked, all layers of a node can be moved in one go if the Shift key is being pressed while dragging. This feature can also be set at run time by the VcGantt property **MoveLayersAsNodeWithShiftKey**.

Use snap targets in interactions

If this box is ticked, the snap target functionality can be used while dragging a node/layer, meaning to specify whether a node/layer "snaps" at the defined snap targets of the respective objects . This feature can also be set at run time by the **VcGantt** property **UseSnapTargetsInInteractions**.

Show snap lines

Ticking this box enables snap lines to be shown while nodes are being resized or dragged with the snap target mode switched on. These lines help to better recognize the defined snap targets.

This feature can also be set at run time by the **VcGantt** property **ShowSnap-Lines**.

Show snap targets

Ticking this box enables snap markings to be shown while nodes are being resized or dragged with the snap target mode switched on. These lines help to better recognize the defined snap targets.

This feature can also be set at run time by the **VcGantt** property **ShowSnap-Markings**.

Allow vertical node movement via diagram

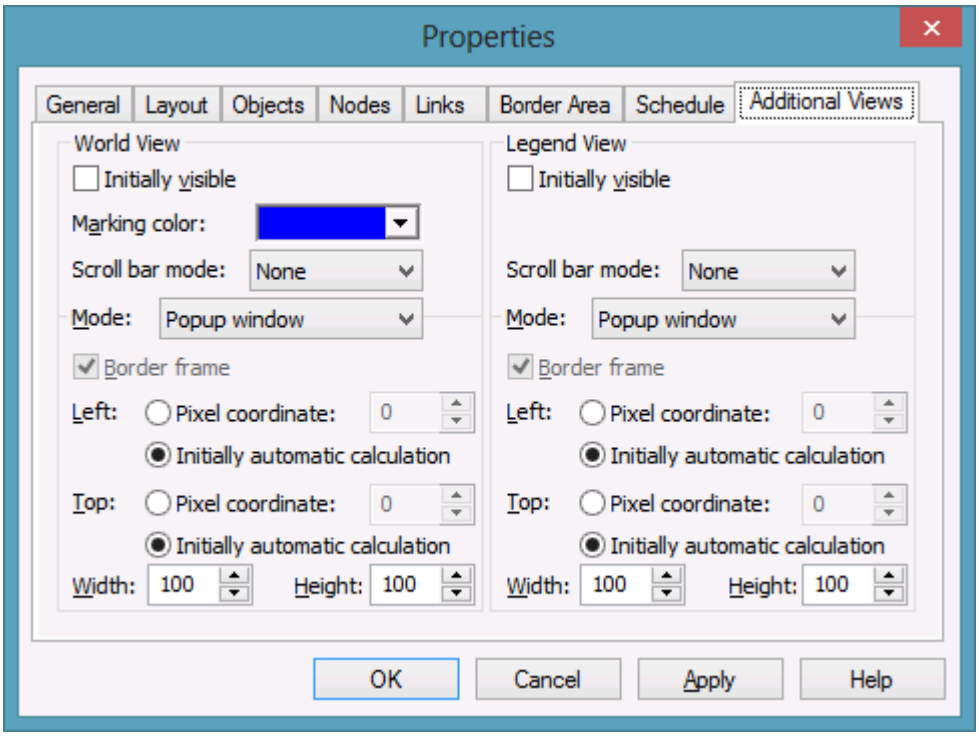
Tick this box if you want the user to be able to change the order of the activities or their group affiliation by dragging nodes from one row to another in the diagram area. If a node consists of more than one layer, the Shift key needs to pressed while dragging vertically. This feature can also be set at run time by the VcGantt property **AllowVerticalNodeMovement**

Allow vertical node movement via table

Tick this box if you want the user to be able to change the order of the activities or their group affiliation by dragging nodes from one row to another in the table area. If a node consists of more than one layer, the Shift key needs to pressed while dragging vertically.

This feature can also be set at run time by the VcGantt property **AllowVerticalNodeMovementViaTable**.

4.5 The "Additional Views" Property Page



On this property page you can set the properties of the "world view" and the legend view.

Both views are additional small windows.

The world view displays the diagram completely. Two frames in it indicate the sections actually displayed in the main window. One of them shows the section in the Gantt Graph, the other one shows the histogram section.

The legend view lets you display a legend.

At run time, you can switch on or off both views in the default context menu by clicking **Show world view** or **Show legend view** respectively. You can alternatively use the **Close** button of the title bar to switch off either view.

The description of the possible settings which you find below, is valid for both views, if not stated otherwise.

Initially visible

Activate this check box if the view is to be visible when the program is started.

This property can also be set by the API calls **VcWorldView.Visible** and **VcLegendView.Visible**

Marking color (only World View)

Select the line color of the frame that indicates the displayed section in the World View.

This property can also be set by the API calls **VcWorldView.MarkingColor** and **VcLegendView.MarkingColor**.

Scroll bar mode

You can select a mode of displaying scrollbars. By using scrollbars, empty areas are avoided and there is more space for displaying the chart or the legend.

- **None:** The world view always displays the complete chart or legend. Thus empty areas may occur if the world view's proportions do not correspond to those of the chart/the legend.
- **Horizontal:** A horizontal scrollbar is displayed if required.
- **Vertical:** A vertical scrollbar is displayed if required.
- **Automatic:** A horizontal or a vertical scrollbar is displayed if required.

This property can also be set by the API calls **VcWorldView.ScrollBar-Mode** and **VcLegendView.ScrollBarMode**.

Mode

Select the view mode. The below options are available:

- **Left fixed:** The view is displayed on the left side of the VARCHART ActiveX control window. Only the width can be set, whereas the position and the height are fixed.
- **Right fixed:** The view is displayed on the right side of the VARCHART ActiveX control window. Only the width can be set, whereas the position and the height are fixed.
- **Top fixed:** The view is displayed on the top of the VARCHART ActiveX control window. Only the height can be set, whereas the position and the width are fixed.
- **Bottom fixed:** The view is displayed on the bottom of the VARCHART ActiveX control window. Only the height can be set, whereas the position and the width are fixed.
- **Position not fixed:** The view is a child window of the current parent window of the VARCHART ActiveX. It can be positioned at any position

and be of any extension. The parent window can be modified by the property **VcWorldView.ParentHWnd**.

- **Popup window:** The view is a popup window and has its own frame. The user can modify its position and extension, he can open it by the default context menu and close it by the **Close** button in the frame.

This property can also be set by the API calls **VcWorldView.Mode** and **VcLegendView.Mode**.

Border frame

*Not active if the mode **Popup window** has been selected.* Activate this check box if the view is to have a frame and select a color in the drop down list..

This options can also be set by the API calls **VcWorldView.Border** and **VcWorldView.Border.Color** or **VcLegendView.Border** and **VcLegend-View.Border.Color**

Left

*Only active if the mode **Position not fixed** or **Popup window** was selected.* Select the left position of the view. There are two options:

1. Specify a **Pixel coordinate** value. Note that this is a system coordinate.
2. Select the **Initially automatic calculation** option.

This property can also be set by the API calls **VcWorldView.Left** and **VcLegendView.Left**

Top

*Only active if the mode **Position not fixed** or **Popup window** has been selected.* Select the top position of the view. There are two possibilities:

1. Specify a **Pixel coordinate** value. Note that this is a system coordinate.
2. Select the **Initially automatic calculation** option.

This property can also be set by the API calls **VcWorldView.Top** and **VcLegendView.Top**

Width

*Not active if the mode **Top fixed/Bottom fixed** was selected.* Select the horizontal extension of the view. Note that the pixel coordinate is a system (device) coordinate.

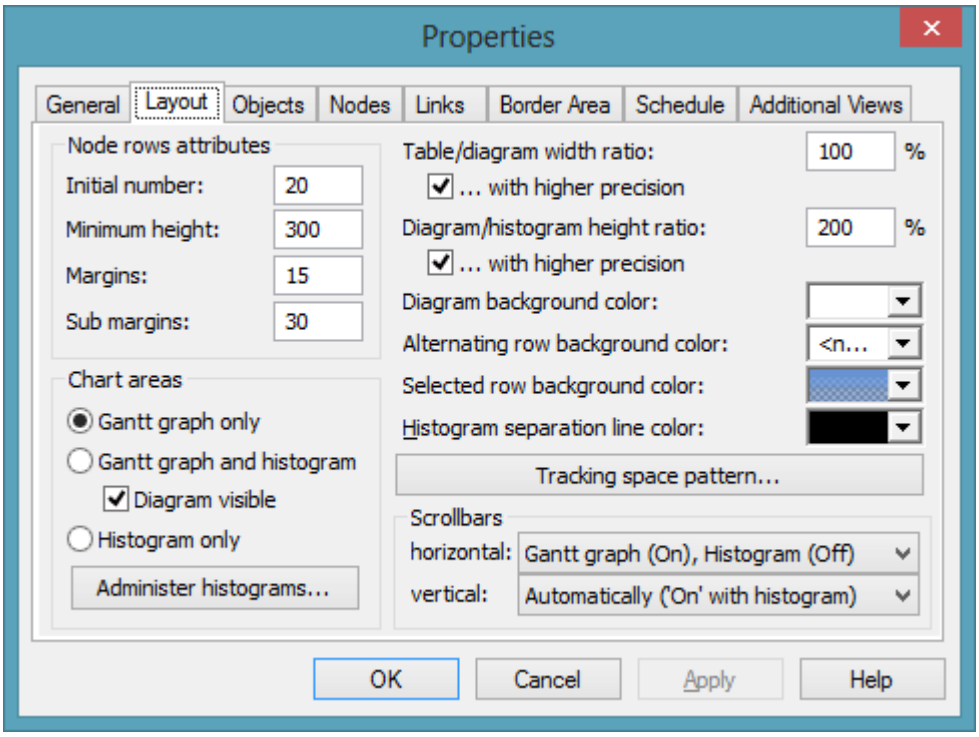
This property can also be set by the API calls **VcWorldView.Width** and **VcLegendView.Width**

Height

*Not active if the mode **Left fixed/Right fixed** was selected.* Select the vertical extension of the view. Note that the pixel coordinate is a system (device) coordinate.

This property can also be set by the API calls **VcWorldView.Height** and **VcLegendView.Height**

4.6 The "Layout" Property Page



On this property page you can establish and modify the layout of the chart.

Initial number

Specify the minimum number of node rows to be displayed in the diagram area on the start of the program.

This feature can also be set by the property **VcGantt.NoOfInitialRows**.

Minimum height

Set the minimum height of the node rows (unit: 1/100 mm). The values permitted range between 2 and 1000.

The minimum row height only becomes effective if there is no activity in the row or if existing activities do not exceed the minimum row height. In all other cases the row height automatically adapts to the space required by the activities.

This feature can also be set by the property **VcGantt.MinimumRowHeight**.

Margins

Set the minimum vertical spacing between the node and the upper/lower node row border (unit: 1/100 mm). This property can also be set at run time by the

property **MinimumRowHeight** of the **VcGantt**object. The values allowed to be set range between 2 and 1000.

The minimum row height only takes effect if there is no activity in the row or if existing activities do not exceed the minimum row height. In all other cases the row height automatically adapts to the space required by the activities.This feature can also be set by the property **VcGantt.RowMargins**.

Sub margins

This property lets you set or retrieve the vertical width between the sub rows. The sub rows only exist if groups are optimized and nodes of this group are arranged in several sub rows to prevent them from overlapping.

This feature can also be set by the property **VcGantt.SubRowMargins**.

Chart areas

Specify the section of the diagram (chart area) to be displayed:

- Gantt graph only
- Gantt graph and histogram
- Histogram only.

Administer histograms

The **Administer Histograms** dialog will appear.

Left table/diagram width ratio

Specify the ratio (in %) of the table width to the width of the complete diagram (table area plus diagram area) at the start of the program. In order to display the complete table at the start, select the value "-1".

This feature can also be set by the property **VcGantt.LeftTableDiagram-WidthRatio**.

…with higher precision

Activate this property to enable the usage of the more accurate methods **Left-TableDiagramWidthRatioEx** and **RightTableDiagramWidthRatioEx** or the event **VcTableWidthChangingEx** that all return a value of the type "Double" to calculate the ratio between table and diagram.

If this property is not activated, the methods **LeftTableDiagramWidthRatio** and **RightTableDiagramWidthRatio** or the event **VcTableWidthChanging** will be used.

This feature can also be set by the **VcGantt.UseHigherTableDiagram-WidthRatioPrecision** property.

Diagram/histogram height ratio

Specify the ratio (in %) of the height of the diagram area (histogram excluded) to the height of the histogram at the start of the program. In order to display the complete histogram at the start, select the value "-1".

This feature can also be set by the property **VcGantt.DiagramHistogram-HeightRatio**.

…with higher precision

Tick this box to enable the usage of the more accurate method **Diagram-HistogramHeightRatioEx** or the event **VcHistogramHeightChangingEx** that return a value of the type "Double" to calculate the width ratio between diagram and histogram.

If this property is set to the default value "False", the method **Diagram-HistogramHeightRatio** or the event **OnHistogramHeight** are used.

This feature can also be set by the **VcGantt.UseHigherDiagramHistogram-HeightRatioPrecision** property.

View components background color

This field lets you select the diagram background color. If you combine this property with the **Alternating row background color**, you can generate a color pattern that alternates linewise.

This feature can also be set by the property **VcGantt.DiagramBackColor** or **VcGantt.ViewComponentsBackColor**.

View components border color

This field lets you select the frame color for all panes at a time.

This feature can also be set by the property **VcGantt.ViewComponents-BorderColor**.

Alternating row background color

This field lets you select a second background color for the diagram, which linewise alternates with the **Diagram background color**.

This feature can also be set by the property **VcGantt.DiagramAlternating-RowBackColor**.

Selected row background color

This field lets you select a background color for the current row.

This feature can also be set by the property **VcGantt.SelectedRowBack-ColorAsARGB**.

Tracking space pattern

This button opens the dialog **Edit Pattern Attributes** where you can specify the layout of the the free area, sometimes showing up briefly at the top or bottom margin during LiveUpdate interactions.

This feature can also be set by the according properties **VcGantt.Tracking-SpaceBackColorAsARGB**, **VcGantt.TrackingSpacePattern** und **VcGantt.TrackingSpacePatternColorAsARGB**.

Scroll bars

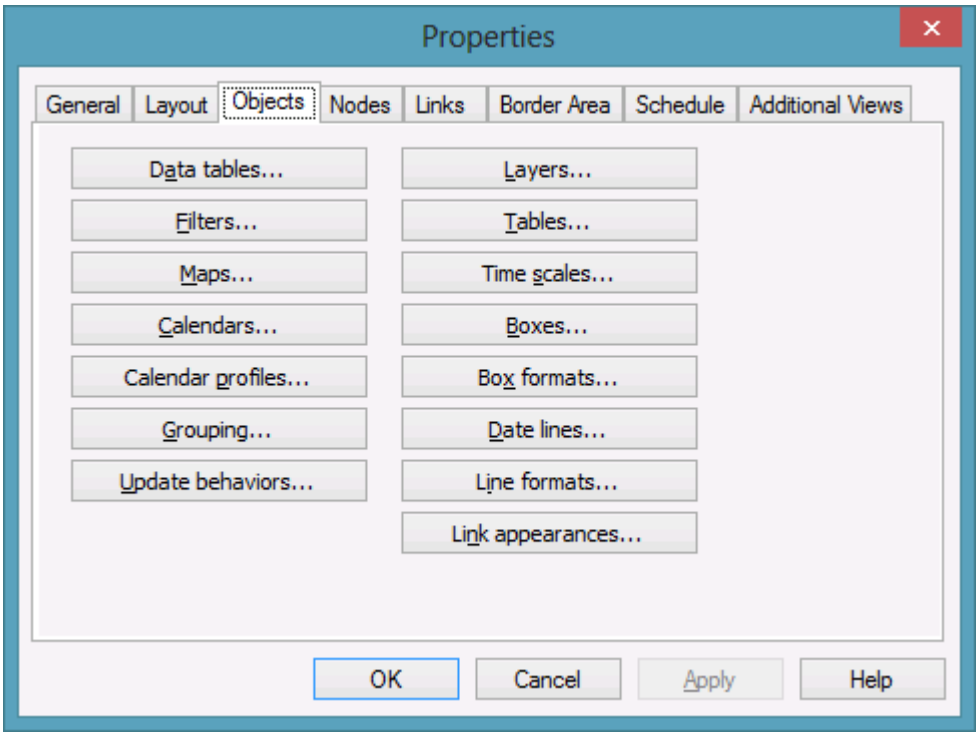
By these options you can set the horizontal and vertical scroll bars. For the horizontal scroll bar, you can choose between the below options:

1. **Gantt graph (on), histogram (off)** - the horizontal scroll bar is located between the Gantt graph and the histogram
2. **Gantt graph (off), histogram (on)** - the horizontal scroll bar is located below the histogram
3. **None** - there is no horizontal scroll bar.

For the vertical scroll bar, you can choose between the below options:

1. **Automatically (but 'on' with histogram)** - a vertical scroll bar will be switched on right of Gantt graph if required; another one is always on right of the histogram.
2. **on** - both, the vertical scroll bar right of the Gantt graph and the one right of the histogram are switched on
3. **off** - both vertical scroll bars are switched off.

4.7 The "Objects" Property Page



Data tables

Opens the dialog **Administrate Data Tables**.

Filters

This button lets you open the **Administrate Filters** dialog box.

Maps

This button will open the dialog **Administrate Maps**.

Calendars

Opens the dialog **Specify Calendars**.

Calendar profiles

Opens the dialog **Administrate Calendar Profiles**.

Grouping

Opens the dialog **Grouping**.

Update behaviors

Opens the dialog **Administrate update behaviors**.

Layers

Opens the **Specify Bar Appearance** dialog box.

Tables

Opens the **Specify Table** dialog box.

Time scales

Opens the **Specify Time Scale** dialog box.

Boxes

Opens the dialog **Administrate Boxes**.

Box formats

Opens the dialog **Administrate Box Formats**.

Date lines

Opens the **Specify Date Lines** dialog box.

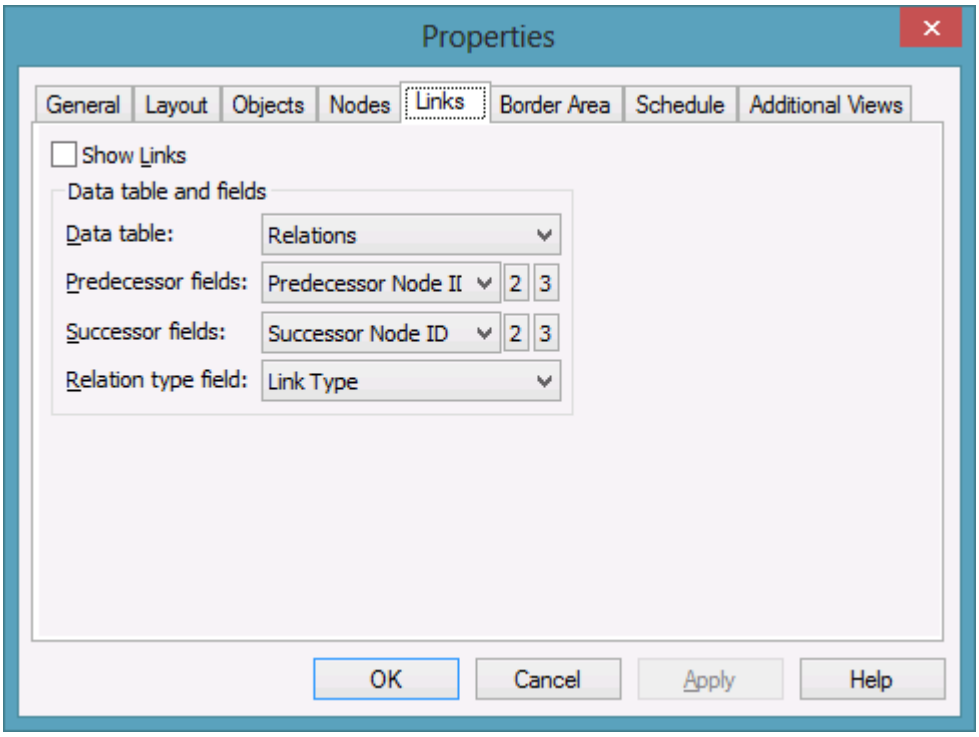
Line formats

This button lets you open the dialog **Administrate Line Formats**.

Link appearances

Opens the dialog **Link appearances**.

4.8 The "Links" Property Page



This property page lets you display links between nodes and establish and modify the appearance of the links.

Show Links

This check box lets you specify whether links and phantom lines representing the links while dragging are to be displayed. This feature can be also set by the API property **VcLinkAppearance.Visible** - but only for the links, not for the lines.

Data table

Select a data table which contains the fields of the links. This feature can also be set by the property **VcGantt.LinksDataTableName**.

Predecessor field

This field lets you select a data field from the **Relations** table that the identification of the predecessor node of the link is stored to. This feature can also be set by the property **VcGantt.LinkPredecessorDataFieldIndex**. This property is an indexed property, which in C# is addressed by the methods set_LinkPredecessorDataFieldIndex (identifierIndex, pvn) and get_Link-PredecessorDataFieldIndex (identifierIndex).

Successor field

This field lets you set the data field or fields from the afore selected data table that the identification of the successor node of the link is/are stored to. This feature can also be set by the property **VcGantt.LinkSuccessorDataFieldIndex**. The property is an indexed property, which in C# is addressed by the methods set_LinkSuccessorDataFieldIndex (identifierIndex, pvn) and get_LinkSuccessorDataFieldIndex (identifierIndex).

Relation type field

Select the data field that contains the relation type. This feature can also be set by the property **VcGantt.LinkTypeDataFieldIndex**.

Pre port symbol

Select a port symbol for each link appearance that accentuates the intersection of the link and the predecessor node.

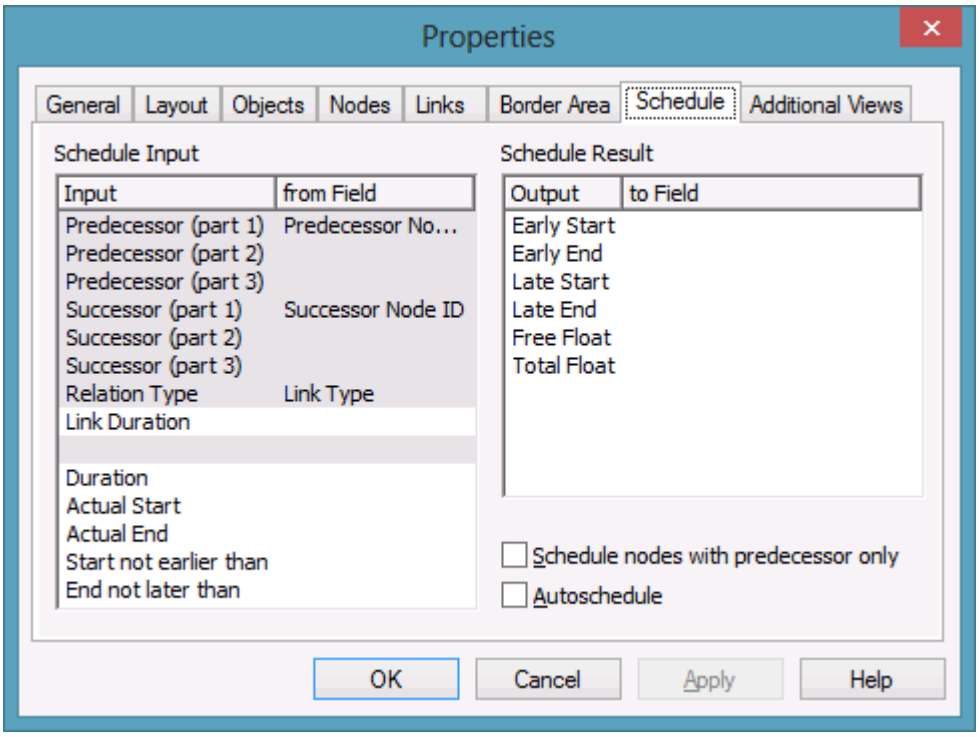
This feature can also be set by the property **Link AppearancePrePortSymbol**.

Suc port symbol

Select a port symbol for each link appearance that accentuates the intersection of the link and the successor node.

This feature can also be set by the property **Link AppearanceSuccPortSymbol**.

4.9 The "Schedule" Property Page



By using this property page you can adapt the date calculation settings of VARCHART XGantt to your interface by specifying the data fields that you wish to use for the input (**Schedule Input**) and the output (**Schedule Result**) of the scheduler. (See "Important Terms: Scheduling".)

Schedule nodes with predecessor only

If you activate this check box, only those nodes will be scheduled that have a predecessor node; otherwise all nodes will be scheduled. A "project start" will be ignored.

If this check box is not ticked, all activities will be taken into account when scheduling.

Autoschedule

If this option is activated, the duration of the depending dates will be recalculated automatically when a link is created or deleted or an when activity is modified.

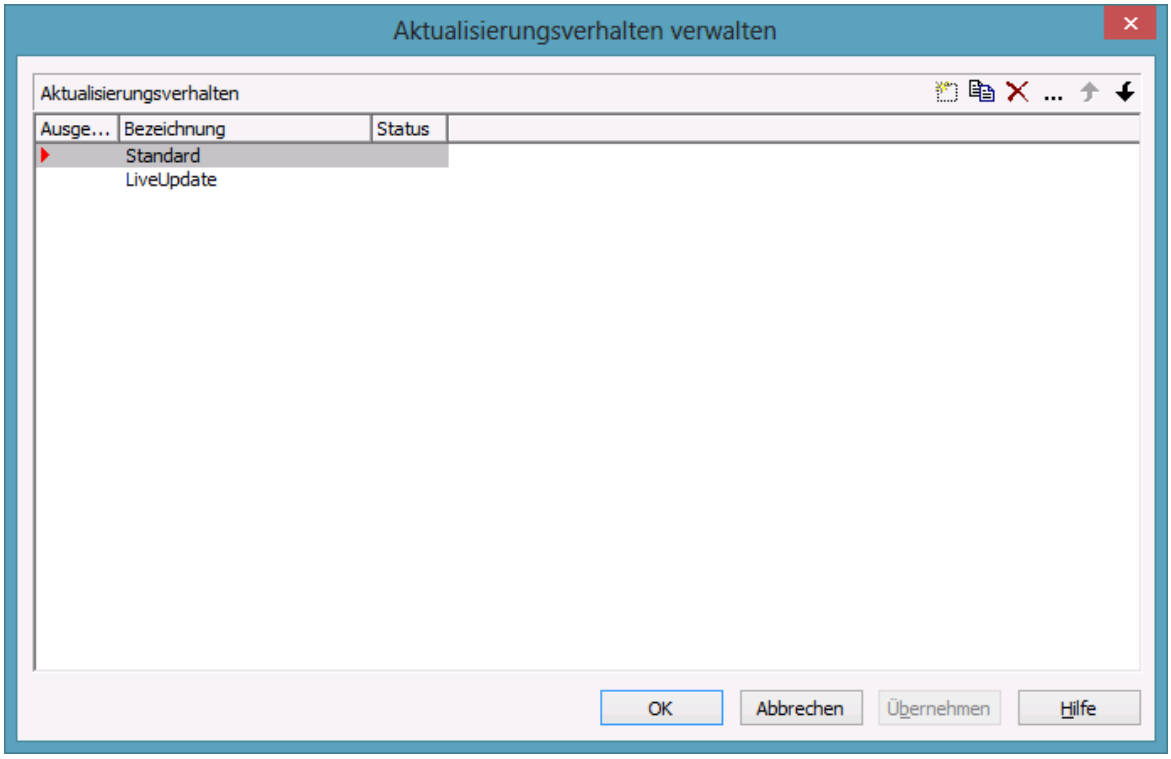
Schedule Input

Please select for each entry of the column the field from which its contents is to be loaded. The scheduler uses data fields of the respective nodes and links tables. The calculations of the scheduler are based on the project start, the duration of the activities and their logic dependence. The fields **Predecessor** and **Successor** cannot be edited by the **Schedule Input** table. They merely display the settings on the **Links** property page.

Schedule Result

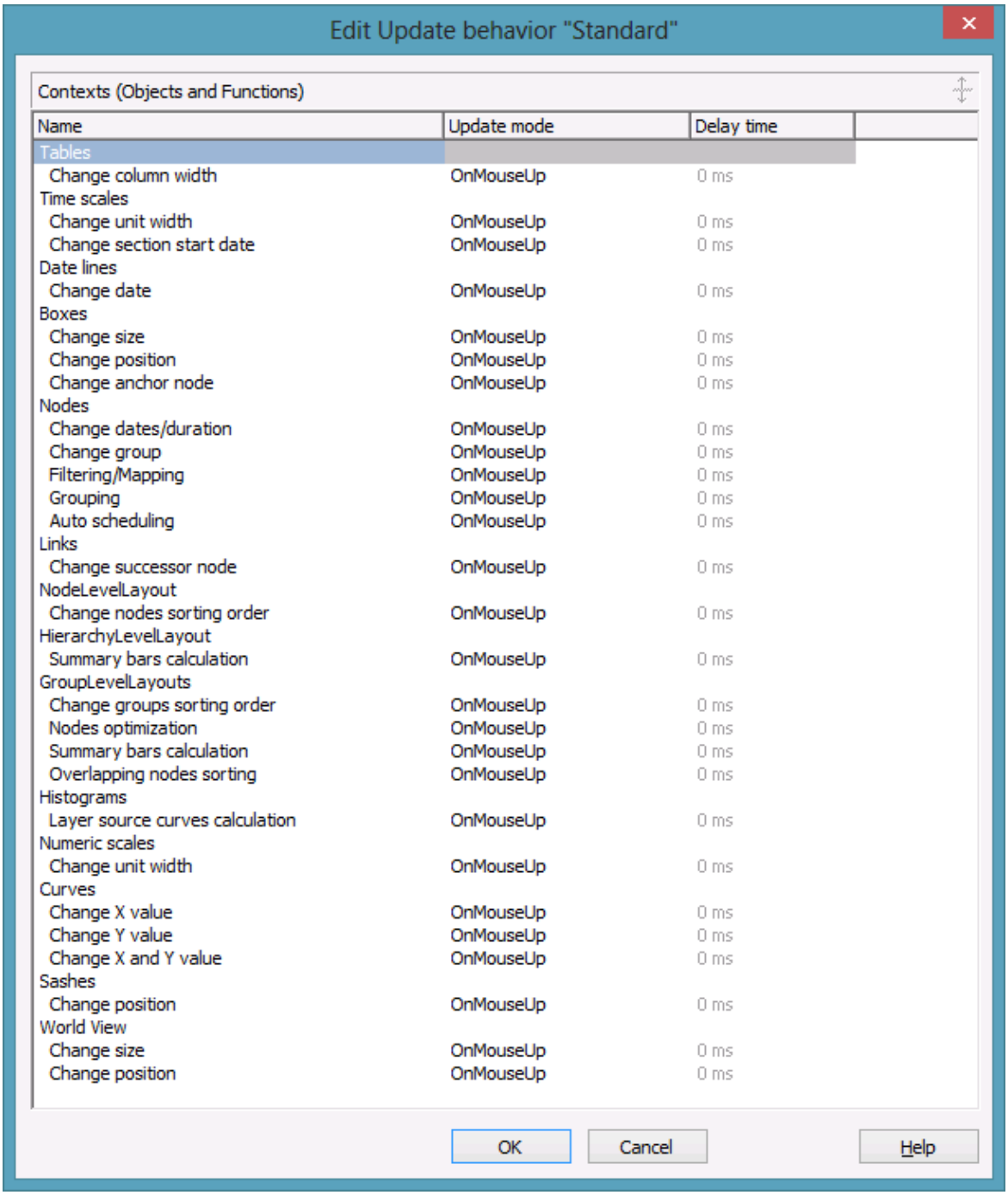
Specify for each result to which field it should be stored. The scheduler only outputs to data fields from the **Maindata** table. The early/late start and end dates plus the total float and free float are calculated from the duration of the activities, the logical dependencies and the project start.

4.10 The "Administrate Update Behaviors" Dialog Box



Click on the corresponding button on the **Objects** property page to open this dialog. Here you can create, copy, delete and shift individual update behaviors.

4.11 The "Edit Update Behaviors" Dialog Box



Name	Update mode	Delay time
Tables		
Change column width	OnMouseUp	0 ms
Time scales		
Change unit width	OnMouseUp	0 ms
Change section start date	OnMouseUp	0 ms
Date lines		
Change date	OnMouseUp	0 ms
Boxes		
Change size	OnMouseUp	0 ms
Change position	OnMouseUp	0 ms
Change anchor node	OnMouseUp	0 ms
Nodes		
Change dates/duration	OnMouseUp	0 ms
Change group	OnMouseUp	0 ms
Filtering/Mapping	OnMouseUp	0 ms
Grouping	OnMouseUp	0 ms
Auto scheduling	OnMouseUp	0 ms
Links		
Change successor node	OnMouseUp	0 ms
NodeLevelLayout		
Change nodes sorting order	OnMouseUp	0 ms
HierarchyLevelLayout		
Summary bars calculation	OnMouseUp	0 ms
GroupLevelLayouts		
Change groups sorting order	OnMouseUp	0 ms
Nodes optimization	OnMouseUp	0 ms
Summary bars calculation	OnMouseUp	0 ms
Overlapping nodes sorting	OnMouseUp	0 ms
Histograms		
Layer source curves calculation	OnMouseUp	0 ms
Numeric scales		
Change unit width	OnMouseUp	0 ms
Curves		
Change X value	OnMouseUp	0 ms
Change Y value	OnMouseUp	0 ms
Change X and Y value	OnMouseUp	0 ms
Sashes		
Change position	OnMouseUp	0 ms
World View		
Change size	OnMouseUp	0 ms
Change position	OnMouseUp	0 ms

This dialog can be reached from the <!Administrate Update Behaviors dialog and allows to switch update modes or to modify

Delay time

Here you can set the delay time after which the modified objects of the live update visually are to appear while the mouse cursor is moving.

Setting this property is only possible if the **Update Mode** was set to **OnPauseWhileMouseMoving**

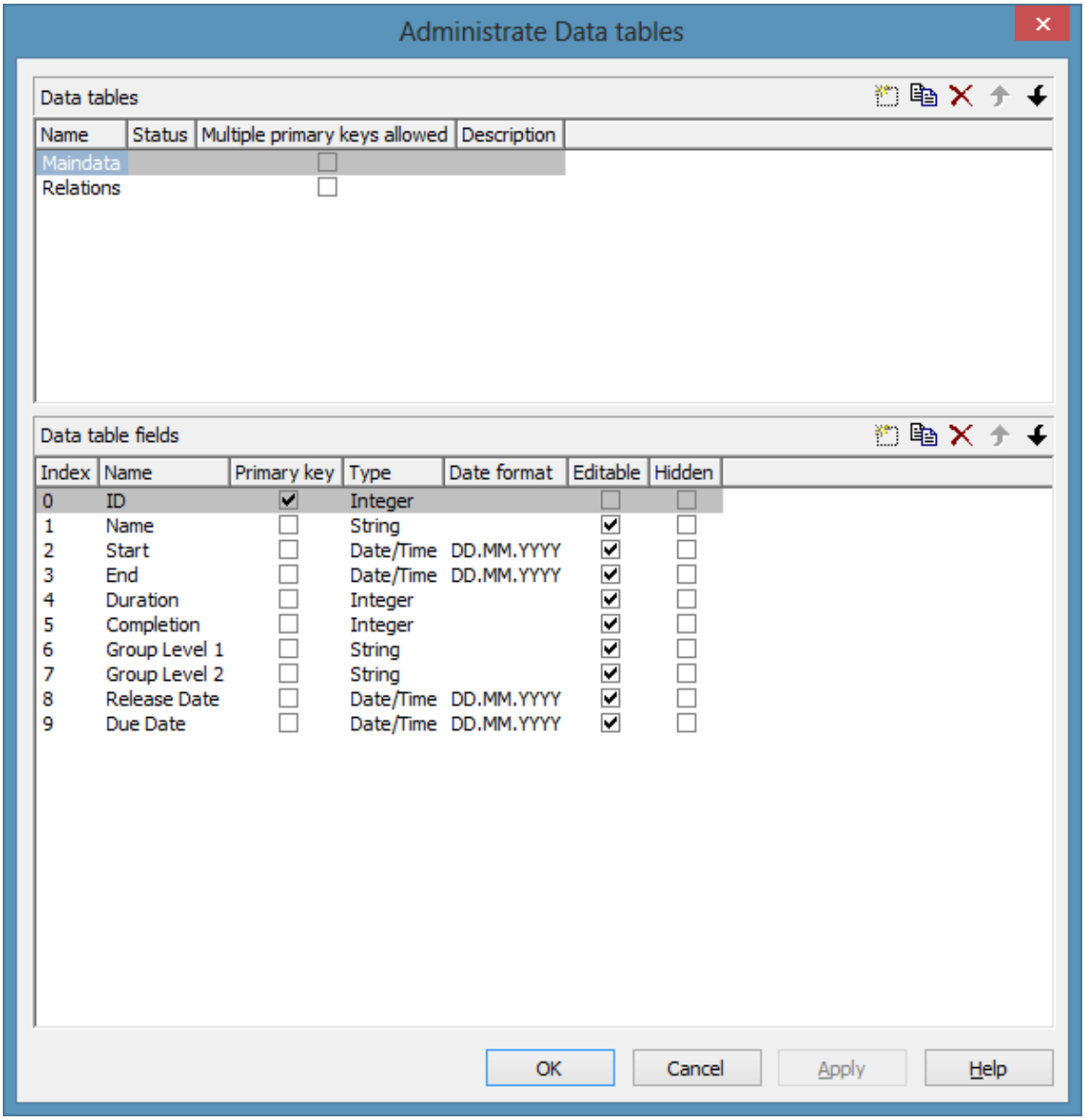
Update mode

Here you can select a cursor action on which the live update is to take place. This is only possible if you are editing an individually created update behavior created.

Name

Lists the names of all tables and relating functions that are affected by the live update. The names can **not** be edited.

4.12 The "Administrate Data Tables" Dialog Box



You can get to this dialog via the property page **Objects**. This dialog lets you create and edit data tables and their data fields.

Data tables

- **Name:** Lists the names of all existing data tables. The names can be edited.

- **Status:** In the **Status** column each data table that has been added ( ) and/or modified ( ) since the dialog box was opened is marked by a symbol.
- **Multiple primary keys allowed:** Here you can define whether the primary key for your table consists of **one** or **more (maximum 3** fields. As soon as you have checked the box **Multiple primary keys allowed** you can select up to three data fields for the primary key in the **Data table fields** section. The box **Multiple primary keys allowed** can only be unchecked if no more than one field is selected as primary key in the **Data table fields** section.
- **Description:** Here you can describe the data table.

Add / copy / delete / edit / promote / demote data table

By these buttons you can create, copy or delete data tables or move them by one position up or down in the list, respectively.

Data Table Fields

Here you can create and edit data table fields of the selected data table.

- **Index:** The index of the data fields cannot be modified, since internally, it serves as a reference. In the API, data fields are referred to by the index.
- **Name:** This column displays the names of the fields of the data table. You can modify the field names after clicking on them.
- **Primary Key:** This check box allows to select a data field from the column to be the primary key of the data record.
- **Type:**This field allows to set the data type of the data field selected. You can choose between:

  String

  Integer

  Date/Time

  Double
- **Date format:** If the type **Date/Time** has been selected, you can specify the date format for the corresponding data field here. Choose a predefined date format or define your own date format (for example DD.MMM.YY hh:mm). You can compose the format of the following strings:

YY or **YYYY** (two-digit or four-digit figure for the year), **MM** or **MMM** (two-digit figure or three-digit character string for the month), **DD** (two-digit figure for the day), **hh** (two-digit figure for the hour), **mm** (two-digit figure for the minute), **ss** (two-digit figure for the second).

Please note that the date format set here needs to be the same as defined for your node dates.

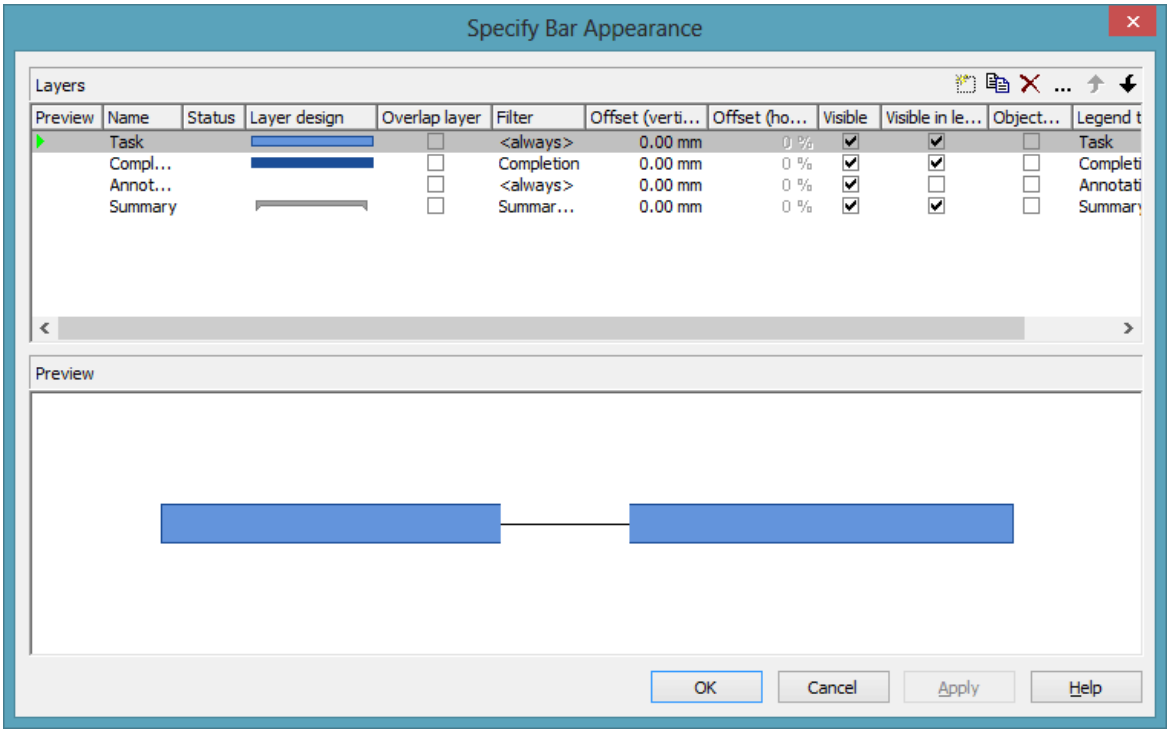
The date format set here only is relevant for entering data, but not for displaying data.

- **Editable:** Please activate this check box for all data table fields that shall be editable in the dialog **Edit Data**.
- **Hidden:** Please activate this check box for all data table fields that shall be hidden in the dialog **Edit Data**.
- **Relationship:**This field allows to define a relationship to another table. The data records of this table will be related to the data records of the other table by the field defined as the primary key. This is why only those tables are offered for selection for which a primary key was defined.

Add / copy / delete / edit / promote / demote data table field

 By these buttons you can create, copy or delete data table fields or move them by one position up or down in the list, respectively.

4.13 The "Specify Bar Appearance" Dialog Box



Activities are represented by bars. The graphical representation of a bar is defined by a bar appearance. The graphical representation is composed by layers that are dynamically assigned to activities by filters.

A layer is the graphical representation of a single date (symbol layers) or a pair of dates (rectangle layers or line layers). The dates are provided by data fields that are specified by the **Edit Layer** dialog box.

Layers are composed by graphical attributes (shape, line color, pattern, etc.) and an annotation. In addition, they can be of different heights and may be displayed with an offset to ensure that all layers assigned to an activity are visible.

If a bar is represented by more than one layer, the layers are drawn consecutively, allowing to overlap. The layer at the top of the **Layer** list in the dialog is drawn first; the layer at the bottom of the **Layer** list is displayed last and may overlap the ones previously drawn. The final bar appearance results from the graphical display of all layers, the filters of which allow the activity to be displayed.

Layer

In the list below, you can define a layer per line.

Preview

Layers marked by a small arrowhead in the **Preview** column are displayed in the preview window in the lower half of the dialog.

A green arrowhead marks the layer on which the cursor is currently positioned. It is displayed in the preview window.

If you click on a layer in the **Preview** column, a red arrowhead will occur and indicate that the layer and its current settings are displayed in the preview window.

Name

This column lists the names of the layers that were defined. The names can be edited.

Status

In this column layers that were added ( ) or modified ( ) after the dialog box was opened, are marked by a symbol.

Layer design

This column displays graphical representations of the layers. To modify the design of a layer, click on the **Edit layer** button in the top right corner of the dialog, or double-click on the desired **Layer design** field to get to the **Edit Layer** dialog box where you can define the graphical attributes and edit the annotation of the layer.

Overlap layer

In the mode **All nodes in one row**, with the option **optimized** switched off, layers may overlap and therefore may hide each other. You can indicate the hidden section by a small overlap layer, that appears below the hidden section and increases or decreases with its size. Only one layer in the list can be an overlap layer. No filter can be applied to it.

Filter

A filter linked to a layer selects the activities that are represented by the layer. To assign a filter to a layer, click on the **Filter** field. Two buttons will appear:

This button lets you open the list of available filters to be selected.

Alternatively, you can click on the **Edit** button to get to the **Administer Filters** dialog box where you can edit, copy, define or delete a filter.

Existing examples of filters: "Standard", "Critical", "Milestone". The chosen filter stipulates the criterium that an activity must fulfil in order to make the layer appear. For example, if you choose the filter **Critical** for a layer named **Early**, the **Early** layer will only be displayed for critical activities.

Offset (vertical)

The vertical offset (displacement of the horizontal center line) is specified in millimeters. Positive values will entail an upward offset, negative values will cause the offset to extend downward.

When clicking in the **Offset (vertical)** field of a layer, two buttons will appear with an arrow pointing upwards and downwards to increase or decrease the vertical offset of the selected layer.

Beside, this button will appear which can take you to the **Configure Mapping** dialog box. Here you can set data-dependent vertical offsets.

After finishing the mapping, the arrow on the button will appear bold.

Offset (horizontal)

(For symbol layers only) When clicking on the **Offset (horizontal)** field of a symbol layer, two buttons will appear with an arrow pointing upwards and downwards. You can use the buttons to increase or decrease the horizontal offset (displacement of the layer date) by steps of 1%, the total range extending from -50 to +50 %.

Visible

Untick this box if you want the layer to be invisible. You can use this feature to hide a layer without deleting it.

Visible in legend

Check this box if you want the layer to be displayed in the legend.

ObjectDraw events

Tick this box to enable the events **OnObjectDraw** and **OnObjectDrawComplete** for nodes in which this layer is used.

Legend text

In this field you can enter a legend text for the layer.

Add layer

This button creates a new layer.

Copy layer

This button copies the marked layer.

Delete layer

This button deletes the selected layer.

Edit layer

This button opens the **Edit Layer** dialog box.

Promote/Demote layer

If a node comprises more than one layer, the layers are stacked on top of each other. The top layer in the list will be drawn first. So, the lower the position of a layer in the list, the more layers it superimposes, i.e. the order of the layers in the list is the order by which they are drawn in the diagram.

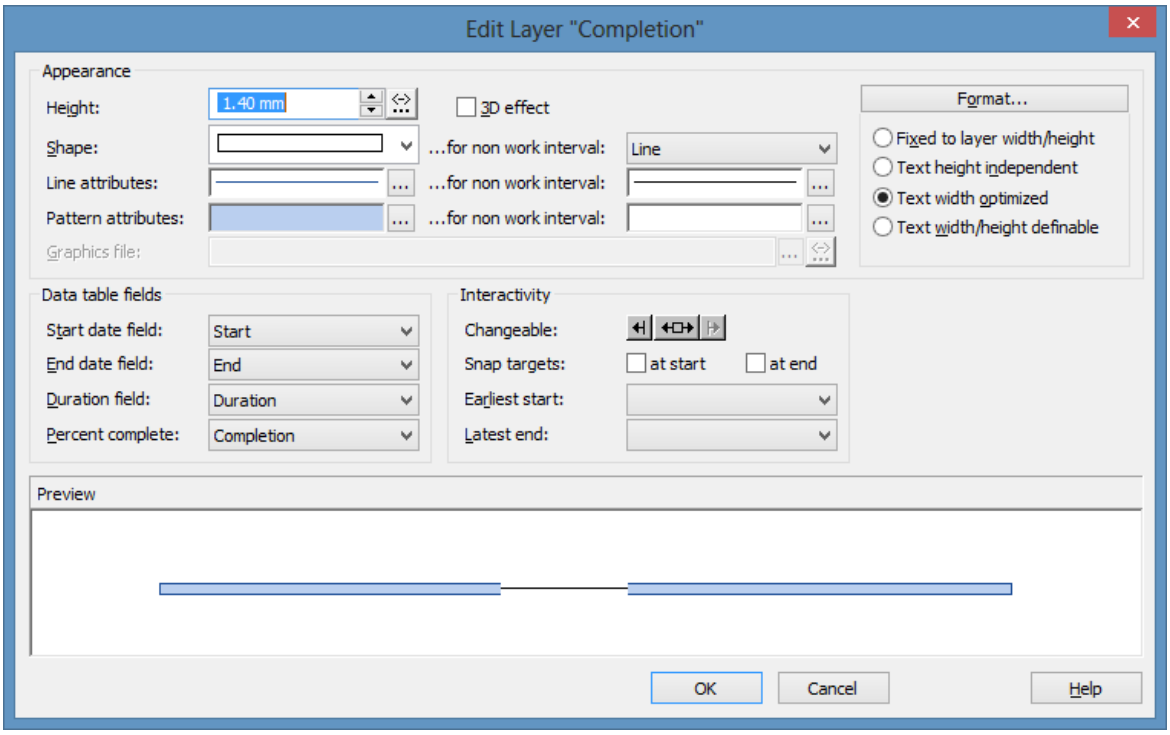
The selected layer will be moved upward by one position in the list, which is equivalent to one position towards the background of the diagram. The layer at the top of the list is superimposed by all other layers.

The selected layer will be moved by one position downward in the list, which ist equivalent to one position towards the foreground of the diagram. The layer at the bottom of the list superimposes all other layers.

Preview window

The preview window displays the layers that are marked in the **preview** column, including their overlaps caused by the drawing priority and by offsets.

4.14 The "Edit Layer" Dialog Box



You can get to this dialog box by clicking on the corresponding button in the **Specify Bar Appearance** dialog. The name of the layer edited is displayed in the head line.

Height

Here you can define the height of the layer in millimetres either by directly entering the desired value into the field or by clicking on either of the two arrows pointing upwards and downwards.

By clicking on this button you reach the **Configure Mapping** dialog box. It allows to assign heights to layers data-dependent.

If a mapping has been configured, the arrow on the button will appear solid.

3D-Effect

Decide whether or not the layer should be given a 3-dimensional perspective.

Shape

Select from the list a shape for the layer. You can choose between:

- **Bitmap layer**: you can browse for a bitmap file in the **Graphics file** field.)
- **Invisible symbol**: only the layer annotation will be visible. The layer also will not be displayed in the legend.
- **Rectangle layer**
- **Wedge-shaped layer**: wedge ascending or descending
- **Line layer**
- Various types of **symbol layers**.

Rectangle, wedge-shaped and line layers are used to show timespans. Wedge-shaped layers are useful for visualising increasing and decreasing activities, e. g. during the project start or end. Symbol layers are used to show specific points in time.

Non work interval shape

Select the form to be displayed for the non work intervals of rectangle layers. Before, the **Layers with NonWork interval** option on the **Nodes** property page has to be ticked.

The drop down list offers the forms <rectancle>, <line>, <empty area> and <no>, <no> having the effect of showing a continous layer. Together with the above mentioned option, one can chose for certain layers to show non work intervals and for others not.

Line attributes

The line type of the layer frame is displayed here. To change it, click on the **Edit** button (···). Then the **Line Attributes** dialog box will open.

Line attributes for non work intervals

Specify the lines for non work interval layer. Click on ··· to open the **Edit line attributes** dialog.

Pattern attributes

Here you can see the currently set layer pattern. Click on ··· to open the **Edit pattern attributes** dialog where you can specify pattern, pattern color or background color.

Pattern attributes for non work interval

Specify pattern and fill color for non work interval layers. Click on ... to open the **Edit pattern attributes** dialog.

Graphics file

(only activated, when for ***Shape*** the option *<Bitmap layer> has been specified*) Select a graphics file to visualize the layer.

Relative path names can also be set. If a relative file name was specified, at run time the first folder to be searched will be the one in the path set by the VARCHART property **FilePath**. If it is not found searching will continue in the current directory of the application and in the installation directory of the VARCHART Control.

... Click on this button to open the **Select Graphics File** dialog box.

By this button you can get to the **Configure Mapping** dialog box where you can configure a mapping for the graphics file. If a mapping was configured, the arrow on the button will be displayed in bold ( ).

The color of the pixel in the left upper corner of the graphics will be replaced by the diagram color, i. e. this color will appear transparent.

Fixed to layer width/height

If you select this option, the height and width of the layer annotation will be fixed to the height and width of the layer.

Text height independent

If you select this option, the height of an annotation outside the layer will be independent of the layer height, whereas its width will depend on the layer width. The height of annotation inside the layer always is restricted by the layer height.

Text width optimized

If you select this option, the width of an annotation outside the layer will be independent of the layer width, whereas its height will depend on the layer height. The width of annotation inside the layer always is restricted by the layer width.

Text width/height definable

If you select this option, the annotation width and height will be independent of the layer width or height respectively. Then you can specify for each field the width and the number of lines individually in the **Edit Layer Format** dialog or by the properties **MinimumWidth** and **TextLineCount** in objects of the type **VcLayerFormatField**.

Start date field

Specify the start date of the selected layer, e.g. Early Start, Late Start, Scheduled Start.

Format

Opens the **Edit Layer Format** dialog.

End date field

In the end field line, specify the end date of the selected layer, e.g. Early Finish, Late Finish, Scheduled Finish.

To define a rectangle or line layer you need to specify a start and end field or a duration. If both an end field and a duration are specified, the duration entry overrides the end field entry. When an interaction occurs, not only the duration field will be updated, but also the end field.

Duration field

The unit of the duration will be interpreted in dependency on the time unit specified on the **General** property page. From the list, select the data field that contains the duration of the selected layer.

Percent complete

(not activated for symbol and bitmap layers) If you want the current layer to display the percentage degree of completion of an activity, select the data field that contains the percentage degree of completion of the selected layer.

The end date visualized by the layer is calculated from the start date field, the end date field or the duration respectively and the percent complete value. The data of the activity will not be changed.

Changeable

These options allow to set whether the user can move by the mouse a layer completely, the start of a layer and/or the end of a layer.

You can enable/disable three options to the user:

1. The layer start can be moved.
2. The whole layer (i.e. the start and end of the layer together) can be moved.
3. The layer end can be moved.

A button appearing pressed indicates that the options is enabled.

Snap targets

Specify whether the layer defines its start and/or end date as snap target.

Earliest start

Date and time of the selected field are considered the lower limit for the start time of the layer when interactively moving the layer or the node.

This feature can also be set by the property **VcLayer.MinimumStartDataFieldIndex**.

Latest end

Date and time of the selected field are considered the upper limit for the end time of the layer when interactively moving the layer or the node.

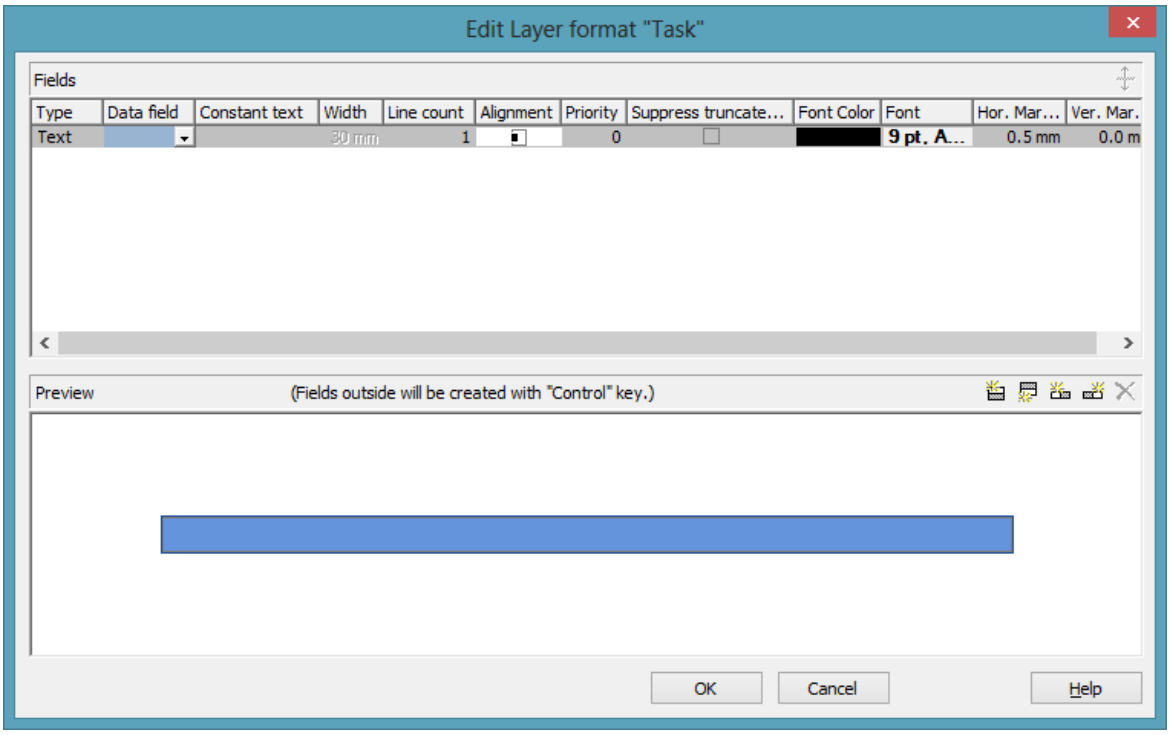
This feature can also be set by the property **VcLayer.MaximumEndDataFieldIndex**.

Preview

In the preview window the layer is displayed with its current settings.

In the preview, bar layers always will be interrupted by a solid line. This line shows how the layer will be displayed at run time, if workfree intervals are highlighted and if a calendar is assigned to the nodes. (These settings are made on the **Nodes** property page. Please note that they do not influence how the layer is displayed in the preview window of the **Edit Layer** dialog.)

4.15 The "Edit Layer Format" Dialog Box



You can get to this dialog box by the **Format** button of the **Edit Layer** dialog box.

Type

The field type (text) is displayed here.

Data field

Select the data field whose content is to be displayed in the current field. Additionally to the data fields defined in the data definition table, you can select the entry <Row number>: Then the number of the row containing the layer is displayed.

If the content of a data field does not fit into the current field, the excess will be cropped in the diagram.

Constant Text

(only if no data field has been specified) Type a constant text to be displayed in the current field.

Width

Specify the width for the selected field (in mm). The maximum width of a field is 90 mm:

Note: Only editable if **Text width/height definable** was selected in the dialog **Edit Layer**.

Line Count

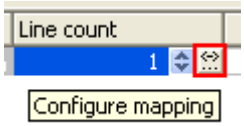
Specify the number of lines of text that can be displayed in the current field.

Note: Only editable if **Text width/height definable** was selected in the dialog **Edit Layer**.

For outside fields of a layer only: You can set the number of text lines dynamically, i.e. in dependence of the length of the text string. For this, two options exist:

1. You can have the number of lines calculated directly, store the results to a field and use them here
2. You can put down the number of lines in a map and assign it here

Case 1: You can have the number of lines calulated by the method **VcLayerFormatField.CalculateLineCount(...)** and store the results to a field. The field can be assigned by the **Configure mapping** dialog, which is to be invoked by pressing the right button that shows a double-headed arrow in the field **Line Count**:



In the dialog popping up, please select a data field from the top selection box and leave the map selection box below empty.

Case 2: For using a map, the map needs to be created and filled before it can be assigned; beside, the map type **vcNumberMap** is to be used. In a map of that type numbers are allocated to character strings. If the character strings put down here are found in a data field (still to be designated), the allocated number of lines will be displayed. Maps can be generated by the property page **Objects** and the button **Maps...**. In the **Configure mapping** dialog you can select a data field and a map, thus designating the data field the content of which is to be compared to the character strings of the map. You can view the content of the selected map in the dialog and modify it in continuative dialogs.

Alignment

Specify the alignment of the content of the selected field (left, centered, right).

Priority

Specify the priority of the layer field. Priority values between -9 and +9 are possible. If the total width of the layer is too small to show the contents of each layer field, the priority determinates of which layer field the content is displayed. At first, the content of the field with the highest priority is displayed completely, if possible. Then the contents of fields with smaller priorities is displayed. If it is not possible to display the content of a field, it will be suppressed or cropped (depending on the setting in **Suppress truncated text**).

Suppress truncated text

Specify whether a text that does not fit into the field is to be suppressed. Otherwise it will be cropped.

Font Color

Indicates the font color for the current field. If you click on the field, two buttons will appear:

by the arrow button you can open the Color picker to select a font color.

by the second button you reach the **Configure Mapping** dialog box. Here you can configure data-dependent font colors. If a mapping has been configured, the arrow on the button will be displayed in bold ( ).

Font

Indicates the font style for the current field. If you click on the field, two buttons will appear:

... The Windows **Font** dialog box will appear.

by the second button you reach the **Configure Mapping** dialog box. Here you can configure data-dependent fonts. If a mapping has been configured, the arrow on the button will be displayed in bold ( ).

Apply selected property to all fields

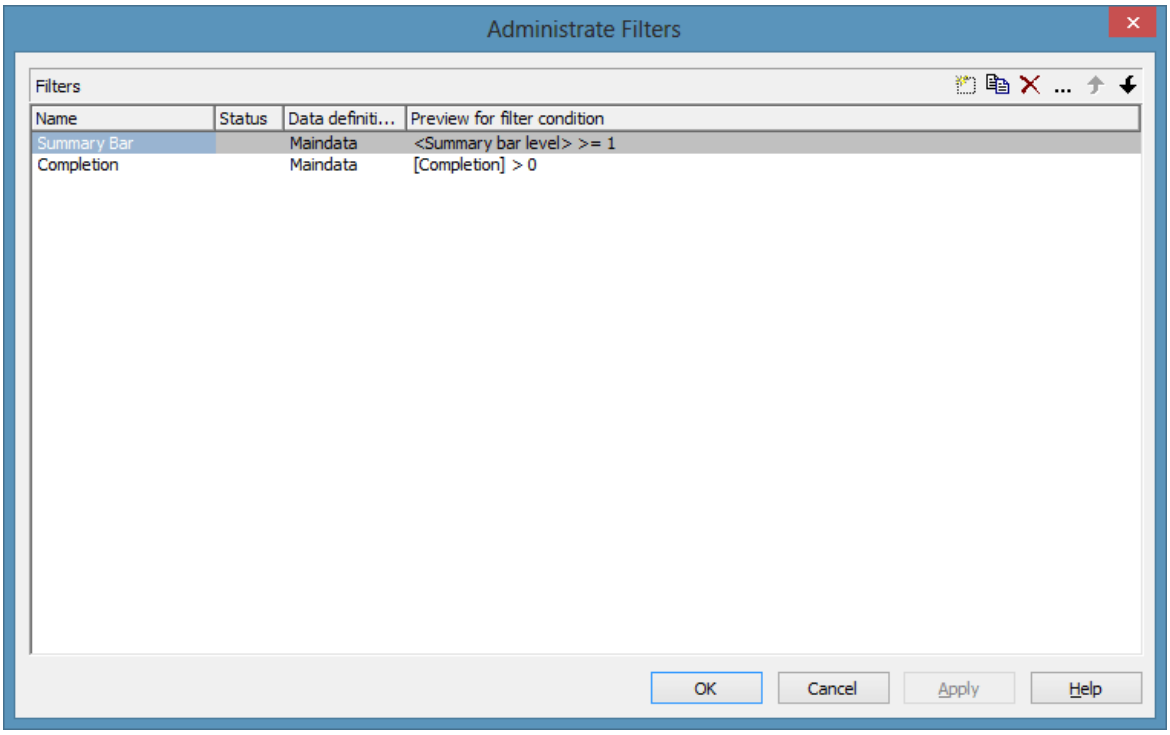
Applies the marked property to all fields.

Preview

The current fields are displayed in the preview window. If you click on a field, you can modify its attributes in the **Fields** table.

With the help of the buttons above the preview window you can add new fields or delete the marked field. If you want to add new fields outside of the layer, press the Ctrl button. You also can use the Del button to delete fields.

4.16 The "Administrate Filters" Dialog Box



You can get to this dialog box

- by the **Objects** property page
- for layers: by the **Specify Bar Appearance** dialog box
- for table formats: by **Edit Table** dialog box
- for links: by the **Filter** button of the **Link** property page
- for histogram curves: by the **Filter** select box of the **Edit Histogram** dialog
- for nodes: by the **Filter** button of the **Nodes** property page.

Name

Lists the names of all existing filters. The names can be edited.

Status

In the **Status** column all filters added ( ) or modified ( ) after the dialog box was opened are marked by a symbol.

Preview for filter condition

This column displays the conditions of the filters. Conditions cannot be edited in this dialog. To modify the filter condition, click on the **Edit filter** button.

Add filter

A new filter is created. You can modify its default name by double-clicking and editing it. New filters are created in a context-sensitive way, i. e. the matching data definition table will be used automatically.

Copy filter

Copies the selected filter.

Delete filter

The marked filter in the list will be deleted. You can only delete filters that are not currently used.

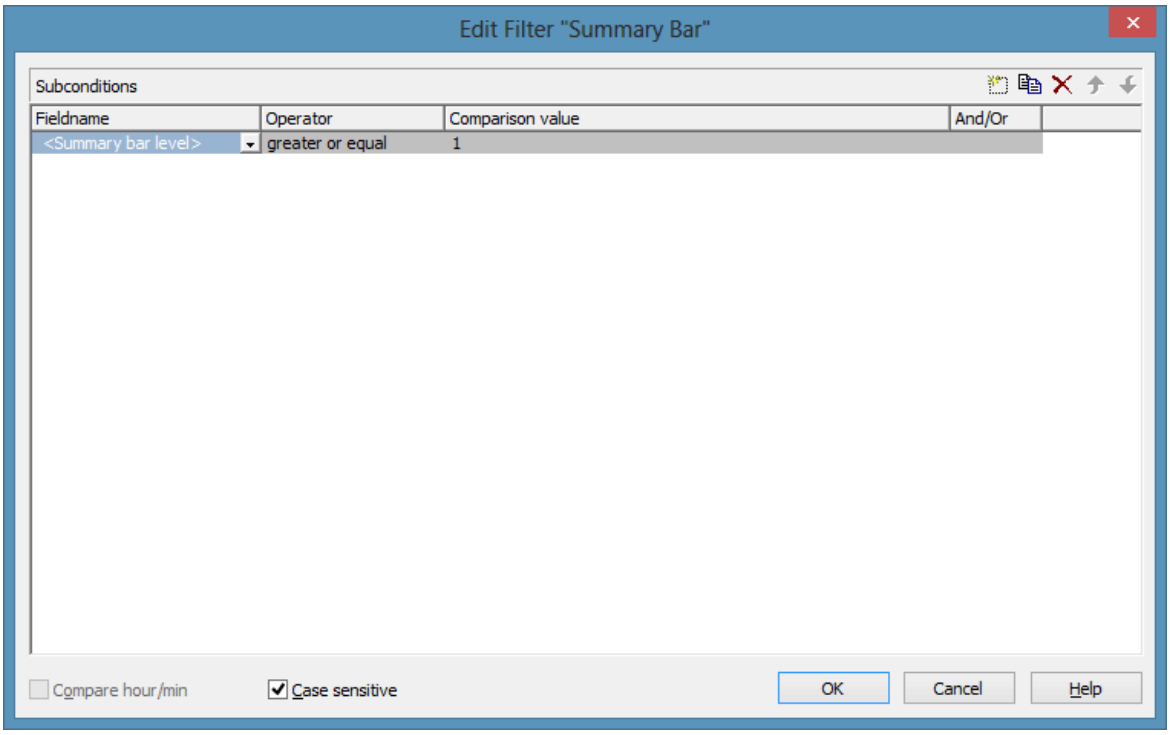
Edit filter

Press the **Edit filter** button to view or modify the condition of a filter. The **Edit Filter** dialog box will appear where you can edit the condition of the corresponding filter.

Promote / demote filter

By these buttons you can move the filter by one position up or down in the list.

4.17 The "Edit Filter" Dialog Box



You can get to this dialog box either

- by the **Objects** property page
- or by the **Administrate Node Appearances** dialog box
- or by the **Administrate Link Appearances** dialog box, where you can activate the **Administrate Filters** dialog box and then click on the **Edit filter** button. The head line of this dialog box displays the name of the filter being edited.

Add subcondition

Inserts a new line for a subcondition above the selected line.

Copy subcondition

Copies the selected subcondition.

Delete subcondition

Deletes the selected subcondition.

Evaluate subcondition earlier/later

If a filter consists of several subconditions, the subconditions are evaluated one after the other. The top subcondition in the table is evaluated first.

Click on the **Evaluate subcondition earlier/later** button to move the selected subcondition by one position upward or downward in the list.

Fieldname

This list contains all data fields available to be compared with the comparison value as well as the following predefined entries:

- The <summary bar level> entry can be used for displaying summary bars in Gantt diagrams. For example, specify a filter "<summary level> greater or equal 1" and assign it to a layer (e.g. "Summary level 1") in order to display summary bars for level 1. Please note that the option **Summary bars** in the **Edit Grouping** dialog has to be activated.
- Filters containing the <grouping level> entry can be used for example in the **Edit Table** dialog (for Gantt diagrams) as row filters for basic rows.
- <Gantt: collapsed>: entry for collapsed groups
- <Gantt: nodes in separate rows>: entry for displaying all nodes in separate rows
- <Gantt: nodes overlaid>: entry for displaying nodes overlaid, if necessary
- <Gantt: row>: entry to define filters for special rows
- <Gantt: summary node>: entry for summary bars
- <Node Read Only>: entry for defining filters for nodes that are defined as read only.

This feature can also be set at run time by the VcFilterSubCondition property **DataFieldIndex**.

Operator

The operator compares the value of a data field with a comparison value.

Comparison value

This column shows the current comparison value. The **Comparison value** select box lists all fields (in square brackets) that can be used as comparison values. The type of the data fields offered as comparison values correspond

to the data type of the data field specified in the **Fieldname** column. For example, if the data field "Early Start" is specified in the **Fieldname** column, for the comparison value you can select either a date field (e. g. "Early End") or the <today> option or the <input> option.

With the help of the <input> option you can specify a variable filter. In variable filters only the field name and the operator are specified, but not the comparison value. You can specify the comparison value when necessary. You can use a variable filter when you open a project and want to select the activities to be displayed.

Dates need to be entered in the format defined on the **General** property page. If you have selected a date field in the **Fieldname** field, two arrow buttons will appear as soon as you click on this field. The first arrow button lets you open a combobox with all available date data fields. The other arrow button opens a Date dialog box from which you can select a date by mouse-click. You can also edit the date direct.

Numeric values or texts must be typed manually into the **Comparison value** field.

With the operators "equal" and "unequal" you can use wildcards in text fields:

*: no sign or any number of signs

?: exactly one sign

If you do not want to use the signs * or ? as wildcards, but want to search for these signs, you have to set a backslash in front of them:

*: *

\?: ?

If the backslash does not follow a * or ?, the program searches for the sign \.

Examples:

Activity 1 : Name = "Construction"

Activity 2 : Name = "*Construction"

Possible filters for activity 1:

[Name] = C*

[Name] = C?nstruction

Possible filters for activity 2:

[Name] = *C*

[Name] = **

[Name] = ?C*

And/Or

This column shows the logical connection of two subconditions in the table.

Choose the AND operator to connect the current subcondition and the next subcondition in the table to select only those objects that fulfil both subconditions. Choose the OR operator to select those objects that fulfil at least one of the subconditions.

If you have formulated several subconditions, linking them partly with AND and partly with OR, the AND links will be processed first. (AND links are stronger than OR links).

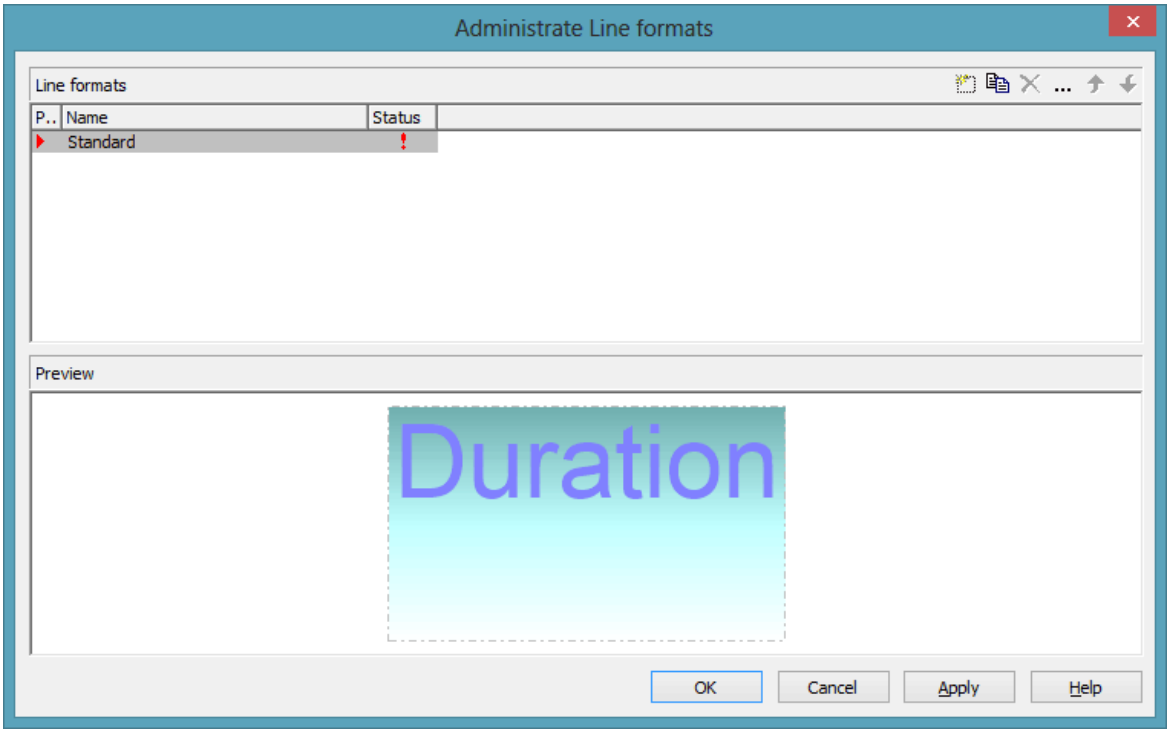
Compare hour/min

Activate this check box if the hours and minutes of a date are to be considered when dates are compared.

Case sensitive

Activate this check box if the comparison of the entries is to be case-sensitive.

4.18 The "Administrate Line formats" Dialog Box



You can get to this dialog box

- by clicking the corresponding button on the **Objects** property page
- by clicking ··· in the **Line format** field of the **Administrate Line grids** dialog.

Preview

In this column a red triangle marks the line format which is displayed in the preview below.

Name

Lists the names of all existing line formats. The names can be edited.

Status

In the **Status** column each filter that has been added ( ) and/or modified ( ) since the dialog box was opened is marked by a symbol.

Add line format

A new line format will be created. You can modify its default name by double-clicking and editing it.

Copy line format

Copies the selected line format.

Delete line format

The marked filter in the list will be deleted. You can only delete filters that are not currently used.

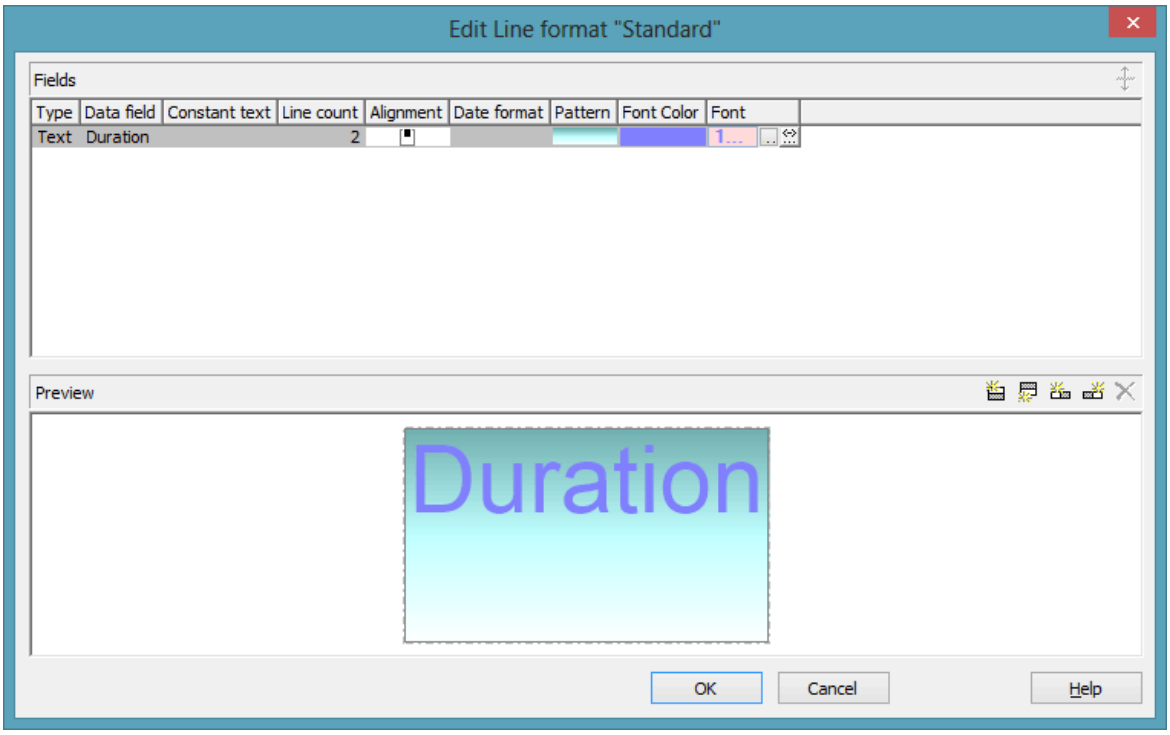
Edit Line format

Opens the dialog **Edit Line format** which lets you specify the attributes of the line format such as color, pattern etc.

Promote / demote filter

By these buttons you can move the line format by one position up or down in the list.

4.19 The "Edit Line format" Dialog Box



You can get to this dialog box

- by clicking the button **Line formats** on the **Objects** property page and then the button ··· in the **Administrate line formats** dialog
- by clicking ··· in the **Line format** field of the **Administrate Line grids** dialog.

Type

The field type (text) is displayed here.

Data field

Select the data field whose content is to be used as line grid annotation. In addition to the data fields defined in the data definition, you can select the entries <Date> or <Group title>: The current date or the group title (if grouping is switched on) is displayed.

If the content of a data field does not fit into the current field, the excess will be cropped in the diagram.

Constant Text

(only if no data field has been specified) Type a constant text to be displayed in the current field.

Line Count

Specify the number of lines of text that can be displayed in the current field.

Alignment

Specify the alignment of the content of the selected field (left, centered, right).

Date format

If you have selected <Date> as data field for the annotations, you can specify the date format here. To compose the date you can use the following tokens:

D: first letter of the day of the week (not adjustable)

TD: Day of the Week (adjustable by using the event **OnSupplyTextEntry**)

DD: two-digit figure for the day of the month: 01-31

DDD: first three letters of the day of the week (not adjustable)

M: first letter of the name of the month (not adjustable)

TM: name of the month (adjustable by using the event **OnSupplyTextEntry**)

MM: two-digit figure for the month: 01-12

MMM: first three letters of the name of the month (not adjustable)

YY: two-digit figure for the year

YYYY: four-digit figure for the year

WW: two-digit figure for the number of the calendar week: 01-53

TW: text for "calendar week" (adjustable by using the event **OnSupplyTextEntry**)

Q: one-digit figure for the quarter: 1-4

TQ: name of quarter (adjustable by using the event **OnSupplyTextEntry**)

hh: two-digit figure for the hour in 24 hours format: 00-23

HH: two-digit figure for the hour in 12 hours format: 01-12

Th: Text of "o' clock" (adjustable by using the event **OnSupplyTextEntry**)

TH: "am" or "pm" (adjustable by using the event **OnSupplyTextEntry**)

mm two-digit figure for the minute: 00-59

ss: two-digit figure for the second: 00-59

TS: short date format, as defined in the regional settings of the windows control panel

TL: long date format, as defined in the regional settings of the windows control panel

TT: time format, as defined in the regional settings of the windows control panel

xC/XC: You can set a maximum ten-place, simple upward counting from a reference date onward, for example "15:05:07:16:00", which equals 15 months, 5 days, 7 hours, 16 minutes, 0 seconds. The notation is: **xC44:C33:C22:C11:C00**. In written language: Show at least 2 digits for the counters 4...0 and a preceding "-" symbol if the value is negative. The separators are variable and can be replaced by other separators symbols. "x" means: Display a preceding "-" symbol if the value is negative, but no "+" symbol if it is positive. "X" means: Display a preceding "-" symbol if the value is negative and a "+" symbol for positive values. In the dialog **Edit Time Scale Section...** the check boxes **Use reference date** and **Adjust major ticks to reference date** need to be ticked, also, the parameter **Serial annotation** has to be set to **No**. In the application the reference date is set at run time by the call **VcRibbon.set ReferenceDate**, overriding any settings in the dialog.

Note: Characters which are not to be interpreted as part of the date should be preceded by a backslash '\'. '\\' for instance results in '\'. The special characters: **':, /, -'** and **blank** don't need '\' as prefix

Pattern

Here you can select the fill pattern and colors for the current field. By clicking on ··· you open the **Edit pattern attributes** dialog where you can specify a pattern, a background color and, if needed, a second pattern color

by clicking on . You can define your own colors in addition to the ones suggested. Transparent colors are also available.

By clicking on you open the **Configure Mapping** dialog box. Here you can configure data-dependent patterns and colors. If a mapping has been configured, the arrow on the button will be displayed in bold ( ).

Font Color

Indicates the font color for the current field. If you click on the field, two buttons will appear:

by the arrow button you can open the Color picker to select a font color.

by the second button you reach the **Configure Mapping** dialog box. Here you can configure data-dependent font colors. If a mapping has been configured, the arrow on the button will be displayed in bold ( ).

Font

Indicates the current font style. If you click on the field, two buttons will appear:

... The Windows **Font** dialog box will appear.

by the second button you reach the **Configure Mapping** dialog box. Here you can configure data-dependent fonts. If a mapping has been configured, the arrow on the button will be displayed in bold ( ).

Apply selected property to all fields

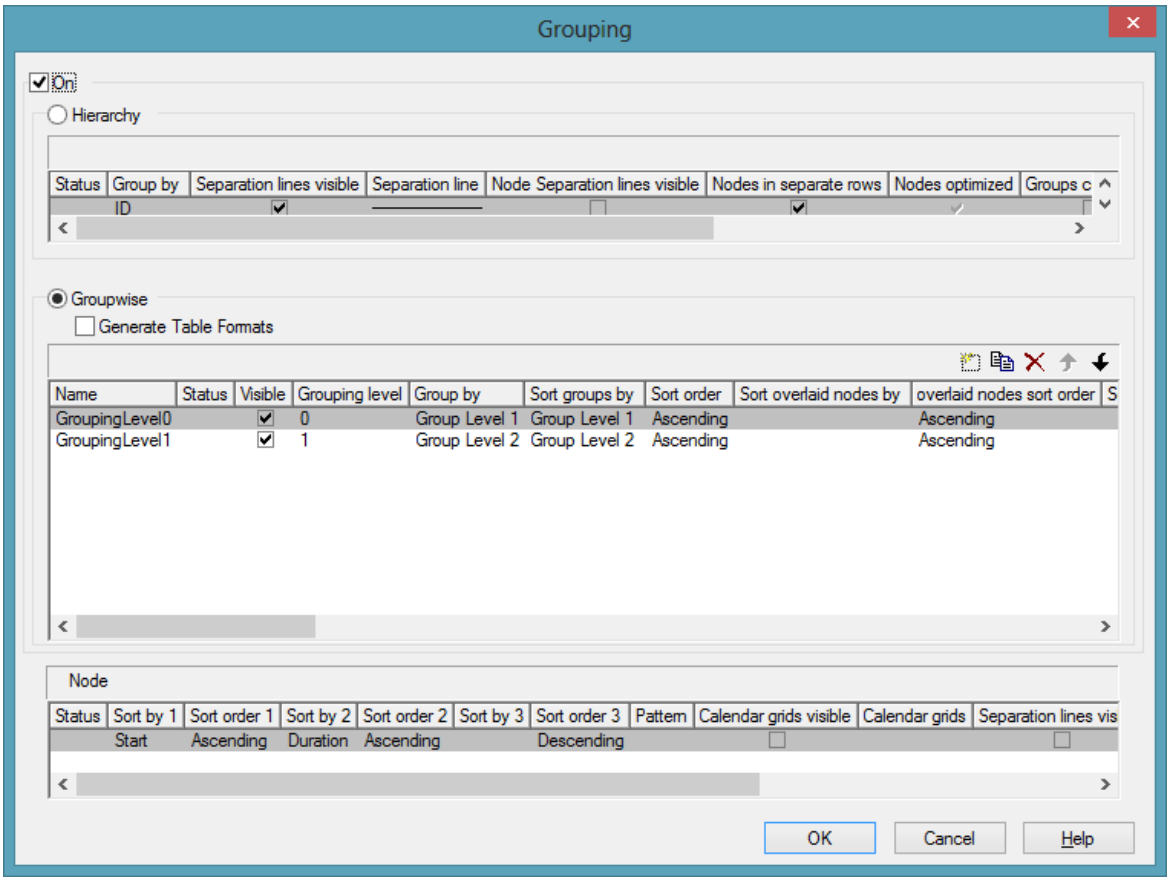
Applies the marked property to all fields.

Preview

The current annotations are displayed in the preview window.

With the help of the buttons above the preview window you can add new fields or delete the marked field. You also can use the **Del** button to delete fields.

4.20 The "Grouping" Dialog Box



In this dialog you can set options to hierarchical and grouping arrangements of nodes, sorting of nodes and to the layout of these structures.

The dialog shows three different sections: **Hierarchy**, **Groupwise** and **Nodes**, where you can set the corresponding options.

On

The grouping of nodes either in the form of a hierarchy (according to a hierarchy code) or in the form of grouping according to different criteria is switched on or off.

> Hierarchy

If you activate this radio button, the activities will be arranged in a hierarchy, according to a hierarchy code. In the code, hierarchy levels are separated by dots. If you select this option, the section **Groupwise** automatically will become inactive.

In the table below the **Hierarchy** button you can make further settings concerning the hierarchical arrangement.

Group By

Select the data field which contains the code by which the activities are arranged.

Separation lines visible

Tick this box to display separating lines between different hierarchical levels.

This feature can also be set by the property **VcHierarchyLevelLayout.-ShowSeparationLines**.

Separation line

By clicking on ... you can open the dialog **Line attributes** and specify the style of the separation lines.

The line attributes can be also set by the corresponding properties **VcHierarchyLevelLayout.SeparationLineColor**, **VcHierarchyLevel-Layout.SeparationLineThickness** and **VcHierarchyLevelLayout.-SeparationLineType**.

Nodes in headers

Specify whether each node of a group will be displayed in a separate row or not.

If this option is activated, the table section of the activities is suppressed, so you will need to use the layer format or tooltip to identify the activities for the user.

Nodes overlaid

Specify whether the node layout on this hierarchy level is to be optimized or if nodes overlap.

Groups collapsed

If you select this option, the 2nd and all further levels will be displayed initially collapsed when the program is started. They can be expanded interactively.

Summary Bar

If you tick this box, summary bars will be displayed for all levels. If you want to display summary bars only for special levels, you have to define a layer with an appropriate filter condition (<Sum bar level = ...).

This feature can also be set by the property **VcHierarchyLevelLayout.-SummaryBarsVisible**.

Collapse groups automatically

If you tick this box, every group save the one just being touched will be collapsed when a node/a group is being moved interactively.

Restore automatically collapsed groups

When this check box is ticked every group that was automatically collapsed before is restored again when a node/a group is being moved interactively.

Expand target group automatically

When this check box is ticked the target group is expanded automatically when a node/a group is being moved interactively.

Restore automatically expanded group

When this check box is ticked every group that was automatically expanded before is restored again when a node/a group is being moved interactively.

Pagebreak after Group

After clicking on , the following options can be selected:

- **None**: no page break will be inserted
- **On page full**: if a group would be separated by a page break, the page break will be inserted after the preceeding group already
- **After each group**: a page break is inserted after each group

This features can also be set by the property **VcHierarchyLevelLayout.-PageBreakMode**.

Maximal level for pagebreaks

Here you can specify up to which hierarchy level page breaks after each group are to be carried out. If the level is set to 4, for example, no page break will be carried out after level 4.

If the level is set to the default -1, page breaks are carried out on each level.

This feature can also be set by the property **VcHierarchyLevelLayout.-LevelMaximumForPagebreaks**.

> Groupwise

If you activate this radio button, the activities will be arranged in groups (grouped by different criteria) and the section **Hierarchy** automatically will become inactive.

In the area below the <bGroupwise button you can set all further grouping options - mostly concerning the layout (pattern, calendar grid, line grid etc.). You can define different settings for each grouping level. By clicking on the corresponding buttons  levels can be created, deleted, copied or the order of the levels can be changed.

Generate Table Formats

If this check box is activated, for each grouping level an own table format will be created: Subtitle_n, Collapsed_n. The formats probably have to be adapted by the dialog **Edit table format**, especially the data field.

If this check box is not activated, no table formats will be created for new grouping levels. You may have to create them yourself, if required. This option is helpful, because it allows to get along with only two table formats for grouping (Subtitle and Collapsed) that you can modify by maps and filters.

Name

Specify a name for the corresponding grouping level.

Visible

Specify whether or not the groups of this level are to be displayed.

Grouping level

The level, for which the settings of this line are valid is displayed here. You can change the order of the levels by clicking on the corresponding arrow buttons above the table.

Group by

Select the data field by which the activities on the current grouping level are to be grouped. If you leave this field blank, the activities on the current grouping level will not be grouped.

Sort groups by

Select the data field by which the groups on the current grouping level should be sorted when the program is started. If you do not set anything here, the sequence of the nodes will derive from the sequence of loading.

Sort order

Set the sorting order (ascending or descending) on the current grouping level.

Sort overlapping nodes by

Select the data field by which the nodes of a group that are put in a single row are to be sorted. If you do not set anything here, the sequence of the nodes will derive from the start date and the duration of the activities, i.e. the earliest and the shortest activities will be farthest in front. This property can only apply if the property **VcGroupLevelLayout.NodesArranged-Optimized** was set to **False**.

Overlapping nodes sort order

Set the sorting order (ascending or descending) of the overlapping nodes.

Sort optimized nodes by

Select the data field by which the nodes of a group that are put in a single row are to be sorted. If you do not set anything here, the sequence of the nodes will derive from the start date and the duration of the activities, i.e. the earliest and the shortest activities will be farthest in front. This property can only apply if the property **VcGroupLevelLayout.NodesArranged-Optimized** was set to **True**.

Optimized nodes sort order

Set the sorting order (ascending or descending) of the optimized nodes.

Pattern

If you click on ... you open the dialog **Pattern attributes**. Here you can specify the background pattern and two pattern colors of the group title row as well as by clicking on assign the respective property in dependence on data.

Calendar

Select the data field that contains the name of a calendar, which should be used for the group node.

Line grid visible

Specify whether a line grid is displayed.

Line grid with subgroups

Specify, whether the line grid shall be displayed for subgroups as well.

Line grids

By clicking on you can select a line grid for the grouping level or create a new one in the **Administrate Line grids** dialog which you can open by clicking on ... . For further information about line grids see chapter **The Administrate Line grids** dialog.

Calendar grid visible

Specify whether a calendar grid is displayed. For that the property **VcCalendarGrid.Visible** needs to be set to **True**.

Calendar grid with subgroups

Specify, whether the calendar grid shall be displayed for subgroups as well.

Calendar grids

By clicking on you can select a calendar grid for the group or create a new one in the **Administrate Calendar grids** dialog which you can open by clicking on ···. For further information about calendar grids see the chapter **The Administrate Calendar grids** dialog.

If you select <From Scale> the first not visible calendar grid from the time scale will be displayed.

Separation lines visible

Tick this box to display separating lines between different groups.

Separation lines at top

If you tick this box, the separation line will be drawn above a group (instead of below).

Separation line

You can edit the appearance of the separating lines after clicking on the **Edit** button.

Nodes in headers

Specify whether each node of a group will be displayed in a separate row or not.

If this option is activated, the table section of the activities is suppressed, so you will need to use the layer format or tooltip to identify the activities for the user.

Nodes overlaid

Specify whether the node layout on this group level is to be optimized or if nodes overlap.

Groups collapsed

If you select this option, the groups will be displayed initially collapsed, i. e. only the group titles will be visible, but not the nodes.

Modifications allowed

If you tick this box, the user can collapse expanded groups and vice versa. The user can collapse/expand groups by double-clicking on the group heading in the table section, by clicking once on the minus or plus symbol next to the group heading or by the context menu of a group.

Summary bar

If you tick this box, summary bars will be displayed. To specify summary bars for a specific level, you have to define a layer with an appropriate filter condition (<Sum bar level = ...).

Group node visible

Tick this box to display bars in the diagram for those groups coming from a separate group data table. For that purpose you also have to tick the **Extended data tables** option on the **General** property page before.

Moving groups vertically via tables

When this check box is ticked you can change the order of groups by drag interactions in the table area.

Collapse groups automatically

If you tick this box, every group save the one just being touched will be collapsed when a node is being moved interactively.

Restore automatically collapsed groups

When this check box is ticked every group that was automatically collapsed before is restored again when a node is being moved interactively.

Expand target group automatically

When this check box is ticked the target group is expanded automatically when a node is being moved interactively.

Restore automatically expanded group

When this check box is ticked every group that was automatically expanded before is restored again when a node is being moved interactively.

Moving groups vertically via diagram

When this check box is ticked you can change the order of groups by drag interactions in the diagram area.

Pagebreak after Group

After clicking on , the following options can be selected:

- **None**: no page break will be inserted
- **On page full**: if a group would be separated by a page break, the page break will be inserted after the preceeding group already
- **After each group**: a page break is inserted after each group

This features can also be set by the property **VcGroupLevelLayout.Page-BreakMode**.

> Nodes

The below settings describe the options that you can select for grouped or ungrouped nodes concerning in particular sorting options as well as the layout of the node rows.

Note: Please note that the settings for the sorting of the activites are only valid when opening the diagram. If you want to sort the activities again later, please use the VcGantt method **SortNodes**.

Sort by 1 to 3

Specify the data fields by which the activities are to be sorted when the diagram is opened. You can sort the activities by up to three data fields, in ascending or descending order respectively (**Sort Order 1 to 3**).

If you specified a data field by which the activities are to be grouped (**Grouping by**), each group will be sorted separately.

Pattern

If you click on ... you open the dialog **Pattern attributes**. Here you can specify the background pattern and two pattern colors of the node line as well as by clicking on assign the respective property in dependence on data.

Calendar grid visible

Specify whether a calendar grid is displayed.

Calendar grids

By clicking on you can select a calendar grid for the node or create a new one in the **Administrate Calendar grids** dialog which you can open by clicking on ···. For further information about calendar grids see the chapter **The Administrate Calendar grids** dialog.

If you select <From Scale> the first not visible calendar grid from the time scale will be displayed.

Separation lines visible

Specify whether a separation line is displayed.

Separation lines at top

If you tick this box, a separation line will be drawn above a node (instead of below).

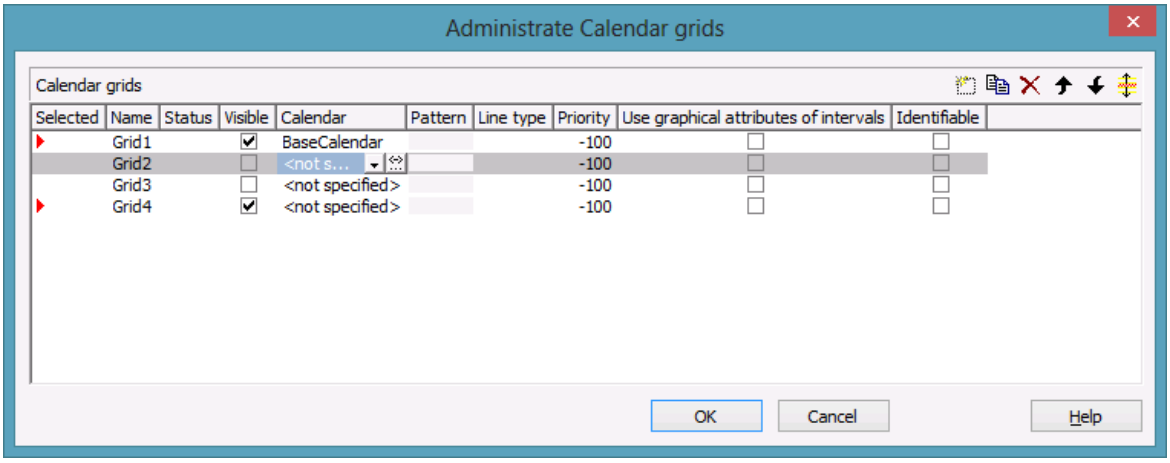
Separation line

The layout of the separation lines can be edited in the **Line attributes** dialog box which appears when you click on the **Edit** button.

Separation lines step size

Specify after how many activities a separating line is drawn.

4.21 The "Administrate Calendar grids" Dialog Box



You can get to this dialog by clicking on ··· in the field **Calendar grids** in the dialog **Grouping**, section **Groupwise**.

By clicking on the corresponding buttons you can add, copy or delete calendar grids.

The arrow buttons allow to move a calendar grid by one line down or up, while the button lets you assign the feature just activated to all calendar grids listed.

The below features can be set to calendar grids:

Selected

By clicking on this field you can select this calendar grid to apply to the grouping level. A red arrow indicates that this calendar grid was selected.

Name

Enter a name for the calendar grid.

Status

Status: In this column, each calendar grid that was added ( ) and/or modified ( ) after opening the dialog box is marked by a symbol.

Visible

Activate this check box for the calendar grids to be displayed.

Calendar

The calendar selected here will apply to all groups of this level. If no calendar is selected here, the calendar of the level to which the calendar grid was assigned will apply.

Pattern

Select the fill pattern and color for the calendar grid. By clicking on ··· you open the **Edit pattern attributes** dialog where you can specify a pattern, a background color and, if needed, a second pattern color . You can define your own colors in addition to the ones suggested. Transparent colors are also available.

Line type

When clicking on this button (···), the **Line attributes of calendar grid** dialog box will appear, where you can enter the settings of the border lines of the calendar grid.

Priority

Lets you set the priority of a calendar grid. It refers to other calendar grids and to layers (> 0: in front of the layers, < 0: behind the layers).

Use graphical attributes of intervals

Specify whether the graphical attributes that have been set for the intervals are to be displayed.

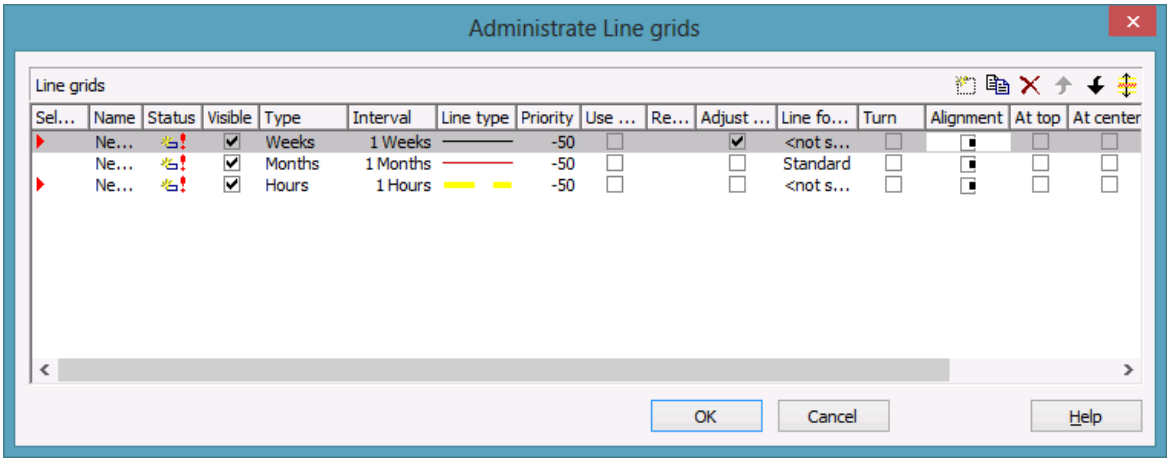
Identifiable

This option allows to set whether or not the calendar grid can be identified by the VcGantt method **IdentifyObjectAt**. A tool tip text for instance can only appear if a calendar grid can be identified; the same is valid for the context menu popping up on right-clicking the mouse. For a tool tip text to appear, the corresponding interval also has to be identifiable; please see the **Calendar grid** section in the **Edit time scale section** dialog.

Snap targets start/end

Tick this check box to have the calendar grid's relevant positions defined as "snap targets" for nodes/layers to be moved.

4.22 The "Administrate Line grids" Dialog Box



You can get to this dialog by clicking on ··· in the field **Line grids** in the dialog **Grouping**, section **Groupwise**.

By clicking on the corresponding buttons you can add, copy or delete line grids.

The arrow buttons allow to move a line grid by one line down or up, while the button lets you assign the feature just activated to all line grids listed.

The below features can be set to line grids:

Selected

By clicking on this field you can select this line grid to apply to the grouping level. A red arrow indicates that this calendar grid was selected.

Name

Enter a name for the line grid.

Status

In this column each line grid grid that was added ( ) and/or modified ( ) after opening the dialog box is marked by a symbol.

Visible

Tick this check box for the line grids to be displayed

Type

Lets you set the basic unit of the line grid, e.g. days, weeks, etc.

Interval

Lets you set the size of the interval between the grid lines as an integer multiple of the basic unit of the grid.

Line type

When clicking on the button in this field, the **Line attributes of line grid** dialog box will appear, where you can set shape and color of the borderlines of the line grid.

Priority

Lets you set the priority of a line grid. It refers to other line grids and to layers (> 0: in front of the layers, < 0: behind the layers.

Use reference date

Tick this check box if the start value of the line grid should coincide with the reference date selected.

Reference date

Select the reference date from the date picker.

Adjust to reference date

Tick this check box to position the line grid on a different value of the time unit, i.e. the one defined by the reference date, for example on 13:17 of a day.

If this option is not selected, the lines of a line grid are positioned on the beginning of a time unit, for example on 00:00 h of a day.

Line format

By clicking on you can select a line format for the line grid or create a new one in the **Administrate Line formats** dialog which you can open by clicking on ···. For further information about line formats see the chapter **The Administrate Line formats** dialog.

Turn

If you tick this check box, the annotations at the lines of the date line grid can be turned by 90 degrees (vertically).

Alignment

Here you can specify the horizontal alignment of the line annotations.

At top

Tick this check box to position the annotations of the lines in the line grid at the top of the Gantt graph.

At center

Tick this check box to position the annotations of the lines in the line grid at the center of the Gantt graph.

At bottom

Tick this check box to position the annotations of the lines in the line grid at the bottom of the Gantt graph.

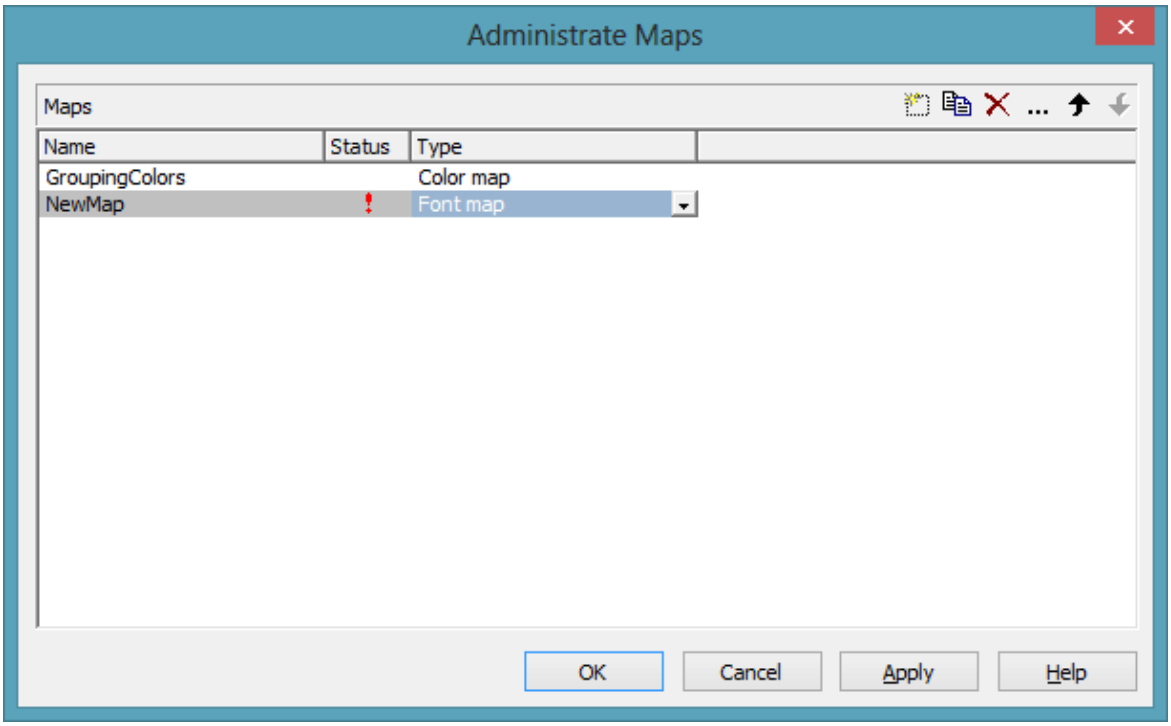
Observe DST

Tick this check box to have daylight saving time observed.

Snap target

Tick this check box to have the line grid's relevant positions defined as "snap targets" for nodes/layers to be moved.

4.23 The "Administrate Maps" Dialog Box



You can invoke this dialog by clicking the **Maps** button either on the **Objects** property page or in the **Configure Mapping** dialog box.

Name

This column lists the names of all existing maps. All names can be edited.

Status

In the **Status** column each map that has been added ( ) and/or modified ( ) since the dialog box was opened is marked by a symbol.

Type

Select the map type:

- Color maps
- Pattern maps
- Graphics file maps
- Fonts
- Millimetres

- Number map

Add map

A new map will be created. You can modify its default name by double-clicking and editing it.

Copy map

Copies the selected map.

Delete map

The marked map in the list will be deleted. You can only delete maps that are not currently used.

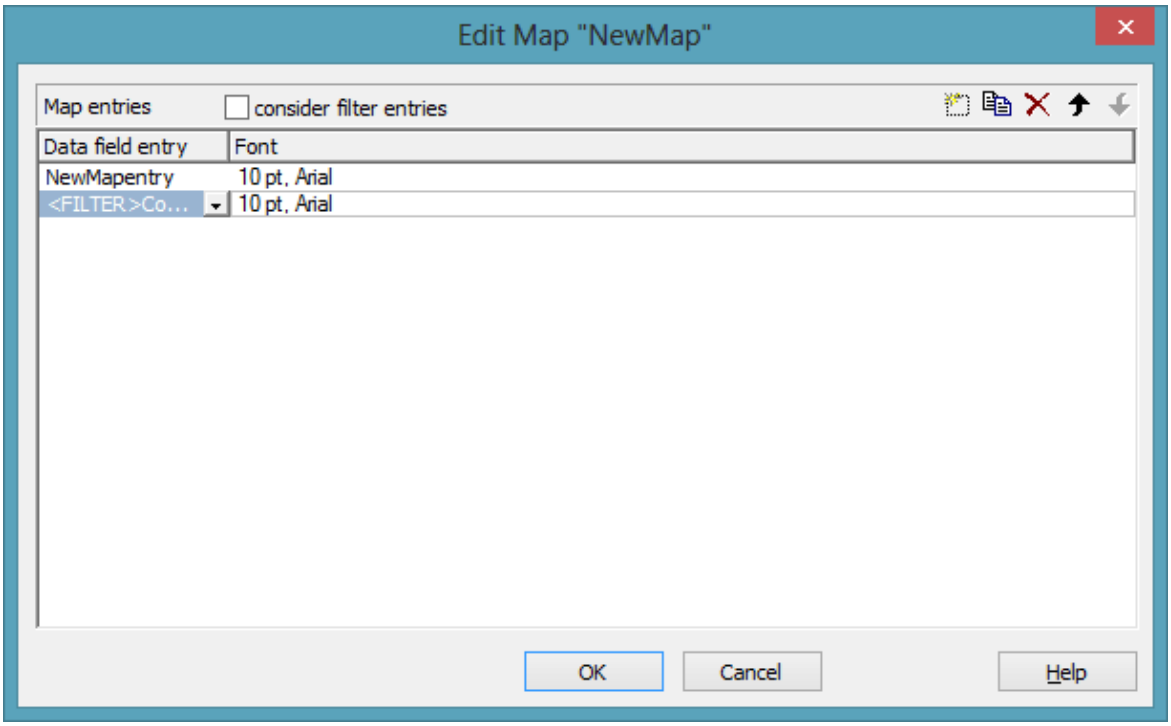
Edit map

The **Edit Map** dialog box will appear.

Promote / demote map

By these buttons you can move the map by one position up or down in the list.

4.24 The "Edit Map" Dialog Box



You invoke this dialog box by clicking the **Edit map** button (...) of the **Administrate Maps** dialog box.

In a map you can set up to 150 allocations. If you wish to set more allocations, please create a new map, e. g. as a copy of an existing one.

consider filter entries

If you have ticked this check box, not only the single values from the list of data field entries are considered as keys but also the filters which can be selected from the drop down list. Thus you can not only specify a single value as key but also a range of values.

Data field entry

Specify the entries of the data field selected for which colors or patterns and legend texts are to be assigned.

Color

To assign a color to a data field entry, please click in the corresponding field in the **Color** column. A dialog box will open that lets you select a color. Also transparent colors are available.

Legend text

Enter a legend text for each data field entry.

Add map entry

A new map entry will be created. You can modify its default name by double-clicking and editing it.

Copy map entry

Copies the selected map entry.

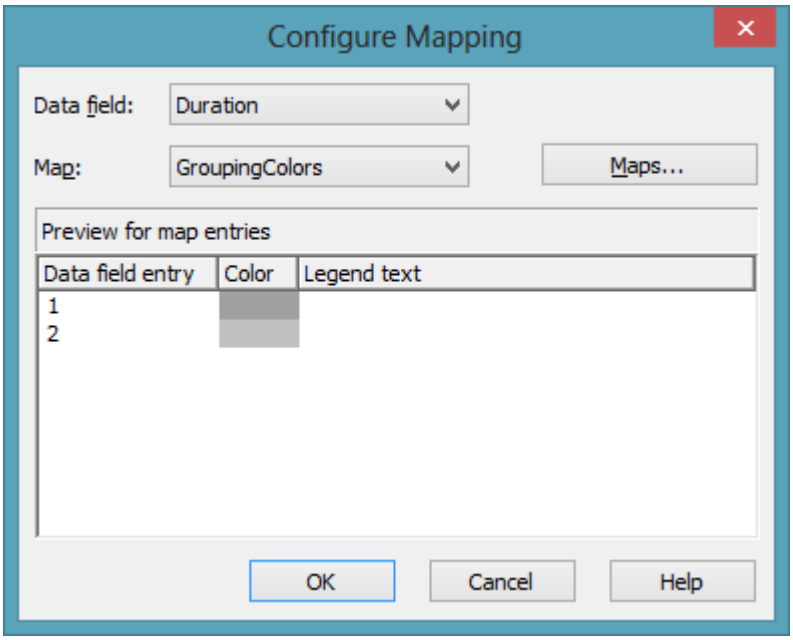
Delete map entry

The marked map entry in the list will be deleted. You can only delete map entries that are not currently used.

Promote / demote map entry

By these buttons you can move the map entry by one position up or down in the list.

4.25 The "Configure Mapping" Dialog Box



In this dialog box you can assign a map to a data field. You will get to it by clicking on the button for the desired attribute in the dialog **Edit layer**.

Data field

Select the data field the entries of which control the desired attributes of the current object.

Map

(only activated if a data field has been specified) Select the map that depending on its type assigns the corresponding attributes to each data field entry.

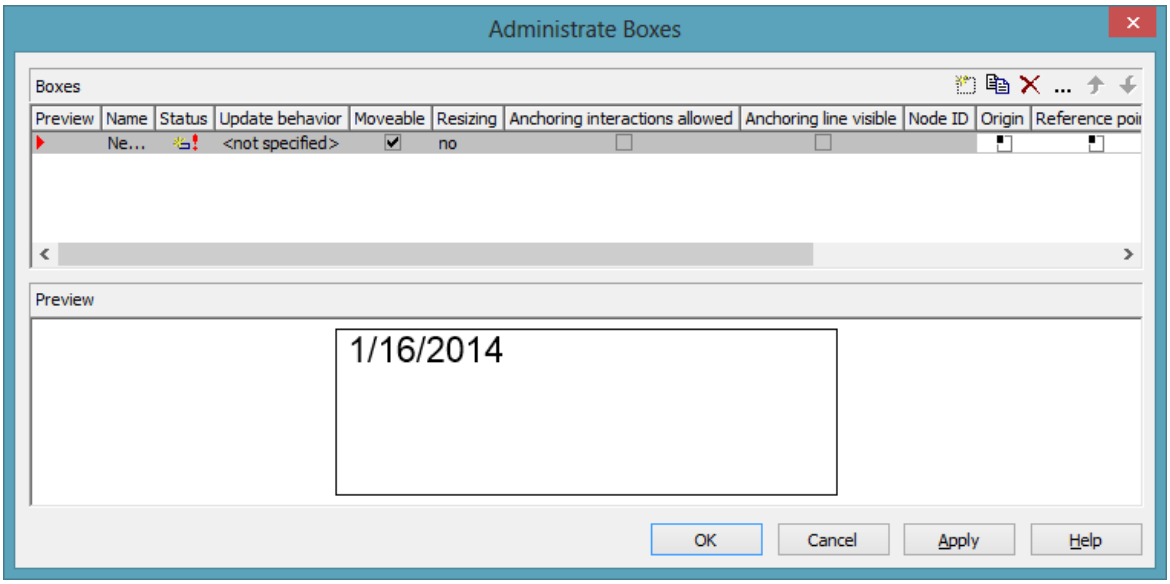
Maps

Opens the **Administrate Maps** dialog box, where you can create, edit, copy or delete maps.

Preview for map entries

The preview shows the selected map: the data field entries and the colors or patterns respectively and legend texts assigned to the data field entries.

4.26 The "Administrate Boxes" Dialog Box



You can get to this dialog box by the **Objects** property page. In the diagram area, boxes can be displayed, that you can administer by the above dialog.

Preview

The preview window shows the box marked in the **Preview** column.

Name

Lists the names of all existing boxes. The names can be edited.

Status

In the **Status** column each box that has been added ( ) and/or modified ( ) since the dialog box was opened is marked by a symbol.

Update behavior

Select an update behavior for this box. Leaving the setting to <not selected> means that the setting for boxes made in the **Edit Update behavior** dialog will apply

Moveable

By moving a box, its offset will be modified. Activate this check box if the box is to be moveable in the diagram at run time. Deactivate the check box if

you have positioned a box correctly and do not want it to be moved at run time.

Resizing

Here you can specify whether the size of a box can be modified interactively. You can select whether only height, only width or both height an width can be modified. When the pointer is placed on the frame of the box, its form changes to a double-headed arrow. Now hold the left mouse button pressed and change width and/or heigth by moving the mouse in the desired direction.

Tipp If you have selected **width and height** you can place the pointer on the corner of the box and both dimensions can be modified at the same time.

Anchoring interactions allowed

Specify whether anchoring interactions (by mouse or context menu) are possible. Thus the user can tie boxes to nodes or untie them again.

Anchoring line visible

Specify whether a line between the reference points (origin, reference point) of a node and of a box which are anchored is displayed.

Node ID

Here you can enter a string which is interpreted as Node ID and is used for identifying the node to which the respective box shall be tied. An empty string implicates that the box will not be anchored to a node.

Note: It is neither checked whether the syntax of the string is correct nor whether the node exists. If the node does not exist, no anchoring will take place.

Origin

By the properties **Origin**, **Reference point**, **X Offset** and **Y Offset** you can position a box in the diagram area. The relative position of the boxes is independent of the current diagram size.

Specify the origin, i. e. the point of the diagram from which the offset to the reference point of the box will be measured. Possible values: top left, top

centered, top right, centered left, centered centered, centered right, bottom left, bottom centered, bottom right.

Reference point

Set the reference point of the box, i. e. the point of the box from which the offset to the origin will be measured. Possible values: top left, top centered, top right, centered left, centered centered, centered right, bottom left, bottom centered, bottom right.

X Offset

Set the distance (in mm) between origin and reference point in x direction.

Y Offset

Set the distance (in mm) between origin and reference point in y direction.

Frame

If you click on the **Frame** field, an **Edit** button appears that lets you open the **Line Attributes** dialog box. In this dialog box you can specify the type, the thickness and the color of the box frame line.

Priority

Specify the relative drawing priority of the box in comparison with the other objects in the diagram (nodes, grids, etc.). The priority of nodes is 0. If the priority of a box is higher than the priority of nodes, the boxes overlay the nodes so that an interactive access to the nodes won´t be possible.

Visible

Activate this check box if the box is to be visible at run time.

Box format

The current box format of the box is displayed here. If you click this field, two buttons will appear:

From the combobox you can select a box format.

by the **Edit** button you reach the **Administrate Box Formats** dialog box.

Add box

A new box will be created. You can modify its default name by double-clicking and editing it.

Copy box

A copy of the selected box under a new name is created.

Delete box

The marked box in the list will be deleted.

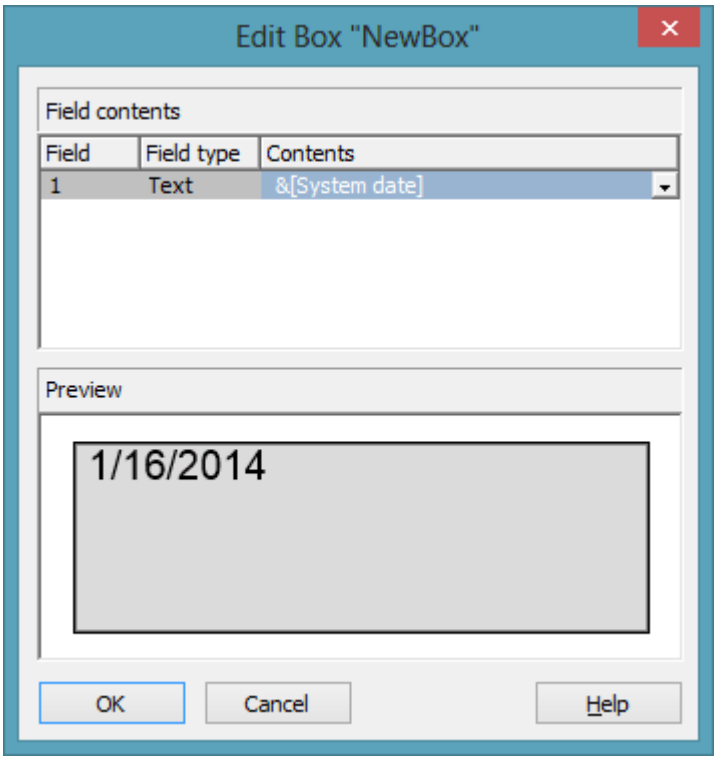
Edit box

The **Edit Box** dialog box will appear.

Promote / demote box

By these buttons you can move teh box by one position up or down in the list.

4.27 The "Edit Box" Dialog Box



You can get to this dialog by the **Objects** property page and the dialog box **Administrate Boxes** by clicking on the the **Edit box** button. This dialog box will also appear at run time when double-clicking on a box.

Field

This column contains the numbers of the box fields. (The number of fields depends on the selected box format.)

Field Type

This column displays the field types (text or graphics).

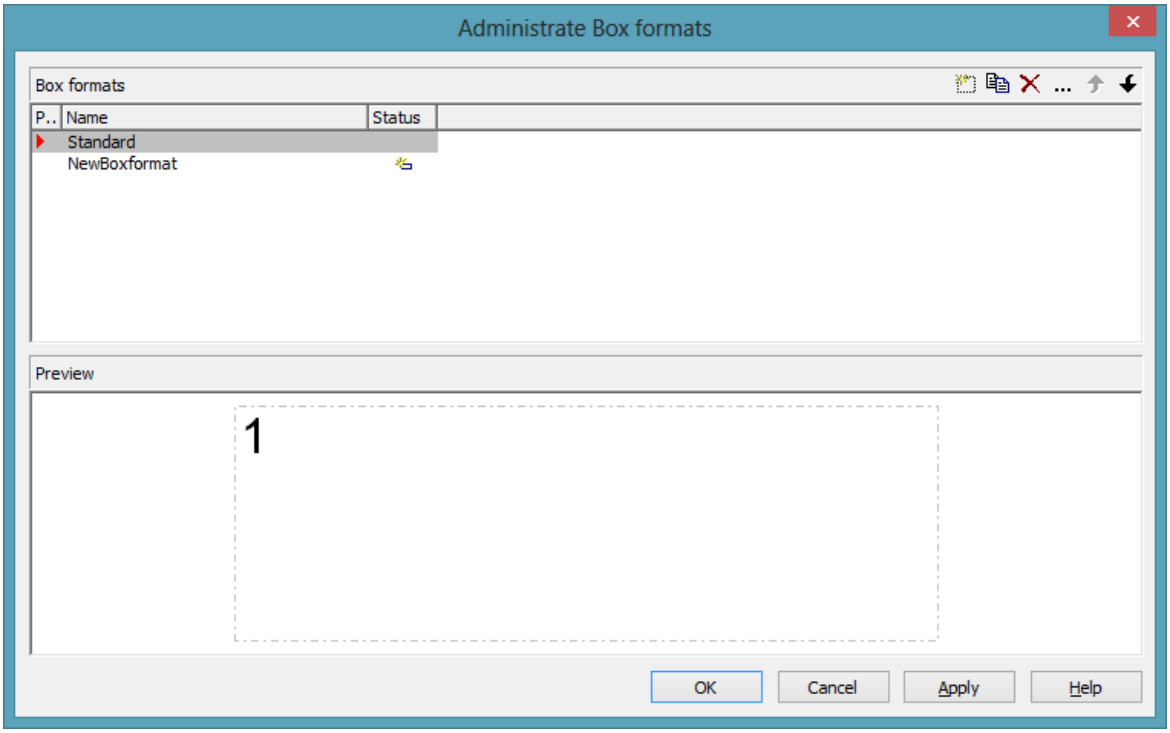
Contents

Type the contents of the field or a graphics file name here.

If a text field contains more than one line, you can use "\n" in the text string to separate two lines of the text field (Example: "Line1\nLine2"). Otherwise the lines will be separated at blanks.

Graphics formats available: WMF, JPG, BMP, GIF, PCX, PNG, TIF.

4.28 The "Administrate Box Formats" Dialog Box



You can get to this dialog box by the **Objects** property page.

Preview

The preview window shows the format marked in the **Preview** column.

Name

Lists the names of all existing formats. The names can be edited.

Status

In the **Status** column each format that has been added ( ) and/or modified ( ) since the dialog box was opened is marked by a symbol.

Add box format

A new format will be created. You can modify its default name by double-clicking and editing it.

Copy box format

A copy of the selected format under a new name is created.

Delete box format

The marked format in the list will be deleted. You can only delete formats that are not currently used.

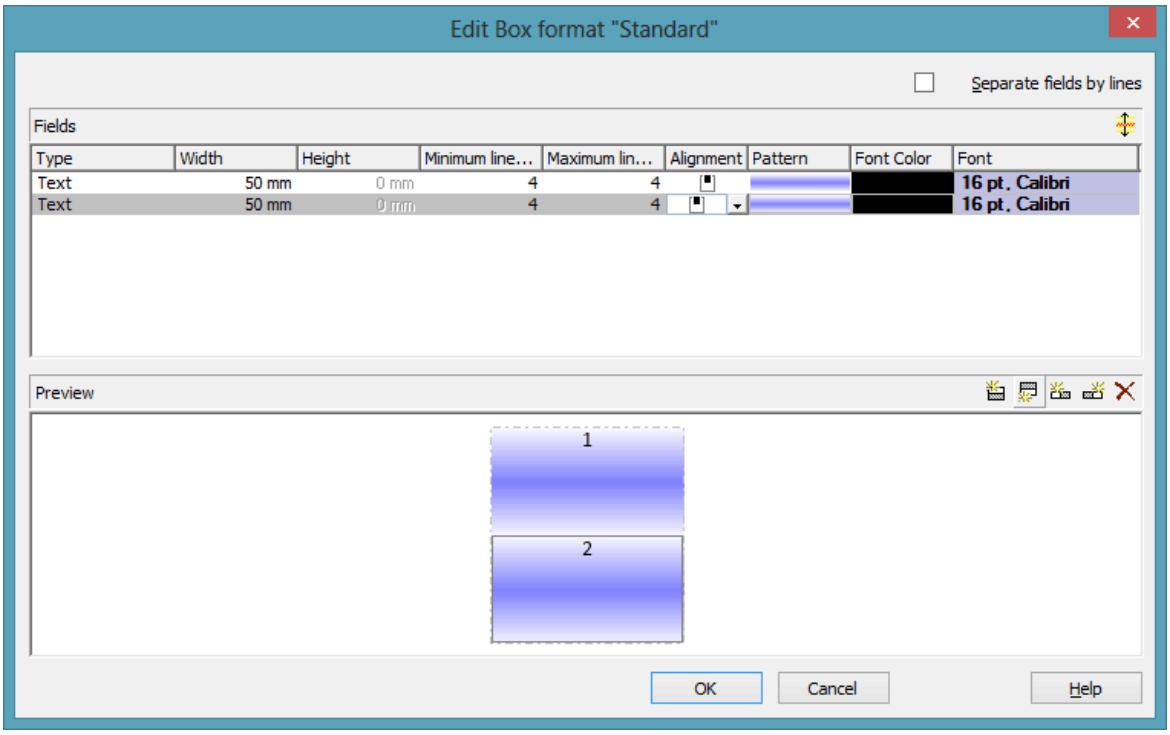
Edit box format

The **Edit Box Format** dialog box will appear.

Promote / demote box format

By these buttons you can move the format by one position up or down in the list.

4.29 The "Edit Box Format" Dialog Box



This dialog box will appear if you activate the **Administrate Box Formats** dialog box on the **Objects** property page and then click on the **Edit box format** button.

Separate fields by lines

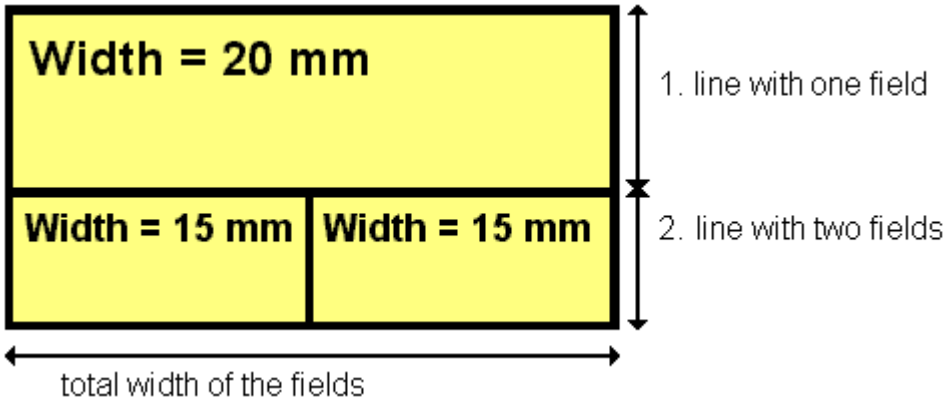
Activate this check box if the box fields are to be separated by lines.

Type

Select the field type: text or graphics.

Width

Specify the width for the selected field (in mm). The maximum width of a field is 200 mm. If the rows are split into two or more fields and the total widths of the rows vary, the total width will be equal to the width of the widest row.



Height

(only for the type graphics) Specify the minimum height for the selected field (in mm). The maximum height is 200 mm.

Minimum/Maximum line count

(only for the type text) Specify the minimum/maximum number of lines of text that can be displayed in the current field. Each field can contain a maximum of nine lines of text.

Alignment

Specify the alignment of the content of the selected field (9 possibilities).

Pattern

Select the fill pattern and color for the current field. By clicking on ... you open the **Edit pattern attributes** dialog where you can specify a pattern, a background color and, if needed, a second pattern color . You can define your own colors in addition to the ones suggested. Also, transparent colors are available.

Font Color

(only for the type text) Indicates the font color for the current field.

by the arrow button you can open the Color picker to select a font color.

Font

(only for the type text) Indicates the font style for the current field.

... The Windows **Font** dialog box will appear.

Apply selected property to all fields

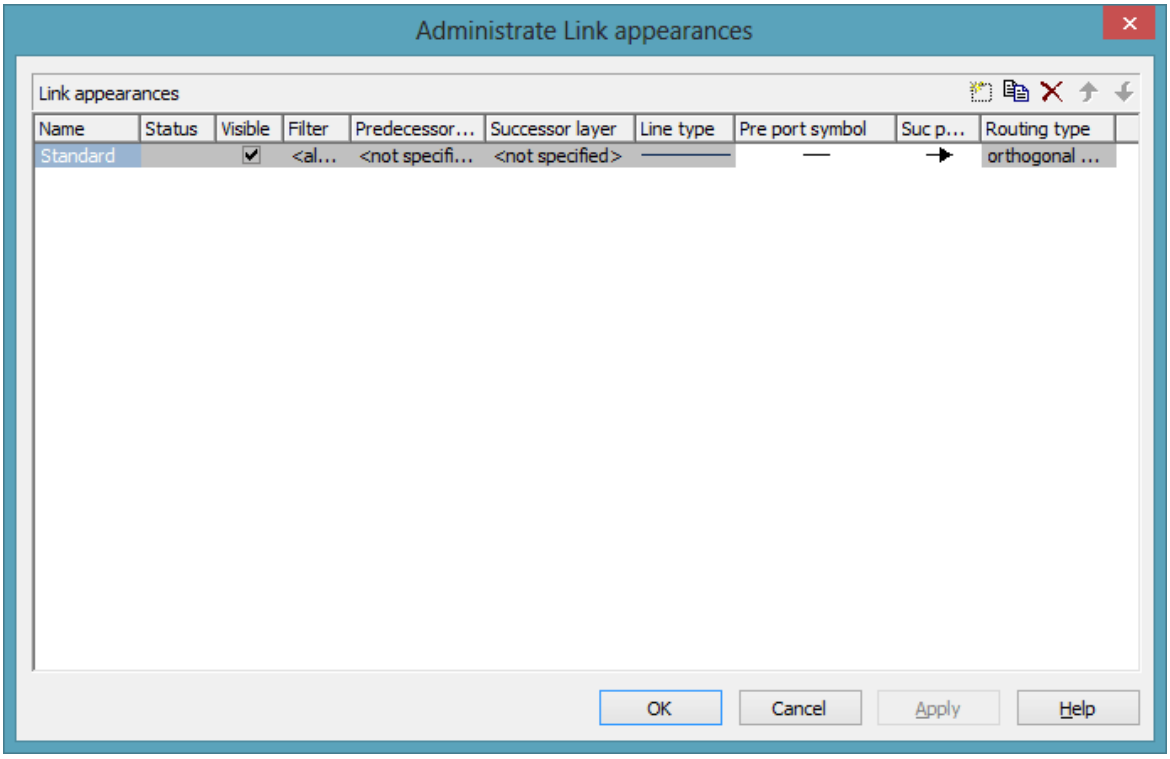
Applies the marked property to all fields.

Preview

The current fields of the box format are displayed in the preview window. If you click on a field, you can modify its attributes in the **Fields** table.

With the help of the buttons above the preview window you can add new fields or delete the marked field. You also can use the Del button to delete fields.

4.30 The "Administrate Link Appearances" Dialog Box



You can get to this dialog by clicking the **Link appearances** button on the **Objects** property page.

Name

This column displays the names of the link apperances available. The names can be edited.

This feature can also be set by the property **VcLinkAppearance.Name**.

Status

In the **Status** column each link appearance that has been added ( ) and/or modified ( ) since the dialog box was opened is marked by a symbol.

Visible

This check box lets you specify whether the links between the nodes should be displayed. This feature can be also set by the property **VcLink-Appearance.Visible**.

Filter

This column displays the filter used for a link appearance. From the select box you can select an appropriate filter.

This feature can also be set by the property **VcLinkAppearance.Filter-Name**.

Predecessor layer

Specify to which layer of the predecessor node the link is to be drawn. If the selected layer is not assigned to a node, the link will be drawn to the first visible layer of this node.

This feature can also be set by the property **VcLink Appearance.-PredecessorLayerName**.

Successor layer

Specify to which layer of the successor node the link is to be drawn. If the layer selected is not assigned to a node, the link will be drawn to the first visible layer of this node.

This feature can also be set by the property **VcLink Appearance.Successor-LayerName**.

Line type

Clicking on an entry in this column will cause an **Edit** button to occur, by which you can get to the **Line attributes** dialog box. There you can set type, thickness and color of the line.

This feature can also be set by the property **VcLink Appearance.LineType**.

Pre port symbol

Select a port symbol for a link that visually accentuates the junction of the link and the predecessor node.

This feature can also be set by the property **VcLink Appearance.-PredecessorPortSymbol**.

Suc port symbol

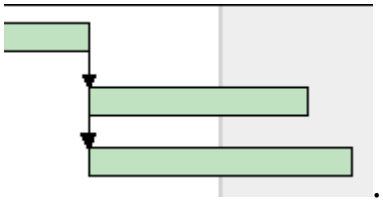
Select a port symbol for a link that visually accentuates the junction of the link and the successor node.

This feature can also be set by the property **VcLink Appearance.Successor-PortSymbol**.

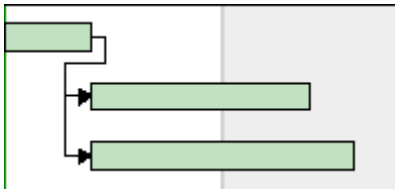
Routing type

This field allows to select a routing type. As the first row of the table containing the link appearance types is reserved for the default link appearance, the item <not specified> is selectable only from the second row on. If <not specified> has been selected, a routing type is used which is further up the list of the LinkAppearance objects.

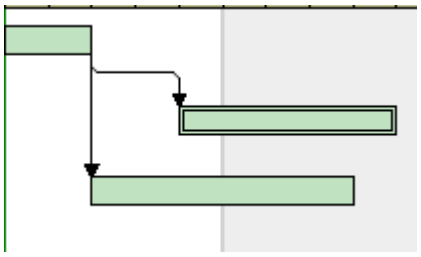
The routing type can also be set by the **VcLinkAppearance** property **RoutingType**.



Straight-lined link type



Orthogonal link type



Orthogonal distinguishable link type

Add link appearance

 A new link appearance will be created. You can modify its default name by double-clicking and editing it.

Copy link appearance

Copies the selected link appearance.

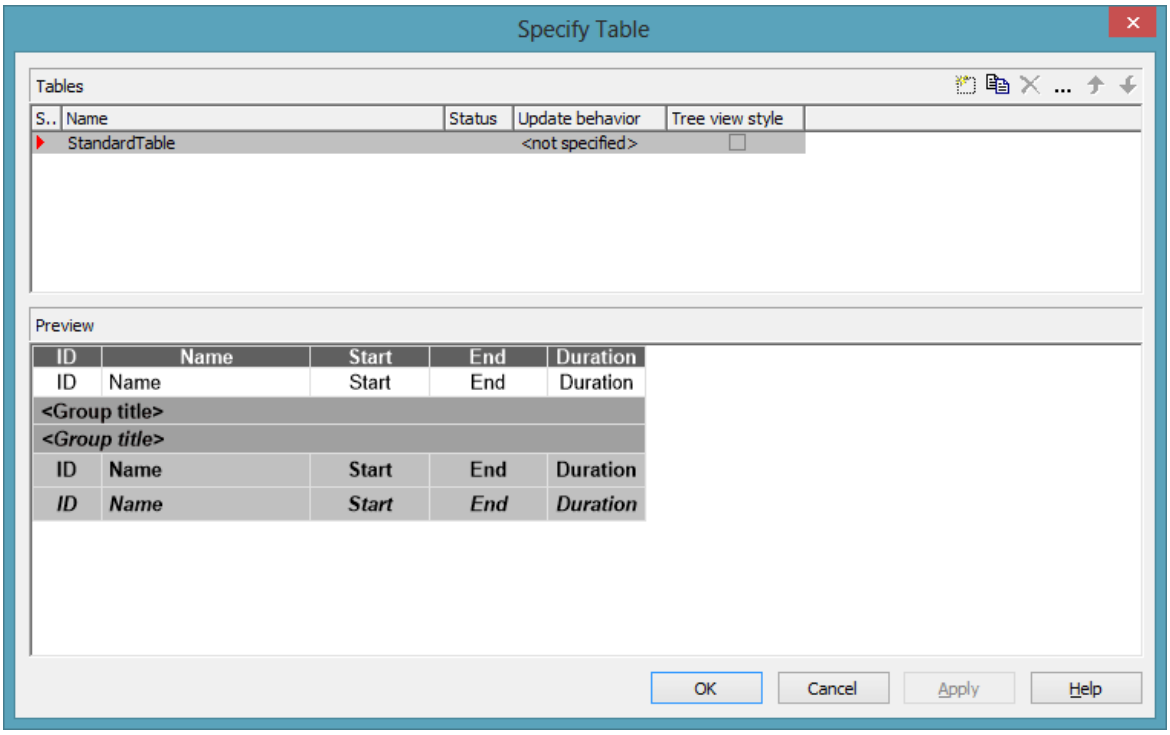
Delete link appearance

The marked link appearance in the list will be deleted. You can only delete link appearances that are not currently used.

Promote / demote link appearance

By these buttons you can move the line format by one position up or down in the list.

4.31 The "Specify Table" Dialog Box



In this dialog box you can establish and administer tables.

Preview

The table marked by a small red arrow in the **Preview** column is displayed in the preview window in the lower half of the dialog above. It simultaneously is the table presently edited.

Name

Lists the names of all tables that are defined. The names can be edited.

Status

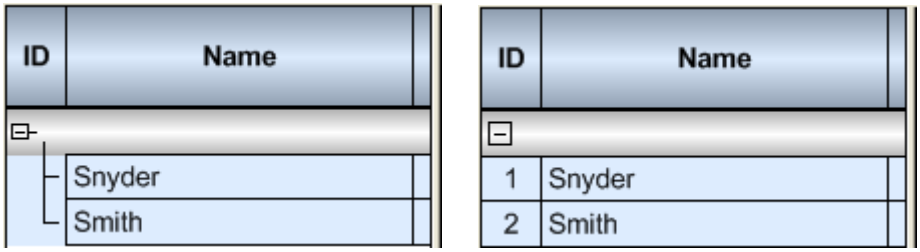
In this column each table that has been added ( ) and/or modified ( ) since the dialog box was opened is marked by a symbol.

Update behavior

Select an update behavior for this table. Leaving the setting to <not selected> means that the setting for tables made in the **Edit Update behavior** dialog will apply

Tree view style

If this check box is activated, nodes will be arranged in tree view style, with lines tracing the logical tree structure. In either case, plus or minus symbols mark levels.

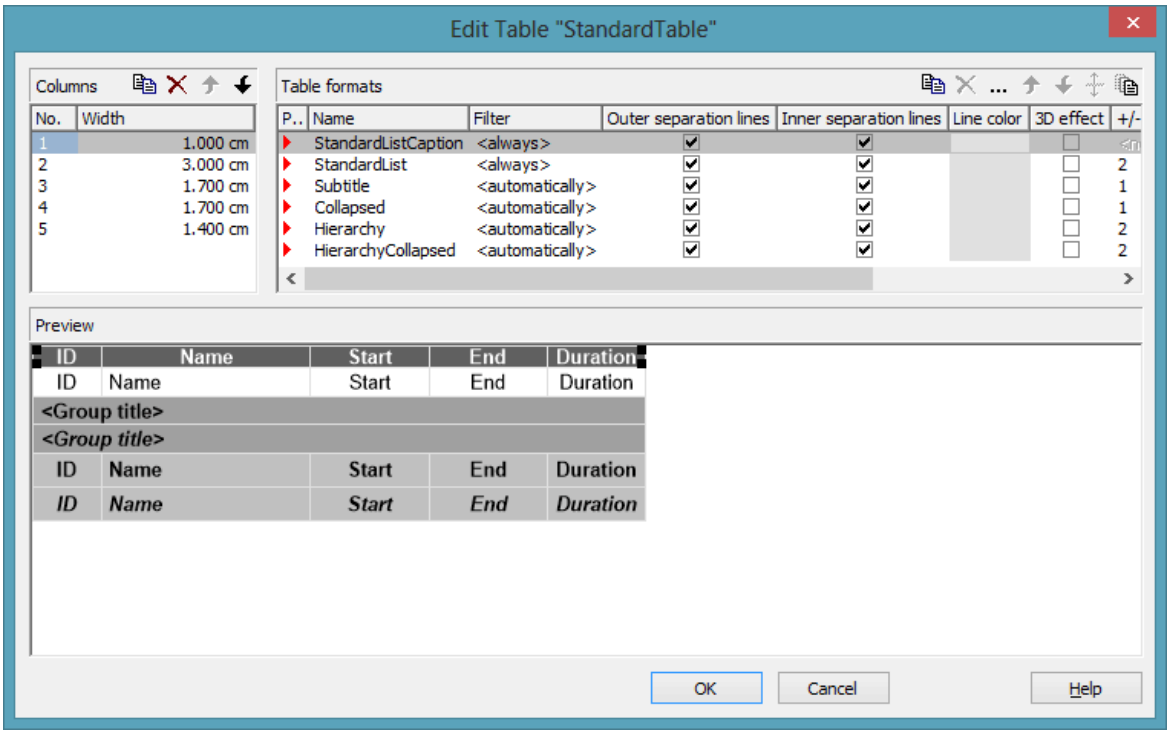


Pictures above: a group with and without the tree view style set

Add / copy / delete / edit / prmote / demote table

 By these buttons you can create, copy or delete the marked table or move it by one position up or down in the list, respectively. The latter may serve to sort the names and thus contribute to improved clarity but has no function in terms of priority.

4.32 The "Edit Table" Dialog Box



In this dialog box you can edit a table.

Columns

The **Columns** list contains the **No.** and the **Width** of each table column. The width can be varied by steps of 1 mm in the range from 0 to 10 cm.

You can define 100 columns at maximum. The sequence of the table columns in the **Columns** list corresponds to the sequence of the table columns in the chart.

 The buttons above the **Columns** list allow to copy or delete table columns or to modify their position in the list.

Table Formats

The **Table Formats** list lets you specify different table formats:

- **Preview:** A table format marked by a red arrow is displayed in the preview window.
- **Name:** A table format by default has a name. **StandardListCaption** is the name of the table format of the table caption. The names can be edited only for the table formats **ListFormat2**, **ListFormat3** and for all table formats that you have specified yourself.

- **Filter:** A table format is combined with a filter that selects the activities to which the table format is to apply. When several filters of this list apply to an activity, the table format of the highest priority will be used. The sequence of the filters in the list of the **Table formats** field of the dialog box inversely corresponds to their priority: the top filter has lowest priority. Four pre-defined filters exist. The format of the <interfaceNode> filter applies to nodes interfacing the nodes selected. The <never> filter never applies. It practically serves as a template for copying. The <automatically> filter applies to nodes of the same group level; the level is to be specified. The <always> filter collects all nodes that were not selected by other filters. It makes sense to put it at the top; in addititon, it cannot be deleted.
- **Outer/Inner separation lines:** Specify whether the table fields are to be separated by lines outside and/or inside the table fields.
- **Line Color:** You can assign a line color to a format.
- **3D effect:** Specify whether the table fields are to be highlighted by a 3D effect.
- **+/- column:**Specify whether in a column + or - shall be displayed for collapsing or expanding subordinated lines. Select the appropriate column from the drop down list.
- **Indent column:**Specify the column to be indented. This only works if there are lines (nodes) subordinated to this line (node). Then the first aubordinated line will be indented. If the **automatically** filter is assigned, the column in which +/- is displayed will be indented.
- **Indent width:**Specify by how much (in mm) the column shall be indented.

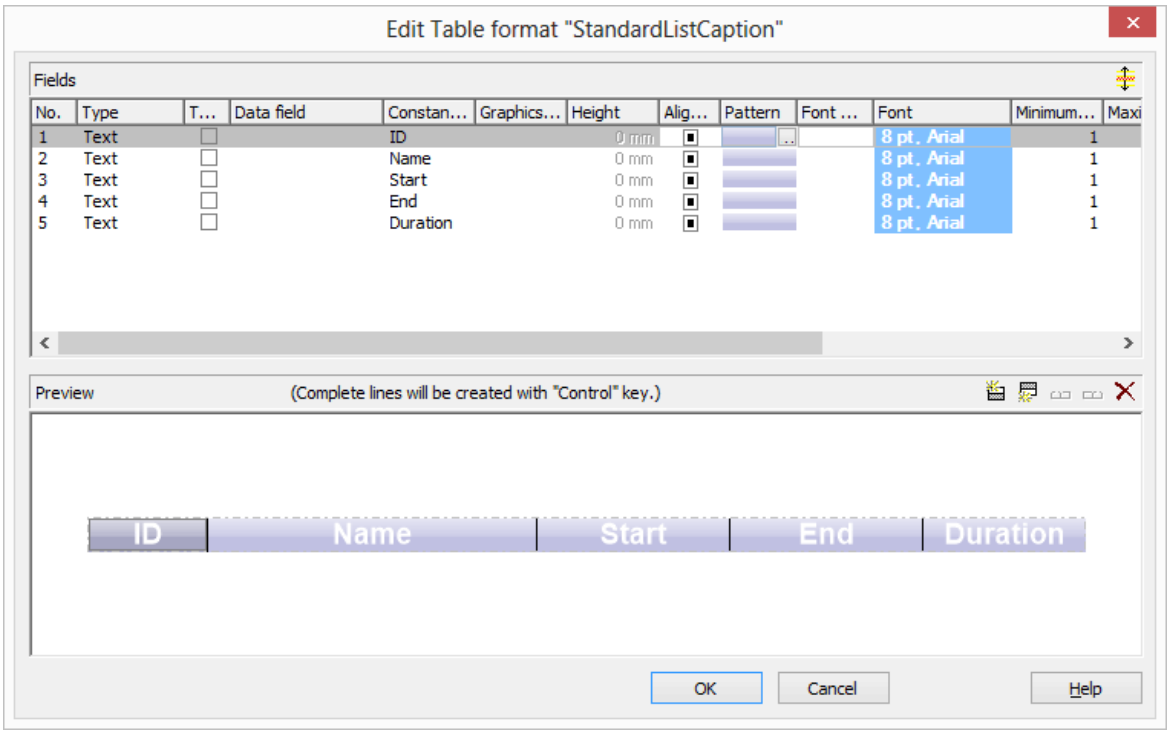
By using these buttons at the top of the **Table Formats** list you can copy or delete table formats or open the **Edit Table Format** dialog.

Note: For the table format **StandardListCaption** (table caption) attributes cannot be assigned by maps.

By using these buttons you can move the table formats in the list, except for the first and the second one that are immobile.

If you have changed the attributes **Outer separation lines** or **Inner separation lines** of a table format and then click on this button, the changed attribute will be applied to all table formats.

4.33 The "Edit Table Format" Dialog Box



In this dialog box you can edit a table format (row type).

No.

Number of the table format field: This number cannot be edited. It is used as the index that allows to call this table format field by the API.

If you create a new table format field in the preview window, it will be annotated by a "?" instead of a number. You can verify its correct number if you leave the dialog by clicking on **OK** and reopen it.

Type

Please select the field type: **text**, **graphics** or **multi-state**. Multi-state fields are used for example to trigger a rotating sequence of different states and of the associated data fields when clicked.

Combi field

If this check box is activated, in the table format field a text and a graphics can be combined as follows:

- **Type**: Text, **Combi field**: no: Only text will be displayed (as specified for **Data field** or for **Constant text**).

- **Type**: Graphics, **Combi field**: no: Only a graphics will be displayed (as specified for **Graphics file name**).
- **Type**: Text, **Combi field**: yes: Text (as specified for **Data field** or for **Constant text**) and a graphics (as specified for **Graphics file name**) will be displayed.
- **Type**: Graphics, **Combi field**: yes: Only a graphics will be displayed (as specified for **Graphics file name**). Text (as specified for **Data field**) is visible only in a tooltip. If possible, it will be displayed as hyperlink.

Data field

Select the data field whose content is to be displayed in the current field. Additionally to the data fields defined in the data definition table, you can select one of the following options:

- <Group title>: the code specified for the current grouping level
- <Row number>: consecutively numbered rows

If the content of a data field does not fit into the current field, the excess will be cropped in the diagram.

Constant Text

(only if no data field has been specified) Type a constant text to be displayed in the current field.

Graphics file name

Indicates the name and directory of the graphics file that will be displayed in the current table format field.

As soon as you click on a **Graphics file name** field, two buttons appear:

Click the first button to open the Windows dialog box **Choose Graphics File**. There you can select a graphics file to be displayed in the current table format field.

If a relative file name has been specified, at run time the file will be searched in the path set in the VARCHART ActiveX property **FilePath** first. If it won´t be found there, the file will be searched in the current directory of the application and in the installation directory of VARCHART ActiveX.

Click this button, if you want to use a map to display graphics in table format fields in dependence on the node data. Then the **Configure Mapping**

dialog box will open which lets you configure a mapping from data field entries to graphics files.

If in the **Configure Mapping** dialog box only a data field, but no map is selected, the content of the data field will be used as graphics file name. If in the data field or in the map no valid graphics file name is found, the file name specified in the **Symbol file field** will be used.

If a mapping has been configured, the arrow on the second button will be displayed in bold ( ).

As soon as you leave the **Symbol File Name** field, a symbol indicates that a mapping has been configured.

When the graphics is displayed, the color of the pixel in the upper left corner will be replaced by the color of the diagram background. That means that all pixels of the graphics that have this color will be displayed transparent.

Height

(only for the type graphics) Specify the minimum height for the selected field (in mm). The maximum height is 99 mm.

Alignment

Specify the alignment of the content of the selected field (9 possibilities).

Pattern

This field lets you set the default background pattern and colors of the table format. By clicking on ... you open the **Edit pattern attributes** dialog where you can specify a pattern, a background color and, if needed, a second pattern color by clicking on . You can define your own colors in addition to the ones suggested. Transparent colors are also available.

By clicking on you open the **Configure Mapping** dialog box. Here you can configure data-dependent patterns and colors. If a mapping has been configured, the arrow on the button will be displayed in bold ( ).

Font Color

Indicates the font color for the current field. If you click on the field, two buttons will appear:

by the arrow button you can open the Color picker to select a font color.

by the second button you reach the **Configure Mapping** dialog box. Here you can configure data-dependent font colors. If a mapping has been configured, the arrow on the button will be displayed in bold ( ).

Font

Indicates the font style for the current field. If you click on the field, two buttons will appear:

... The Windows **Font** dialog box will appear.

by the second button you reach the **Configure Mapping** dialog box. Here you can configure data-dependent fonts. If a mapping has been configured, the arrow on the button will be displayed in bold ( ).

Minimum/Maximum line count

(only for the type text) Specify the minimum/maximum number of lines of text that can be displayed in the current field. Each field can contain a maximum of nine lines of text.

Spacing

Specify the spacing in percent.

Wrapping

Specify the wrapping of rows.

Hor. Margins (left/right)/ Ver. margins (top/bottom)

Specify the margins of the table format fields.

+/- column

Specify whether + or - for collapsing or showing further lines shall be displayed.

Indent column

Specify whether the column shall be indented.

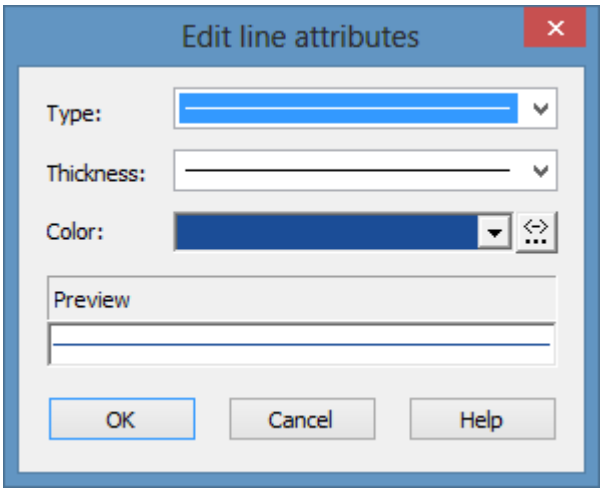
Preview

The current fields of the table format are displayed in the preview window. If you click on a field, you can modify its attributes in the **Fields** table.

 With the help of the buttons above the preview window you can add new fields or delete the marked field. You also can use the Del button to delete fields.

The first four buttons (for adding new fields) are only activated, if it is actually possible to create a new field beside the field marked. This depends on the number of columns of the current table format specified in the **Edit Table** dialog.

4.34 The "Edit Line Attributes" Dialog Box



This dialog which can in each case be invoked by clicking on ··· is available for hierarchy and grouping, for calendar grids, for the bar appearance, for filling of curves and the numeric scales in a histogram, for the link appearance, for intervals and for box frames.

Type

Select the line type (dashed, dotted etc.).

Thickness

Define the line thickness.

Color

Select the line color.

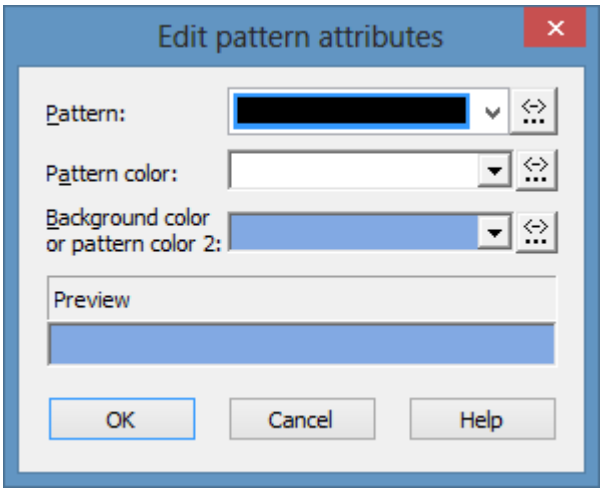
This button will open the **Configure Mapping** dialog box where you can specify the line color data-dependent.

After having mapped the line color, the arrow on the button will appear bold.

Preview

The line appearance based on the current settings is displayed in this field.

4.35 The "Edit Pattern Attributes" Dialog Box



The pattern dialog which can be invoked by clicking on ··· is available for filling of curves in a histogram, calendar grids, group title, intervals, time scale sections, box, line and table formats, layers and for node lines.

This button will open the **Configure Mapping** dialog box where you can specify the pattern, pattern color, background color or background color 2 data-dependent.

After having mapped one/several pattern attributes, the arrow on the button will appear bold.

Pattern

Here you can select a fill pattern.

Pattern color

Select the foreground color of the fill pattern.

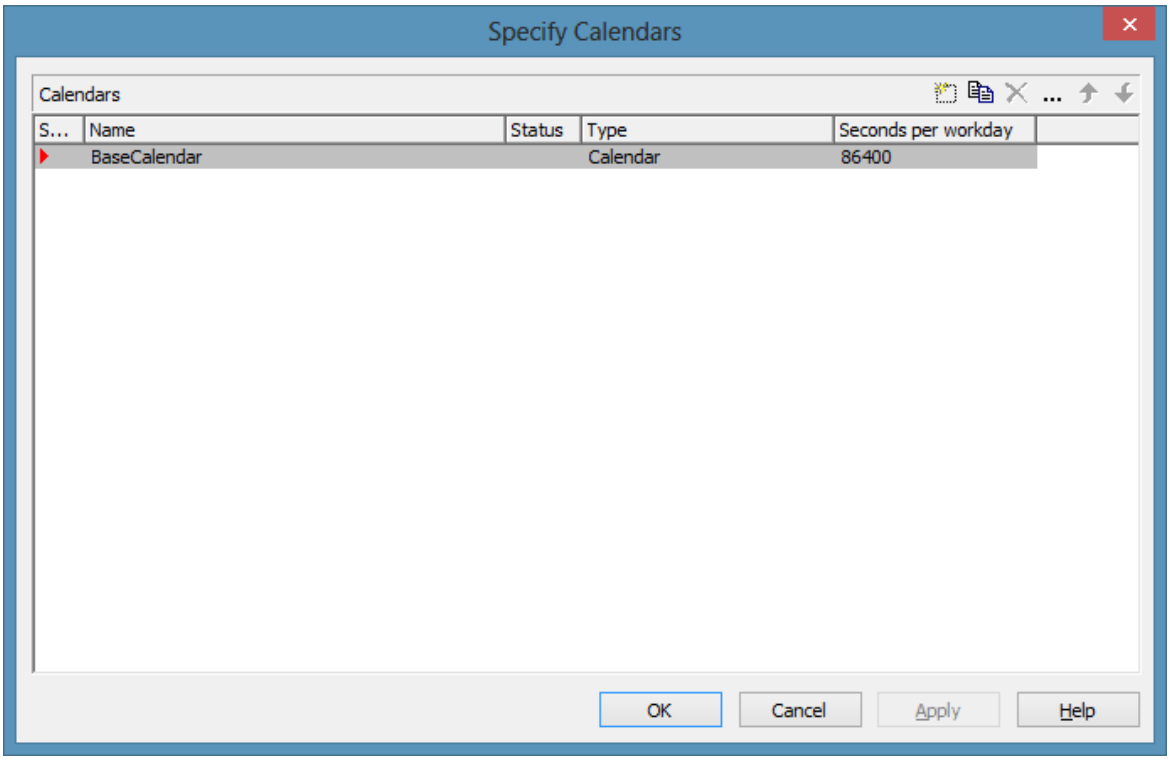
Background color or pattern color 2

Select the background color or a second pattern color.

Preview

The pattern based on the current settings is displayed in this field.

4.36 The "Specify Calendars" Dialog Box



You can get to this dialog box by the **Objects** property page. You can define a separate calendar for each line of the table.

Selected

The calendar marked by a small arrowhead in the **Selected** column is used for the calendar grid.

Name

Lists the names of all calendars defined.

Status

In the **Status** column each calendar that has been added ( ) and/or modified ( ) since the dialog box was opened is marked by a symbol.

Type

Specify the calendar type. Besides ordinary calendars shifts calendars are available, too.

Seconds per Workday

Specify how much seconds the workday has got.

Add calendar

Click on this button to add a calendar.

Copy calendar

The marked calendar is copied.

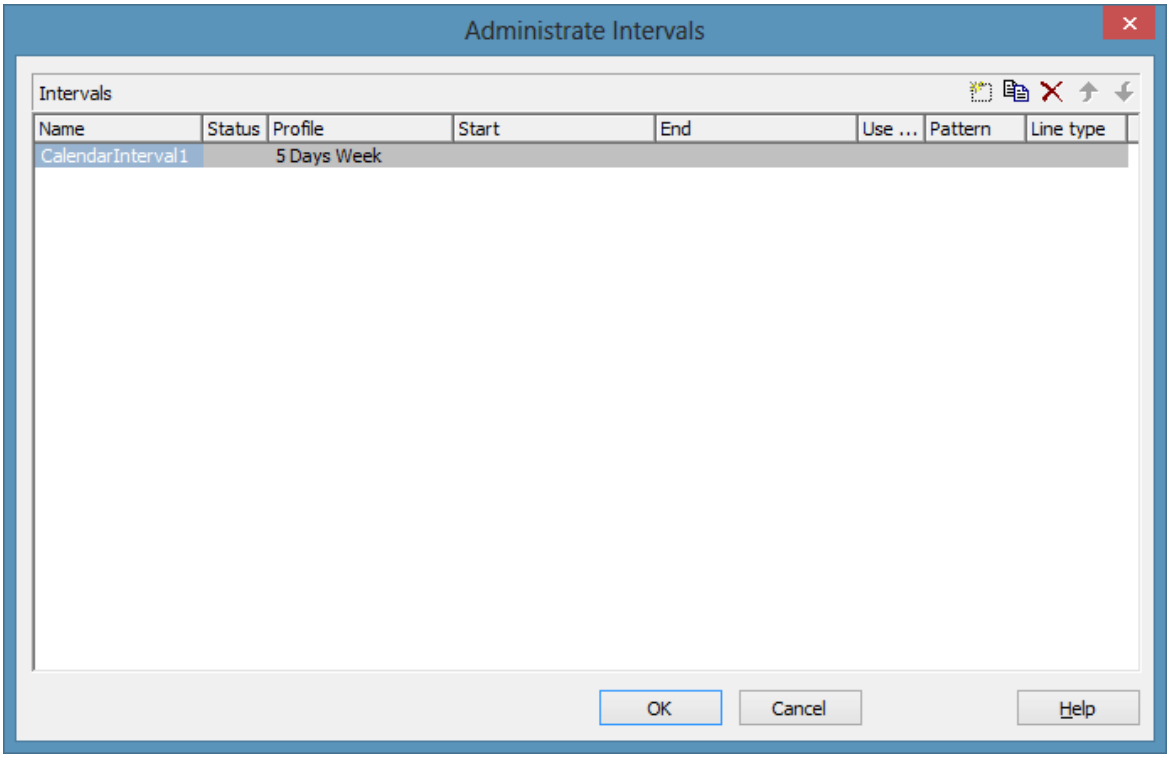
Delete calendar

The marked calendar is deleted.

Edit calendar

You will reach the **Edit Calendar** dialog box.

4.37 The "Administrate Intervals" Dialog Box (Calendar)



In this dialog box you can create and modify intervals.

Name

Lists the names of all intervals. All names can be edited.

Status

In this column each interval that has been added ( ) and/or modified ( ) since the dialog box was opened is marked by a symbol.

Profile

Here you can select a profile for your interval by clicking . If you want to edit the profile click on ··· beside its name to open the **Administrate Calendar profiles** dialog.

Start/End

In this field you can set the beginning or end of of an interval. The date can be easily entered or modified by using the spin control.

Use graphical attributes

If this option is selected, you can select an display a pattern and a line type for the interval. The option is only active for the profil types <Working time> and <Nonworking time>.

Pattern

Click on ··· to open the dialog **Edit pattern attributes**.

Line type

Click on ··· to open the dialog **Edit line attributes**.

Add interval

A new interval will be created. You can modify the marked name by double-clicking and editing it.

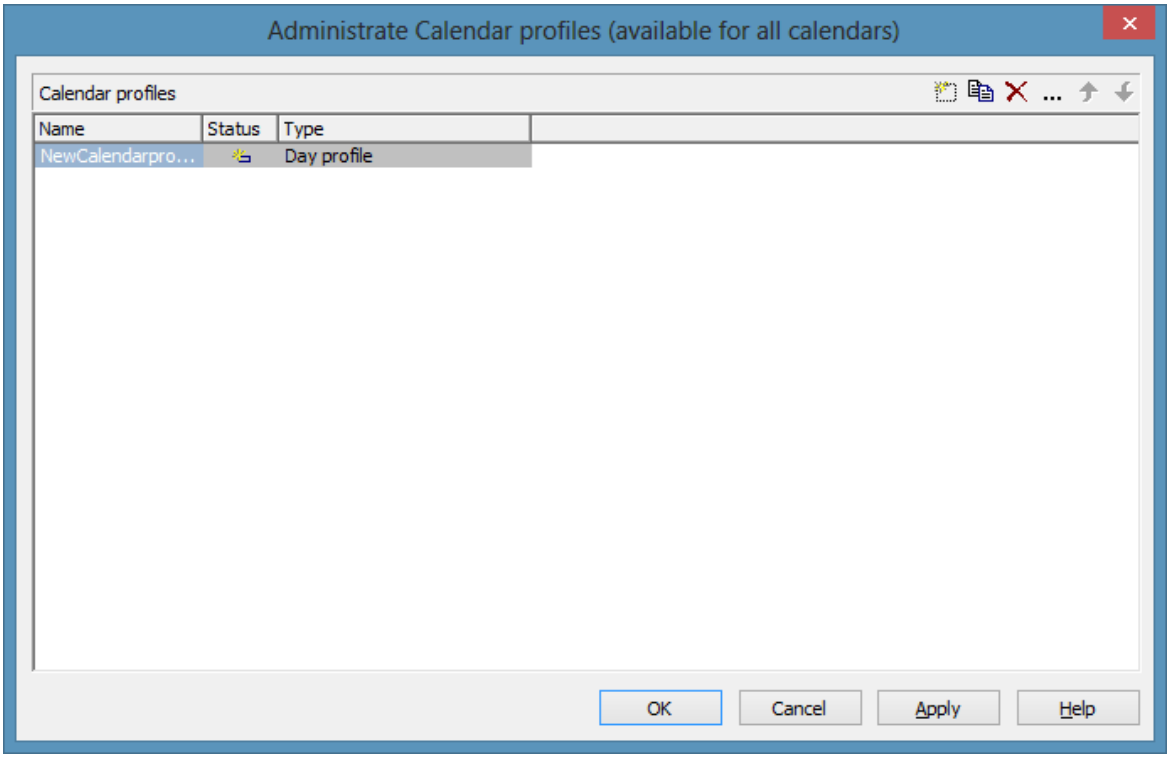
Copy interval

Click on this button to copy the marked interval.

Delete interval

Click on this button to delete the marked interval.

4.38 The "Administrate Calendar Profiles" Dialog Box



In this dialog you can create and modify calendar profiles.

Name

Lists the names of all calendar profiles. All names can be edited.

Status

In this column each calendar profile that has been added ( ) and/or modified ( ) since the dialog box was opened is marked by a symbol.

Type

By clicking you can select the calendar profile type. You can choose between <Day profile>, <Week profile>, <Year profile> and <Variable profile>.

Add calendar profile

A new calendar profile will be created. You can modify the marked name by double-clicking and editing it.

Copy calendar profile

Click on this button to copy the marked calendar profile.

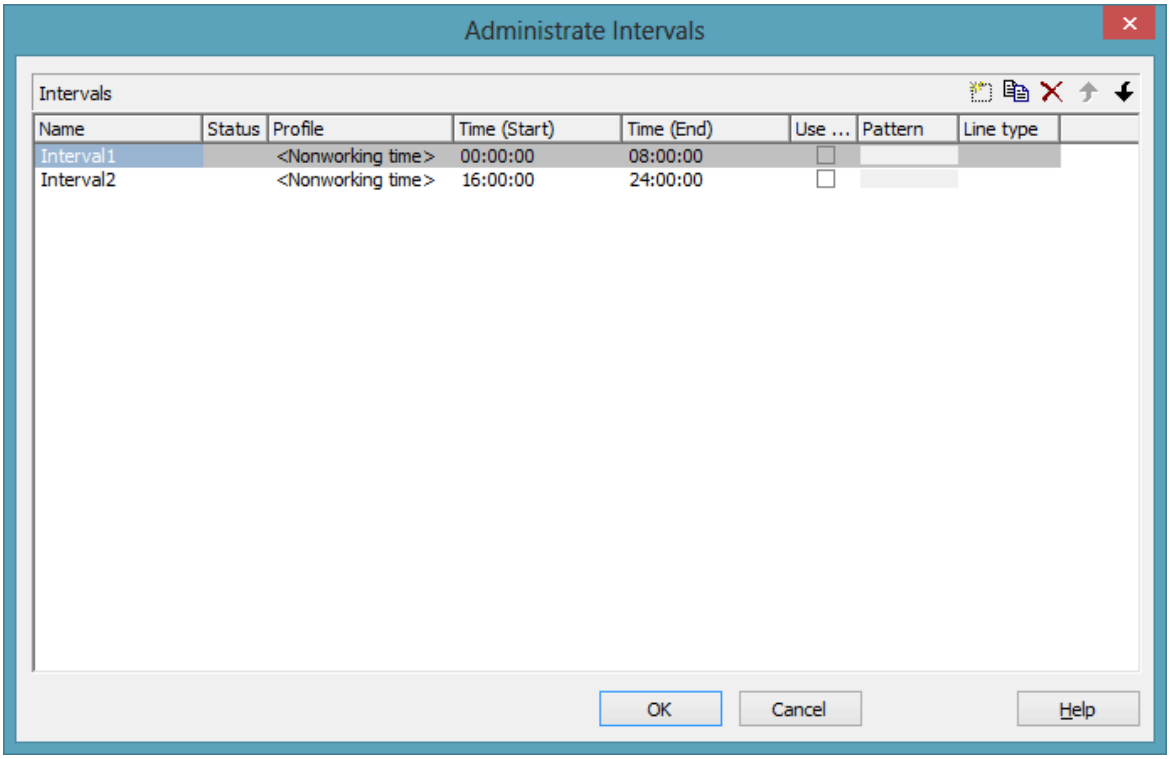
Delete calendar profile

Click on this button to delete the calendar profile.

Edit calendar profile

... You will reach the **Administrate Intervals** (Calendar profiles) dialog box.

4.39 The "Administrate Intervals" Dialog Box (Calendar Profiles, Profile Type <Day Profile>)



You can get to this dialog if you activate the dialog box "Admininstrate Calendar Profiles" on the "Objects" property page, and then click on the "Edit" button of the calendar profile. The different types of profiles offer different setting options. This dialog serves to create and modify intervals of a day profile.

Name

Lists the names of all intervals. All names can be edited.

Status

In this column each interval that has been added ( ) and/or modified ( ) since the dialog box was opened is marked by a symbol.

Profile

Here you can select a profile for your interval by clicking .

Time Start/Time End

In this field you can set the start or end time of an interval by clicking on the arrow buttons.

Use graphical attributes

If this option is selected, you can select an display a pattern and a line type for the interval. The option is only active for the profil types <Working time> and <Nonworking time>.

Pattern

Click on ... to open the dialog **Edit pattern attributes**.

Line type

Click on ... to open the dialog **Edit line attributes**.

Add interval

A new interval will be created. You can modify the marked name by double-clicking and editing it.

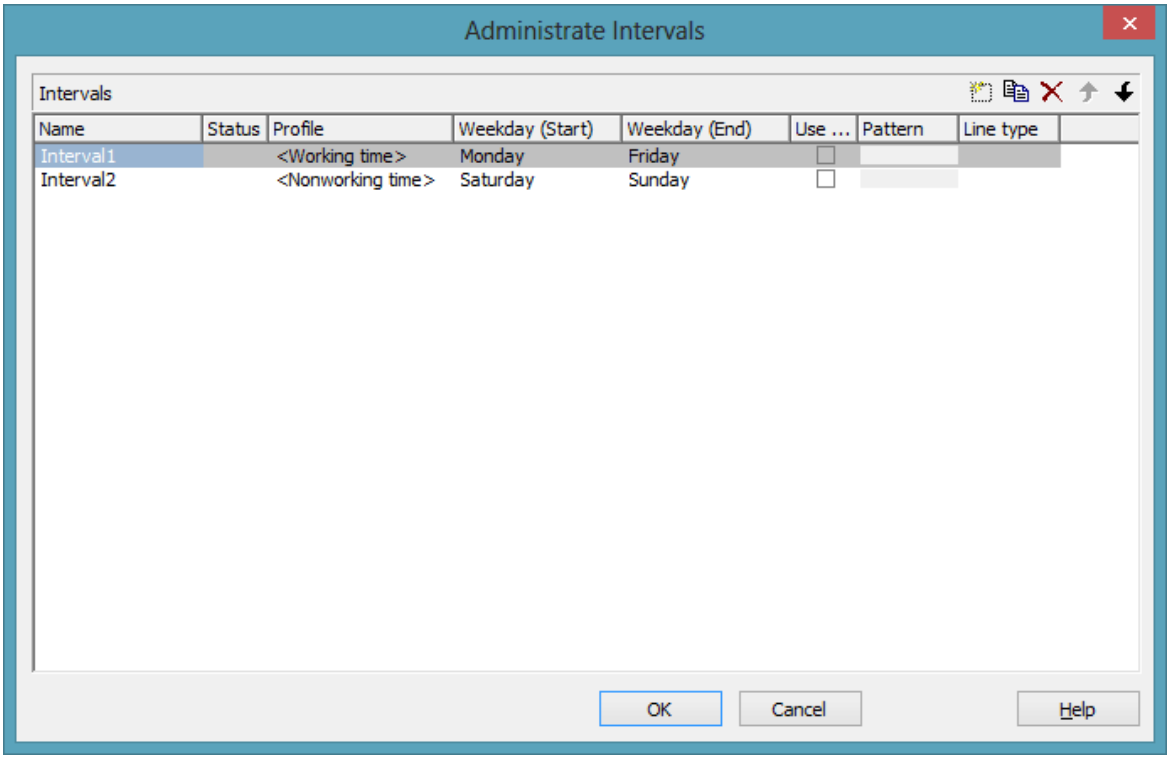
Copy interval

Click on this button to copy the marked interval.

Delete interval

Click on this button to delete the marked interval.

4.40 The "Administrate Intervals" Dialog Box (Calendar Profiles, Profile Type <Week Profile>)

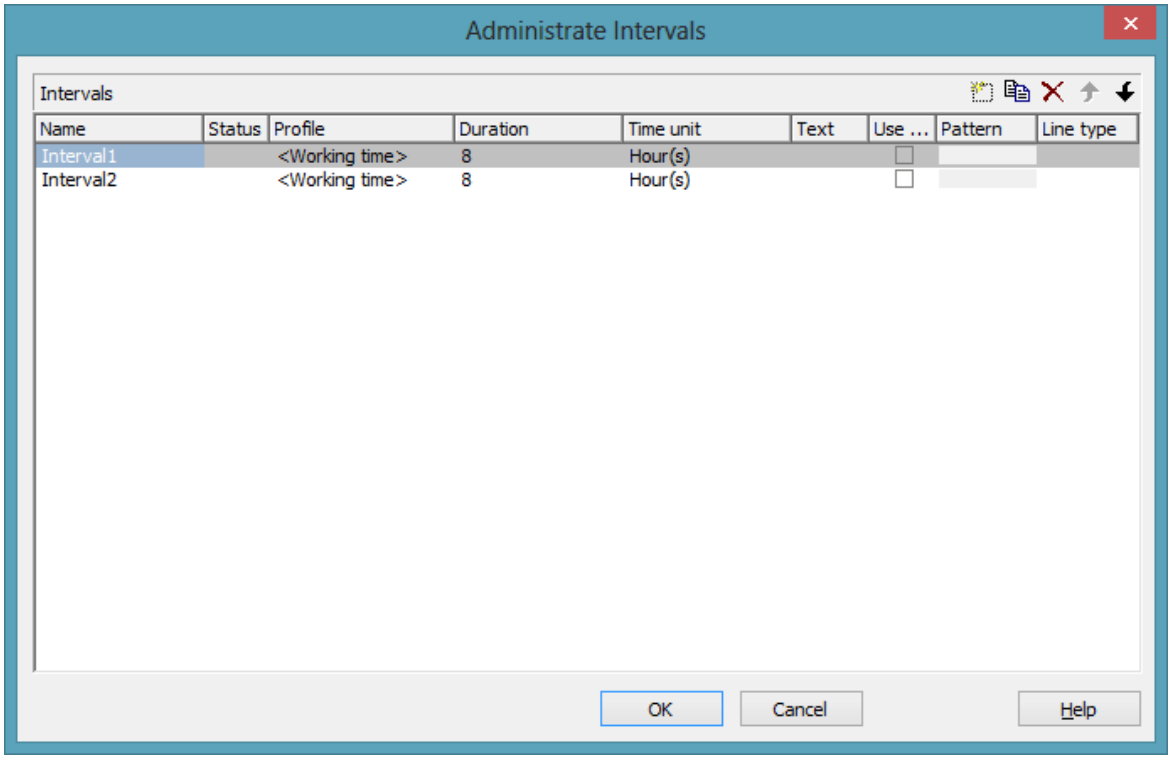


You can get to this dialog if you activate the dialog box "Admininstrate Calendar Profiles" on the "Objects" property page, and then click on the "Edit" button of the calendar profile. The different types of profiles offer different setting options. This dialog serves to create and modify intervals of a week profile.

Weekday Start/Weekday End

By clicking you can set the first/last weekday of the interval.

4.41 The "Administrate Intervals" Dialog Box (Calendar Profiles, Profile Type <Variable Profile>)



You can get to this dialog if you activate the dialog box "Admininstrate Calendar Profiles" on the "Objects" property page, and then click on the "Edit" button of the calendar profile. The different types of profiles offer different setting options. This dialog serves to create and modify intervals of a variable profile.

Duration

Here you can specify the duration of the interval. This feature can also be set by the property **VcInterval.Duration**

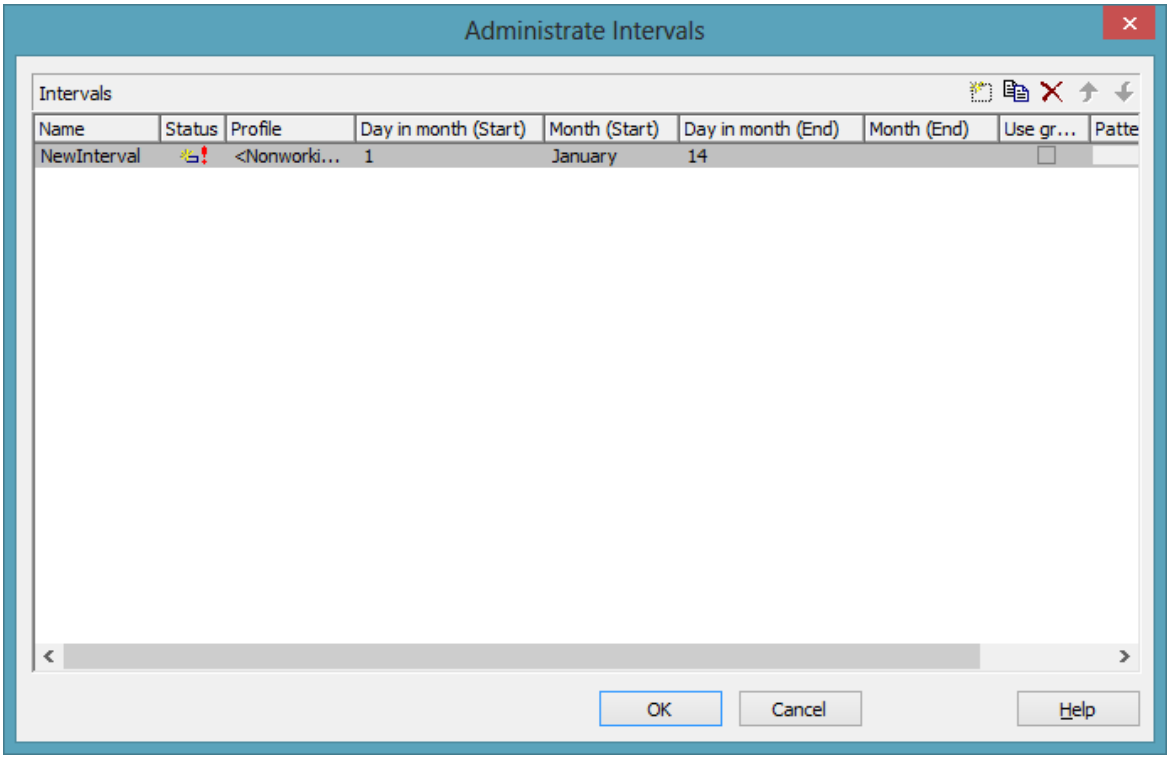
Time unit

Here you can specify the time unit of the interval. This feature can also be set by the property **VcInterval.TimeUnit**

Text

Here you can specify the text of the time ribbon This feature can also be set by the property **VcInterval.Text**

4.42 The "Administrate Intervals" Dialog Box (Calendar Profiles, Profile Type <Year Profile>)



You can get to this dialog if you activate the dialog box "Admininstrate Calendar Profiles" on the "Objects" property page, and then click on the "Edit" button of the calendar profile. The different types of profiles offer different setting options. This dialog serves to create and modify intervals of a year profile.

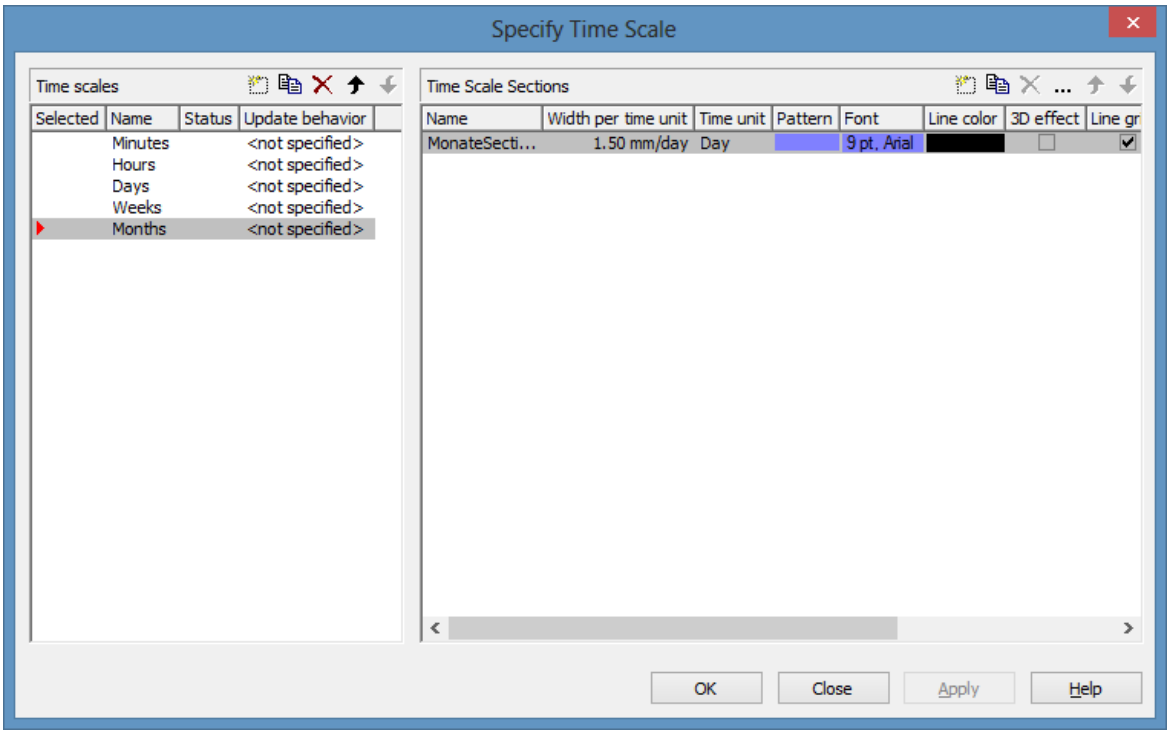
Day in month (Start)/Day in month (End)

By clicking you can set the day in the start/end month of the interval. This feature can also be set by the property **VcInterval.DayInStart/EndMonth**

Month (Start)/Month (End)

By clicking you can set the day in the start/end month of the interval. This feature can also be set by the property **VcInterval.Start/EndMonth**

4.43 The "Specify Time Scale" Dialog Box



This dialog box you can get to by the **Objects** property page. You can create different time scales and define sections to them.

Time scales

- **Selected:** The time scale marked by a small arrowhead in this column is used for the diagram. Please note that the time scale selected here should match the **Time unit** selected on the **General** property page.
- **Update behavior** Select an update behavior for this time scale. Leaving the setting to <not selected> means that the setting for time scales made in the **Edit Update behavior** dialog will apply
- **Name:** Lists the names of all time scales that are defined. The names can be edited.
- **Status:** In this column each time scale that has been added ( ) and/or modified ( ) since the dialog box was opened is marked by a symbol.

Add / copy / delete / edit time scale; up / down

By these buttons you can create, copy or delete time scales or move them by one postion up or down in the table, respectively.

Time Scale Sections

The **Sections** table contains all sections specified for the selected time scale. The following properties can be specified:

- **Name** of the section
- **Width per Unit:** Specify the unit width of the active time scale. The basic unit is the smallest unit into which the time scale is divided. You can specify the basic unit width in millimetres in steps of 100th of a millimetre. The maximum width you can assign to the basic unit is 320 mm, the minimum width is 0.01 mm.
- **Unit** of the section: seconds, minutes, hours, days.
- **Pattern:** Click on ··· to open the **Edit pattern attributes** dialog where you can specify another pattern for the section. If the ribbons had different patterns before, the new pattern will be applied to all sections.
- **Font:** Select the font for the annotation in the section. When you click the first button (▼), the Color Picker box will appear where you can choose the font color. When you click the second button (···), the Windows **Font** dialog box will appear where you can choose the font type. If the ribbons had different fonts (colors or types), the font selected here will be applied to all sections.
- **3D-Effect:** This box lets you decide whether the time scale should be assigned a 3D effect (to give it perspective).
- **Line grids:** Specify whether predefined vertical grid lines should be displayed in the diagram area beneath the current section or not.
- **Calendar grids:** Specify whether a predefined calendar grid should be displayed in the diagram area beneath the current section. If you choose to display a calendar grid, weekends and other workfree periods, for example, will be highlighted by vertical areas.
- **Collapse Workfree Periods:** If you select this option, workfree periods will not be displayed in this section. The calendar that defines the workfree periods is selected in the **Specify Calendars** dialog box.

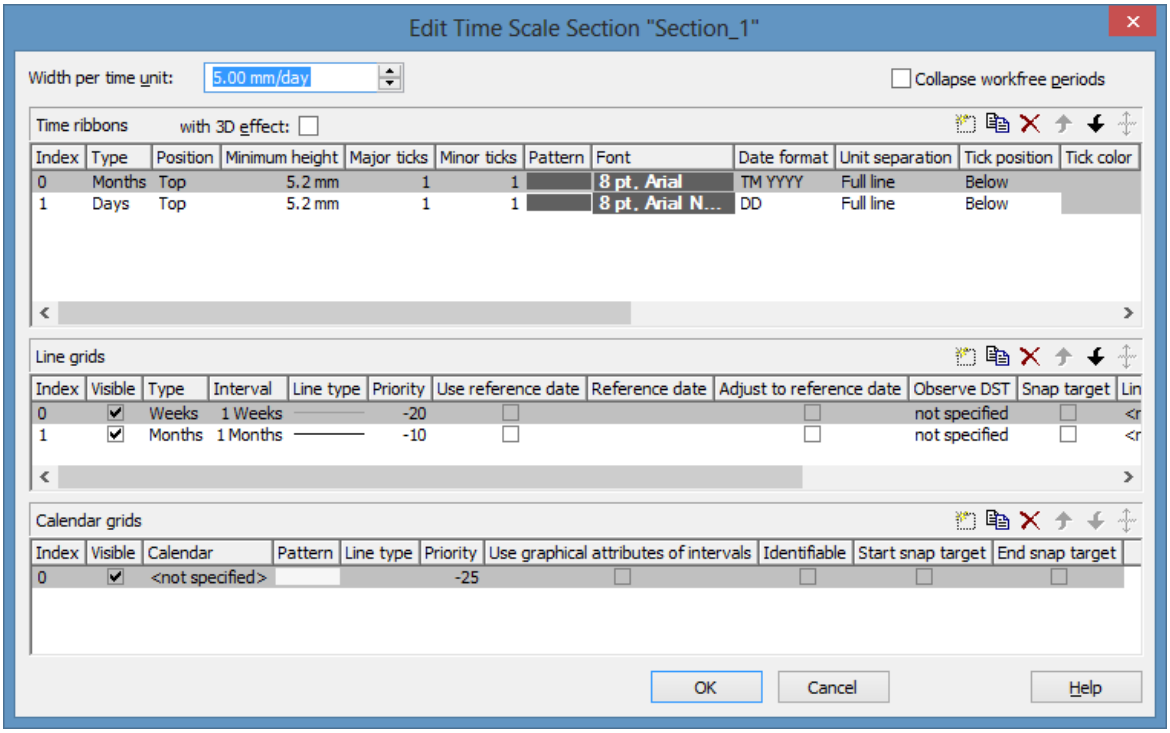
Add/ Copy/ Delete/ Edit/ Promote/Demote time scale section


by these buttons you can create, copy, delete or edit time scales or move them in the table respectively.

The position of the time scale sections in the table corresponds with their position in the diagram.

If you have specified a new section, all sections will be displayed with nearly the same extension. You can change the extension of each section by dragging the mouse. by the API the start of each section can be edited.

4.44 The "Edit Time Scale Section" Dialog Box



This dialog box lets you create and modify time scale sections.

Width per unit

This field lets you edit the width of a unit in the selected section. The new value will be copied to the corresponding field in the **Time Scale Sections** table in the **Specify Time Scale** dialog.

Collapse workfree periods

Here you can specify, whether or not workfree periods should be displayed in the section. The calendar that defines the workfree periods is the one selected in the **Specify Calendars** dialog box.

The setting of this field will be copied to the corresponding field in the **Time Scale Sections** table in the **Specify Time Scale** dialog.

Ribbons

Ribbons serve the purpose of annotating the time scale. A section may have several ribbons (e.g. one showing a monthly and a second one showing a daily scale).

By these buttons you can create, copy and delete ribbons and move them in the table.

The table lets you modify the settings of the ribbons in the selected section:

- **Index**: Displays the serial number of a ribbon (cannot be edited).
- **Type** Lets you set the type of ribbon: seconds, minutes, hours, days, weeks, months, quarters, years, shifts, fiscal quarters, fiscal years.
- **Position:** Lets you specify, whether the ribbon should be displayed at all and if so, whether its position should be at the top or at the bottom of the diagram.
- **Minimum height** Allows to set the minimum height of the ribbon (in mm).
- **Major ticks:** You can set after how many time units a major tick should be displayed, for example after 7 days. (The time unit depends on the ribbon type selected.) The major ticks will be annotated, if sufficient space is available.
- **Minor ticks:** Allows to set after how many time units a minor tick (not annotated) should be displayed, e.g. after one day. The time unit depends on the ribbon type selected.
- **Pattern:** Shows the pattern of the ribbon. Click on ··· to open the **Edit pattern attributes** dialog where you can select a pattern, a color and a second pattern color. If you don't select a new pattern, all ribbons of the time scale section have the pattern specified in the **Specify Time Scale** dialog. If you assign a new pattern to the first ribbon of a section it will be copied to the **Pattern** field in the **Time Scale Sections** table in the **Specify Time Scale** dialog.
- **Font:** Lets you set font specifications to the annotation of the ribbons. If this value is not set, the ribbons of the section will display the font set in the **Specify Time Scale** dialog. To assign a different font color to a ribbon, please click on the drop-down-button ( ) in the ribbon field to get to the color picker. To assign a different font type to a ribbon, please click on the **edit** button (···) of the ribbon field to get to the Windows **Font** dialog box. The font that you define for the first ribbon of a section will be copied to the **Font** field of the **Sections** table in the **Specify Time Scale** dialog.
- **Date format:** Lets you set the date format to the ribbon. The available formats depend on the selected type of ribbon. To compose the date you can use the following tokens:

D: first letter of the day of the week (not adjustable)

TD: Day of the Week (adjustable by using the event **OnSupplyTextEntry**)

DD: two-digit figure for the day of the month: 01-31

DDD: first three letters of the day of the week (not adjustable)

M: first letter of the name of the month (not adjustable)

TM: name of the month (adjustable by using the event **OnSupplyTextEntry**)

MM: two-digit figure for the month: 01-12

MMM: first three letters of the name of the month (not adjustable)

YY: two-digit figure for the year

YYYY: four-digit figure for the year

WW: two-digit figure for the number of the calendar week: 01-53

TW: text for "calendar week" (adjustable by using the event **OnSupplyTextEntry**)

Q: one-digit figure for the quarter: 1-4

TQ: name of quarter (adjustable by using the event **OnSupplyTextEntry**)

hh: two-digit figure for the hour in 24 hours format: 00-23

HH: two-digit figure for the hour in 12 hours format: 01-12

Th: Text of "o' clock" (adjustable by using the event **OnSupplyTextEntry**)

TH: "am" or "pm" (adjustable by using the event **OnSupplyTextEntry**)

mm two-digit figure for the minute: 00-59

ss: two-digit figure for the second: 00-59

TS: short date format, as defined in the regional settings of the windows control panel

TL: long date format, as defined in the regional settings of the windows control panel

TT: time format, as defined in the regional settings of the windows control panel

xC/XC: You can set a maximum ten-place, simple upward counting from a reference date onward, for example "15:05:07:16:00", which equals

15 months, 5 days, 7 hours, 16 minutes, 0 seconds. The notation is: **xC44:C33:C22:C11:C00**. In written language: Show at least 2 digits for the counters 4...0 and a preceding "-" symbol if the value is negative. The separators are variable and can be replaced by other separators symbols. "x" means: Display a preceding "-" symbol if the value is negative, but no "+" symbol if it is positive. "X" means: Display a preceding "-" symbol if the value is negative and a "+" symbol for positive values. In the dialog **Edit Time Scale Section...** the check boxes **Use reference date** and **Adjust major ticks to reference date** need to be ticked, also, the parameter **Serial annotation** has to be set to **No**. In the application the reference date is set at run time by the call **VcRibbon.set ReferenceDate**, overriding any settings in the dialog.

Note: Characters which are not to be interpreted as part of the date should be preceded by a backslash '\'. '\\' for instance results in '\'. The special characters: **':, /, -**' and **blank** don't need '\' as prefix.

- **Unit separation:** You can choose between three options for the separating lines in the ribbon: **straight lines**, **ticks** and **no lines**.
- **Tick position:** Decide whether the ticks and their annotations should be displayed at the top or at the bottom of the ribbon.
- **Tick color:**You can select the color of ticks.
- **Alignment:** You can choose between **centered**, **right**, **left** and **at ticks** for the alignment of the ribbon annotation.
- **Serial annotation:** Lets you specify whether serial numbers are to be displayed in the ribbon instead of dates, and if so, whether null should be the origin at the reference date possibly set.
- **Use reference date:** Activate this check box if the start value of the serial annotation (or of the fiscal year or quarter) should coincide with the reference date selected. Otherwise it will be placed onto the beginning of the section.
- **Reference date:** Select the reference date from the date picker.
- **Adjust to Reference date:**Tick this check box to position the line grid on a different value of the time unit, i.e. the one defined by the reference date, for example on 13:17 of a day.
- If this option is not selected, the lines of a line grid are positioned on the beginning of a time unit, for example on 00:00 h of a day.

- **Calendar:** If you want to display a shift ribbon, select one of the shift calendars created in the **Specify Calendars** dialog box.
- **Observe DST:** Tick this box if daylight saving time is to be considered for this ribbon.

Line grid

In the diagram area and in the histogram, one or more line grids (consisting of vertical lines) can be displayed below the selected section of the time scale.

By these buttons you can create, copy and delete line grids and move them in the table.

The table lets you modify the settings of the line grids in the selected section:

- **Index**: Displays the serial number of a line grid (cannot be edited).
- **Visible**: Activate this check box for the line grids to be displayed.
- **Type:** Lets you set the basic unit of the line grid, e.g. days, weeks, etc.
- **Interval:** Lets you set the size of the interval between the grid lines as an integer multiple of the basic unit of the grid.
- **Line type:** When clicking on the button in this field, the **Line attributes of line grid** dialog box will appear, where you can set shape and color of the borderlines of the line grid.
- **Priority:** Lets you set the priority of a line grid. It refers to other line grids and to layers (> 0: in front of the layers, < 0: behind the layers).
- **Reference Date:**The reference date shifts the beginning of the line grid away from the default start on Monday 0:00 h by the offset specified.
- **Observe DST:** Tick this check box if daylight saving time is to be considered for this line grid.
- **Snap target**: The line grid defines its relevant positions as "snap targets" for nodes/layers to be moved.

Calendar grid

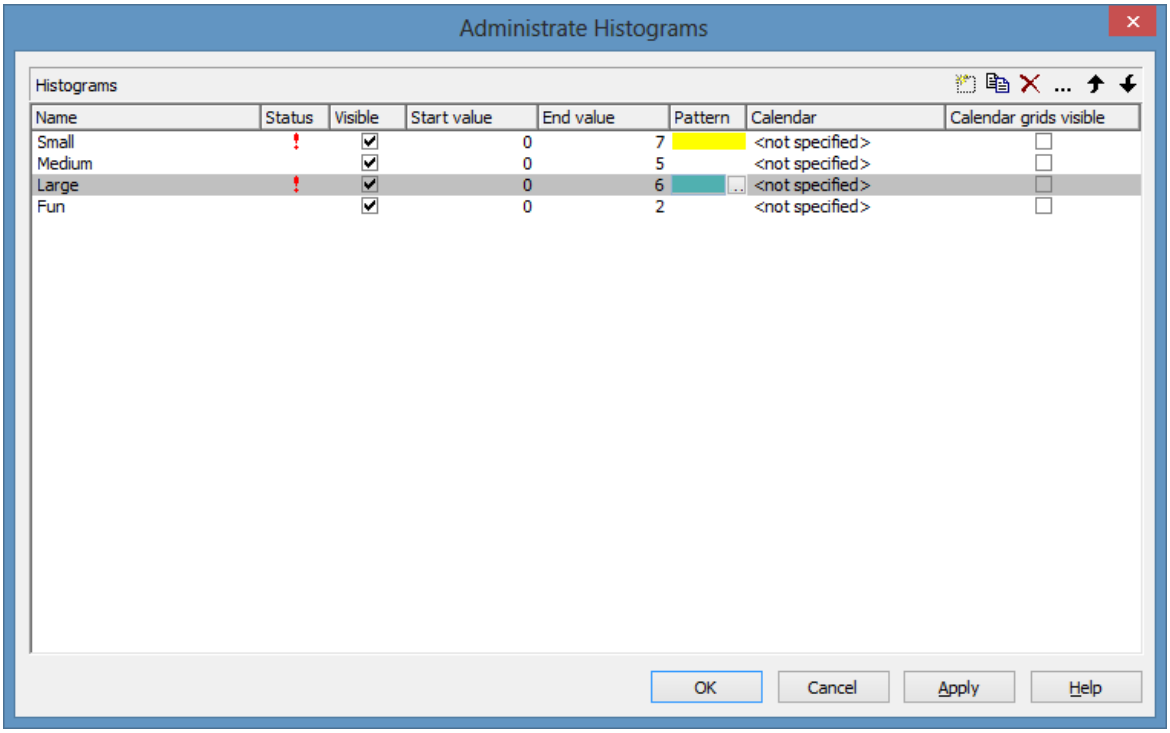
Calendar grids can be displayed in the diagram area and in the histogram of this section. If you choose to display a calendar grid, workfree periods will be highlighted by vertical areas.

By these buttons you can create, copy and delete calendar grids and move them in the table.

The table lets you modify the settings of the calendar grids:

- **Index**: Displays the serial number of a calendar grid (cannot be edited).
- **Visible**: Activate this check box for the calendar grids to be displayed.
- **Calendar**: Select the calendar that specifies the workfree periods displayed by the calendar grid. If you select the entry <not specified>, the calendar selected in the **Specify Calendars** dialog box will be used.
- **Pattern:** When clicking on this button (···), the **Pattern attributes** dialog box will appear, where you can set the type, the foreground and the background color of the pattern for the calendar grid. There are also transparent colors available.
- **Line type:** When clicking on this button (···), the **Line attributes of calendar grid** dialog box will appear, where you can enter the settings of the border lines of the calendar grid.
- **Priority:** Lets you set the priority of a calendar grid. It refers to other calendar grids and to layers (> 0: in front of the layers, < 0: behind the layers).
- **Calendar grid:** The calendar grid defines its relevant positions as "snap targets" for nodes/layers to be moved

4.45 The "Administrate Histograms" Dialog Box



You can get to this dialog by the **Layout** property page. You can create and modify one or more histograms and select which one(s) is/are to be displayed.

Preview

The preview window shows the histogram marked in the **Preview** column.

Name

Lists the names of all histograms that are defined. The names can be edited.

Status

In the **Status** column each histogram that has been added ( ) and/or modified ( ) since the dialog box was opened is marked by a symbol.

Visible

Tick this box if you want the selectes histogram to be displayed.

Start value

Specify the smallest value of the numeric scale of the histogram. If necessary, this value will be adapted to the curve values.

End value

Specify the greatest value of the numeric scale of the histogram. If necessary, this value will be adapted to the curve values.

Pattern

Specify pattern und color for the histogram.

Add histogram

A new histogram is created.

Copy histogram

Copies the selected histogram.

Delete histogram

The marked histogram is deleted.

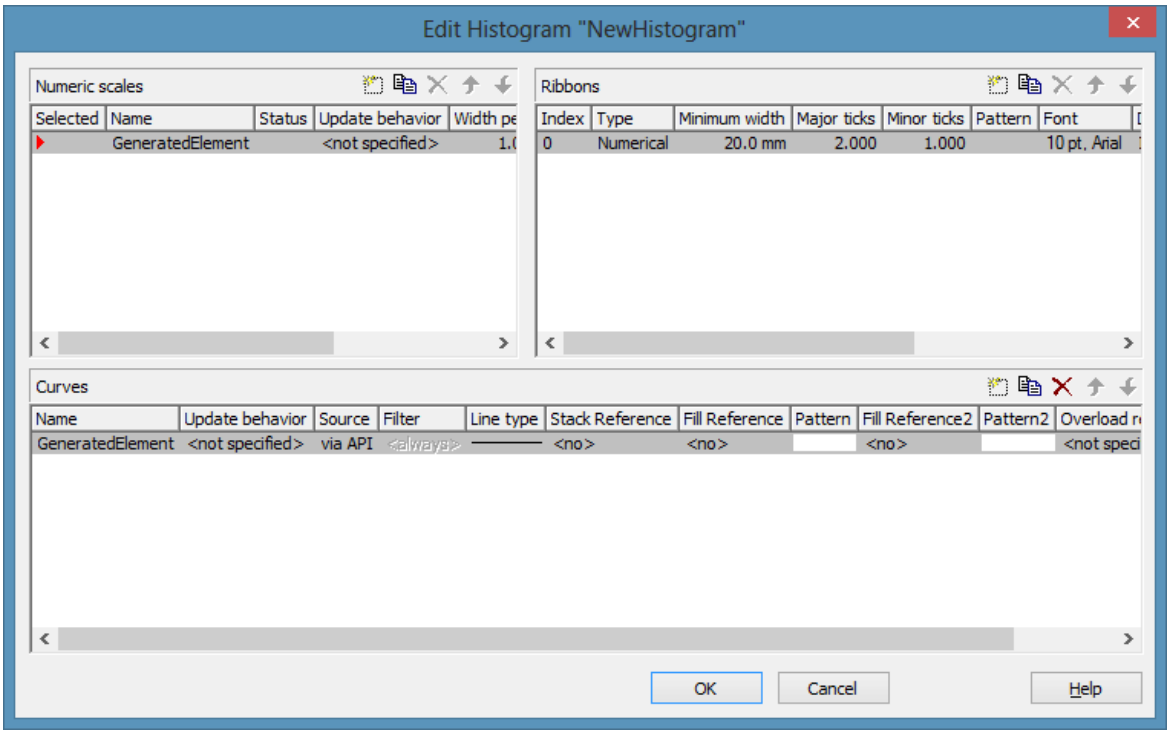
Edit histogram

The **Edit Histogram** dialog box will appear.

Promote / demote histogram

By these buttons you can move the histogram by one position up or down in the list. The order of the histograms in the list equals their order of output.

4.46 The "Edit Histogram" Dialog Box



This dialog box will appear if in the **Administer Histograms** dialog box the **Edit histogram** button (···) is clicked.

For the histogram being edited you can establish several numeric scales that contain one or more ribbon(s), and select the numeric scale to be displayed.

The histogram may contain several curves.

For each curve you can individually define the source by which its data are to be supplied. by filters you can select specific activities to compose the curve. Beside, you can define the appearance of the curves.

Numeric Scales

- **Selected:** The red arrow indicates which one of the numeric scales is displayed.
- **Name:** of the numeric scale
- **Status:** In this column each numeric scale that was added ( ) and/or modified ( ) after the dialog box was opened is marked by a symbol.
- **Update behavior** Select an update behavior for this numeric scale. Leaving the setting to <not selected> means that the numeric scale setting made in the **Edit Update behavior** dialog will apply
- **Width per Unit** in mm, specifies the space between the major ticks

- **Unit specifies the increment of the major ticks**
- **Line color Specify the tick color for all numeric ribbons**
- **Line Grids:** Specify whether a line grid is to be displayed.
- **Line type:** The line type of the line grid is displayed here. To change it, click on the button (···) in the field. Then the **Line Attributes** dialog box will open.

Ribbons

For each ribbon of the marked numeric scale you can set the below properties:

- **Index**: consecutive number of the ribbon (cannot be edited)
- **Type** of the ribbon (numerical or textual). By the button you open a dialog to specify the type.
- **Minimal width** in mm
- **Major ticks:** Enter the number of units after which a major tick including an annotation is to occur.
- **Minor ticks:** Enter the number of units after which a minor tick (smaller tick without annotation) is to occur.
- **Pattern:** By clicking on ··· you open the **Edit pattern attributes** dialog where you can specify a pattern, a pattern color and background color or, if needed, a second pattern color. You can define your own colors in addition to the ones suggested. Transparent colors are also available.
- **Font:** The font style and color of the ribbon are indicated. Click on the button (···) to get to the Windows **Font** dialog box.
- **Double format:** Here you can choose from a list of possible double output formats. **I** represents the figures before the decimal separator and D represents the figures after the decimal separator.
- **Tick color:** Specify the tick color for all numeric ribbons.
- **Object draw events:** Tick this option if you want to enable the events **OnObjectDrawEx** and <**OnObjectDrawCompleteEx**. The event **On-ObjectDrawEx** lets you replace the default annotation ribbon by a customer-defined one, and with the event **OnObjectDrawCompleteEx** you can add something to the annotation ribbon that was drawn by VARCHART XGantt,
- **Unit label:** annotation of the label units of the numeric scale.

Curves

- **Name:** In this column, the names of the curves available are listed.
- **Update behavior Select an update behavior for this curve. Leaving the setting to <not selected> means that the setting for curves made in the Edit Update behavior** dialog will apply
- **Source:** By defining the source, you can specify where the data for calculating a curve are to be taken from. You can choose between two basic alternatives:

  1. by Layer: The curves are generated from the data of layers of those activities, that fulfill the filter criteria. Filters allow to select even more particular activities after more detailed criteria.

  2. By the API: This option sets the values by the API. In the API, the values for a histogram curve can be freely defined by of the VcCurve method **SetValues**. A curve defined this way is independent of user interactions and therefore can be used, say, as a reference curve, to display the availability, for example.

  By the **Edit** button (...) you can open the **Select curve data source** dialog box.
- **Filter:** If desired, a filter for each curve can be set to select for the activities that compose the curve. By the **Edit** button (...) you can open the **Administrate Filters** dialog box.
- **Line type:** Click on the **Linetype** entry to open the **Line attributes** dialog box.
- **Stack Reference:** You can set a reference curve in the **Stack Reference** field on which you want the current curve to be stacked. If you do not want to stack a curve, select the entry <No>. If you do not stack curves, they they may overlap each other. To differentiate between overlapping curves, you should assign them different patterns.
- **Fill Reference:** This field allows you to specify how far down the fill pattern below each curve should reach. If you select <No> in the **Fill Reference** field for a particular curve, there will be no fill pattern beneath this curve. If you enter <Flatline>, the fill pattern will reach down to the flatline. By specifying another curve in the **Fill Reference** field, the fill pattern will be displayed down to this curve.
- **Pattern:** Specify the pattern below each curve. by the **Edit** button (...) youc can open the **Pattern** dialog box where you can specify the pattern.

- **Fill Reference 2:** Select the second reference curve. The filling below the second reference curve is displayed only if the y values of the current curve (the curve defined in this row) are higher than the y values of the second reference curve.
- **Pattern 2:** Specify the pattern and the color of the filling above the second reference curve.
- **Overload results calendar:**Select a calendar created by you for this purpose to store the intervalls that have been calculated by the overload dates. You could this calendar, for instance, for a calendar grid in a group.

In the Tutorial you can find examples for the usage of histograms in the chapters "Using histograms" and "Displaying Capacity Bottlenecks".

Add numeric scale/ribbon/curve

A new object is created.

Copy numeric scale/ribbon/curve

Copies the selected object.

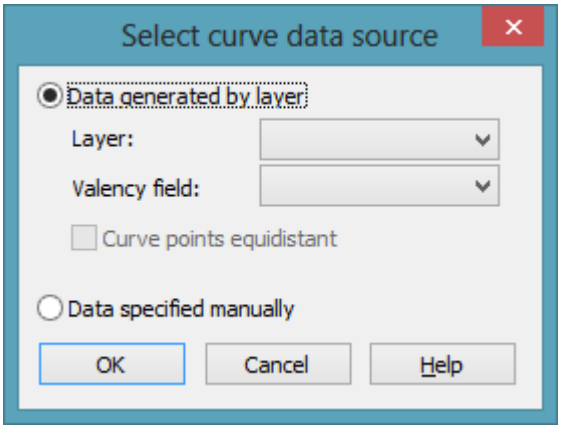
Delete numeric scale/ribbon/curve

The selected object is deleted.

Promote/demote numeric scale/ribbon/curve

By these buttons you can move the selected object by one position up or down in the list.

4.47 The "Select curve data source" Dialog Box



This dialog box you can get to by the **Edit Histogram** dialog.

Data generated by layer

Select this option, if you want the data to be generated by layer. When the activities are summarised to a curve, the start and end dates of the selected layer type (e.g. the "Start-End" layer) of each activity are adopted.

Then specify the following:

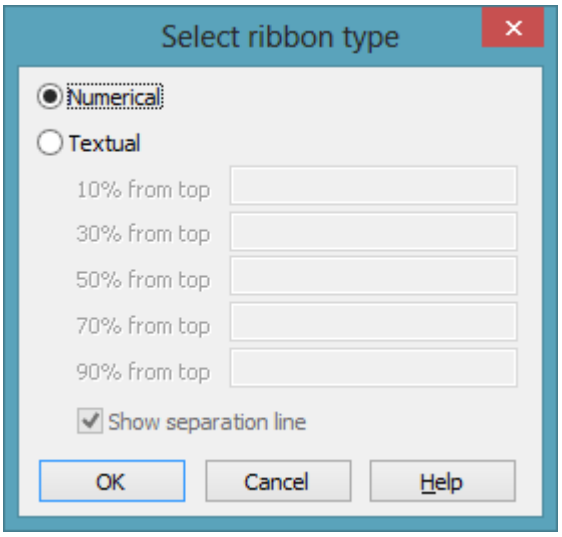
- **Layer**
- **Valency field:** data field from which for each activity the valency for the capacity sum is to be taken.

Data specified manually

Select this option, if the data are to be specified manually. For this option you may choose the option **Curve points equidistant**. Otherwise the curve points will be created only in those points where the y values are changing.

For further information please see the chapter "Important Terms: Histograms".

4.48 The "Select ribbon type" Dialog Box



This dialog box you can get to by the **Edit Histogram** dialog.

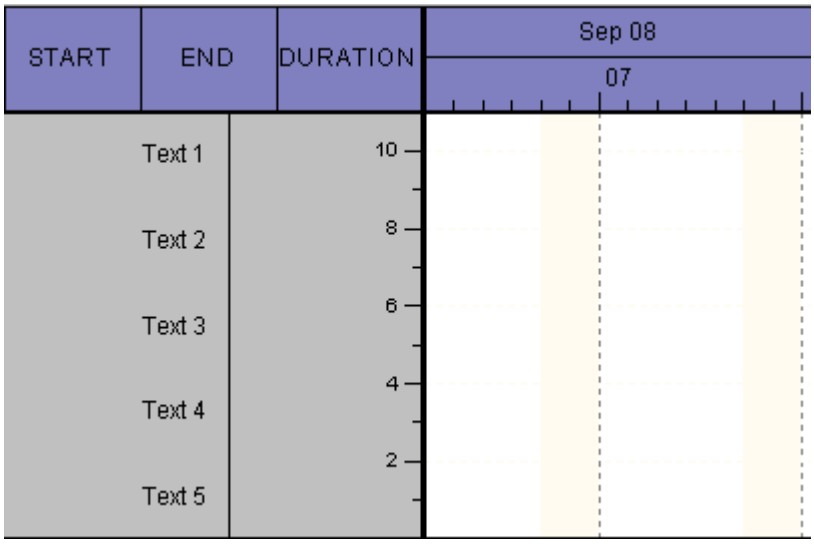
Numerical

Select this option if the current ribbon of the numeric scale is to be annotated with numbers.

Textual

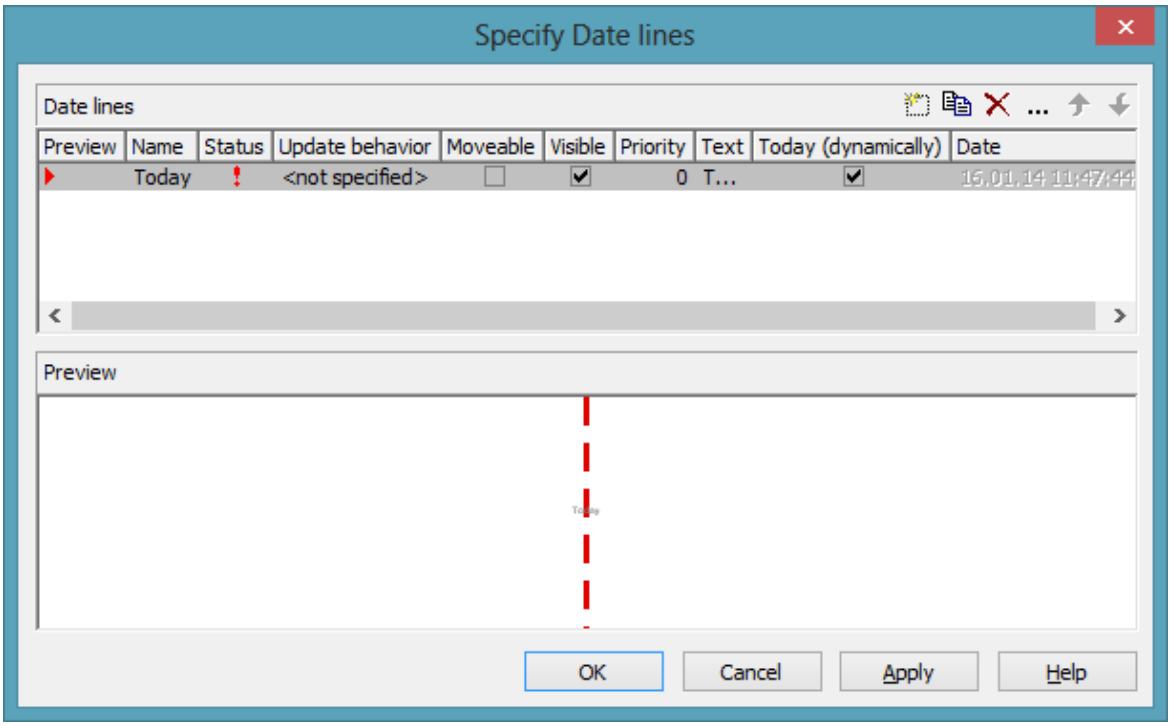
Select this option if the current ribbon of the numeric scale is to be annotated with texts which you can specify for five predefined positions (10%, 30%, 50%, 70 %, 90 % from top).

After having defined more than one ribbons in the dialog **Edit histograms** you can specify whether to draw a vertical separation line on the right of the corresponding ribbon by clicking **Separation line**.



Textual scale and numerical scale

4.49 The "Specify Date Lines" Dialog Box



You can get to this dialog box by the **Objects** property page.

Date lines (vertical lines in the diagram) let you highlight specific dates (the current date or any other date) in your diagram. The date lines that are displayed in the chart can be administered in this dialog box.

Preview

The date line marked by a small red arrowhead is displayed in the preview window.

Name

Lists the names of all date lines that are displayed in the chart. The names can be edited.

Status

In this column date lines that were added ( ) or modified ( ) after the dialog box was was invoked are marked by a symbol.

Update behavior

Select an update behavior for this date line. Leaving the setting to <not selected> means that the setting for date lines made in the **Edit Update behavior** dialog will apply

Moveable

Activate this check box, if you want the date line to be interactively moveable at run time.

Visible

Activate this check box, if you the date line should be visible at runtime.

Priority

Specify the priority of the date line (> 0: on top of of layers, < 0: behind layers).

Text

You can enter a text to be displayed at the date line.

Today (dynamically)

Tick this check box, if on the start of the program the date line should indicate the system date and time. In this case, the **Date** field will be deactivated.

Date

You can modify the date of the date line by marking a section of the date and then selecting a new value by the arrow keys.

Alternatively, you can set the date by the date control. For this, please click on the arrow button (). The **date** dialog box will appear where the selected date is highlighted. If no date was selected, the current date is highlighted. Select a day from the month displayed. You can flip through the months by clicking on the arrow buttons at the top of the calendar. If you click on the name of a month, a select box will appear which lists the names of all months. If you click on the year, a set of arrow buttons will appear by which

you can move to the next or to the previous year. If you click on **Today**, the current date will be selected.

Date

Tick this check box if you want the date line to be identified by the VcGantt method **IdentifyObjectAt**.

This option can also be set by the **VcDateLine.Identifiable** property.

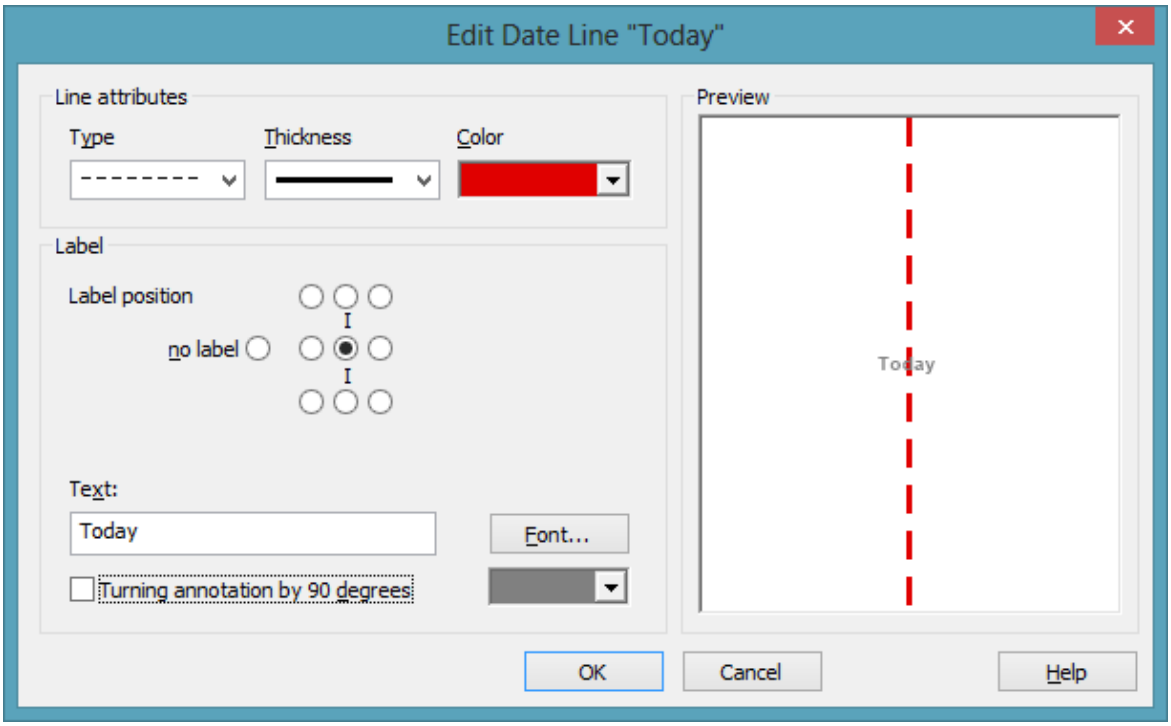
Snap target

Specify whether the date defines its position (hence its date) as "snap target" for nodes/layers to be moved.

Add / copy / delete / edit / promote / demote date line

 By these buttons you can create, copy or delete the date line or move it by one position up or down in the list, respectively.

4.50 The "Edit Date Line" Dialog Box



This dialog box lets you establish and modify time date lines.

Line attributes

Specify the **Type**, **Thickness** and **Color** of the date line.

Label position

Select the position at which a text should be displayed at the date line. If you do not want to display a text, tick the **no label** radio button. It is ticked by default, if no text is specified for the date line. If you specify a text for the date line and then leave the **Text** field, by default the text is displayed at the top right of the line. You can choose a different position for the text, if you want.

Text

Specify the text you want to display at the date line. By default the **Text** field is empty. When you select a text position at the date line the name of the line is transferred to the **Text** field. You can modify the text, if you wish.

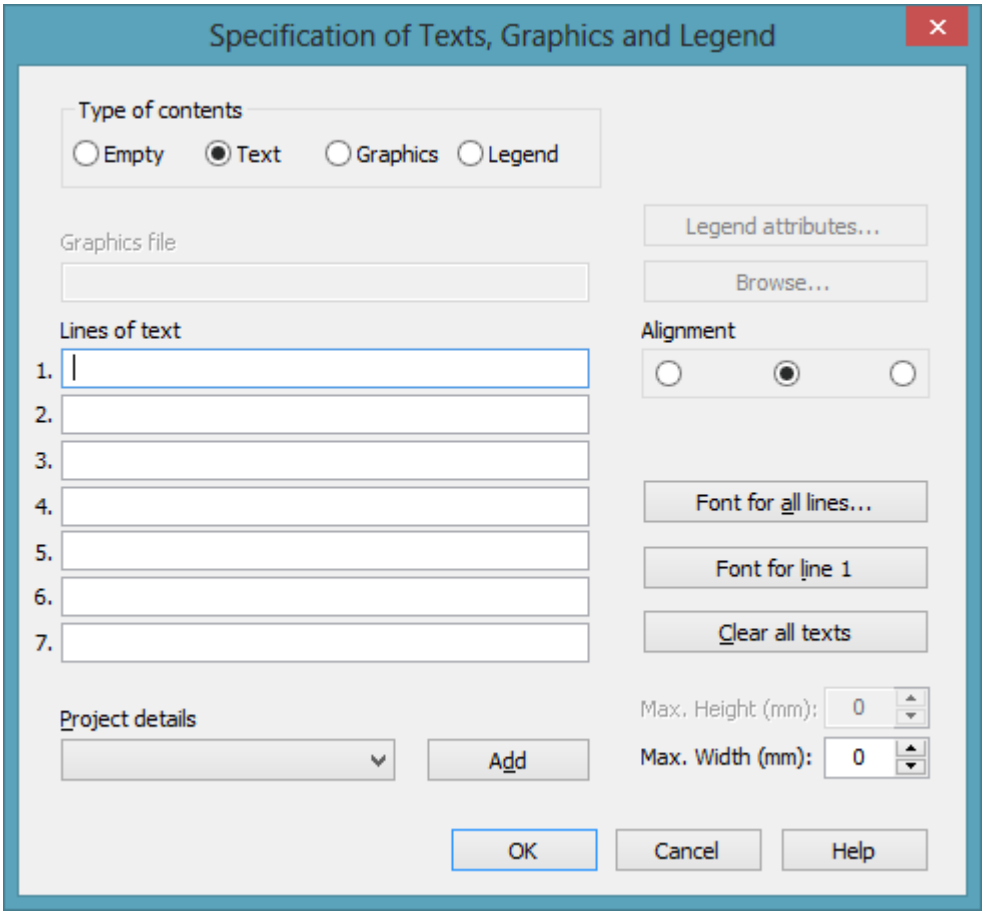
Font

This button lets you get to the Windows dialog box **Font** where you can specify the font for the text at the date line. By the button below, you can get to the Windows color picker, that lets you select a color for the text font of the date line or create a new color.

Turning annotation by 90 degrees

Activate this check box, if the annotation should be displayed in vertical direction.

4.51 The "Specification of Texts, Graphics and Legend" Dialog Box



You can get to this dialog box if you click in the **Border Area** property page on one of the nine buttons above/below the drawing.

Type of contents

Specify the type of information that you want to display at the chosen location:

Empty: If you do not want to output anything at the chosen location, click on this flag.

Text: The text of the six text lines will be displayed at the chosen location.

Graphics: The graphics selected (by the **Browse** button) will be displayed at the chosen location. Graphics are always displayed in alignment centered.

Legend: A legend will be displayed at the chosen location. It describes the layers used in the current diagram.

Following your selection, the sections of the dialog box that are not required are deactivated (all entries are maintained).

Legend attributes

*Only activated when the check box **Legend** has been ticked.* You will open the **Legend attributes** dialog box where you can specify further attributes for the legend.

Graphics file

*Only activated when the check box **Graphics** has been ticked.* Select the graphics file you want to display by clicking on the **Browse** button or type the file name manually in the field. If the selected graphics file is not stored in the installation directory of the VARCHART ActiveX, you must also specify the drive and the directory.

Browse

*Only activated when the check box **Graphics** has been ticked.* Click on this button to reach the **Choose Graphics File** dialog box and select the drive, the directory and the name of the appropriate graphics file.

Lines of text

*Only activated when the check box **Text** has been ticked.* Specify the text (max. 6 lines) you want to display at the chosen diagram position and/or specify substitutes (e.g. &[System date]) to represent project info. If all six lines are empty, the area will not be displayed in the diagram.

Project details

*Only activated when the check box **Text** has been ticked.*

Here you can add several project details (number of pages, page number, system date) to your chart by selecting the appropriate place holder from the list and by clicking on the **Add** button.

The place holders will be replaced by the required data and will continuously be kept up-to-date in the print preview and the printout.

Add

*Only activated when the check box **Text** has been ticked.* When you have selected a project detail from the list, click on **Add** to confirm your choice. The project detail will be inserted in the line where the cursor is currently positioned.

Alignment of text

*Only activated when the check box **Text** has been ticked.* Specify whether the text lines should be output left-aligned, centred or right-aligned.

Font for all lines

*Only activated when the check box **Text** has been ticked.* You will reach the **Font** dialog box where you can specify the font attributes for all six lines. If you use this option to specify the font for all lines, the settings for the font for line 1...6 will be overwritten.

Font for line 1...6

*Only activated when the check box **Text** has been ticked.* To assign a different font to each of the six lines, click on this button. Depending on the line in which the cursor is currently positioned, the notation of this button will change to 1, 2, 3, 4, 5 or 6. You will reach the **Font** dialog box where you can specify the font attributes for each separate line.

Clear all texts

*Only activated when the check box **Text** has been ticked.* Click on this button to delete the contents of all six lines of text.

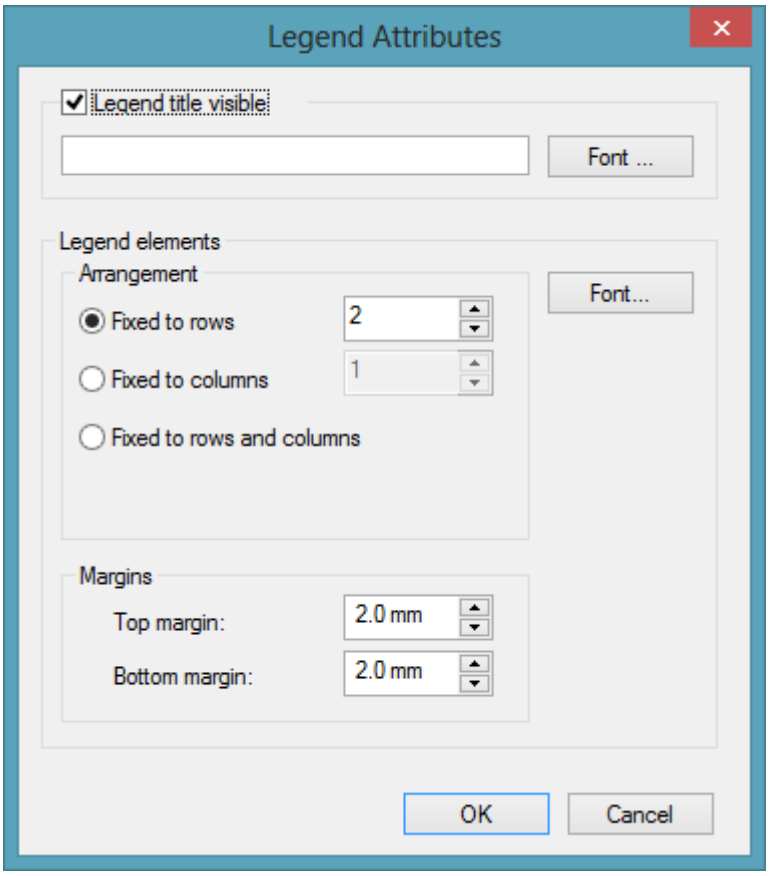
Max. Height (mm)

*Only activated when the check box **Graphics** has been ticked.* If you have specified several fields for text, graphics or legend, you can specify the max. height for the current field to prevent field contexts to be cropped.

Max. Width (mm)

*Only activated when the check box **Text** or **Graphics** has been ticked.* If you have specified several fields for text, graphics or legend, you can specify the max. width for the current field to prevent field contexts to be cropped.

4.52 The "Legend Attributes Dialog Box"



You can get to this dialog at runtime by clicking the corresponding item of the legend's contextmenu or at designtime by clicking the corresponding button in the dialog **Specification of Texts, Graphics and Legend**. The button can only be clicked after having selected **Legend** as **Type of contents**.

Legend title visible

Tick this check box if the legend title shall be displayed and enter a text. By clicking on **Font** you open the corresponding Windows dialog box which lets you specify the font attributes of the legend title.

Arrangement

- Fixed to Rows: Specify the number of rows to be displayed in the legend.
- Fixed to Columns: Specify the number of columns to be displayed in the legend.
- Fixed to Rows andColumns: Specify the number of rows and columns to be displayed in the legend. If the number entered here is lower than the existing layers, the surplus layers are not displayed.

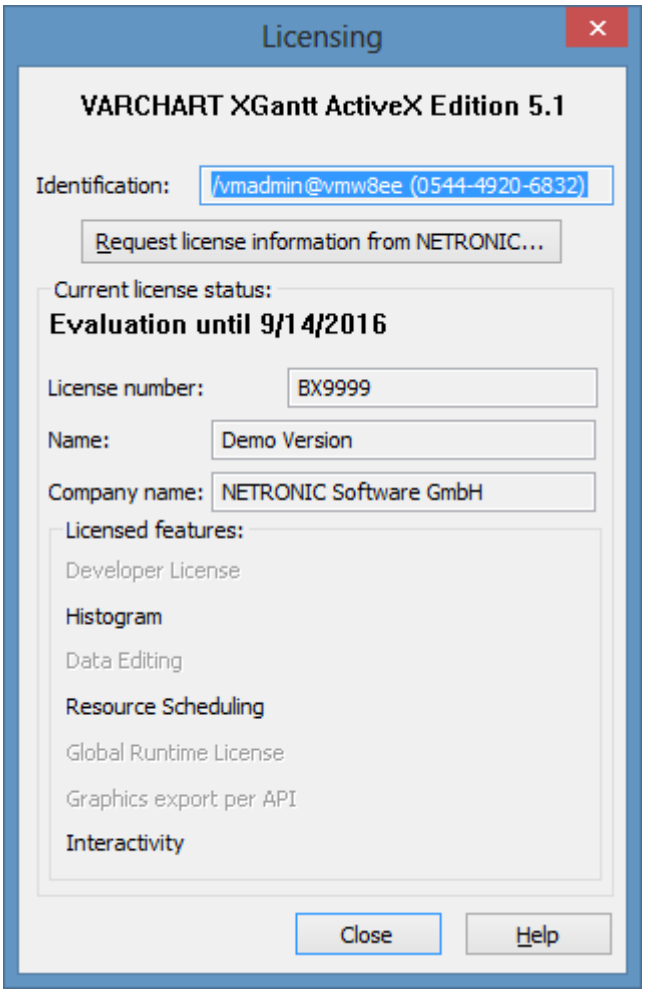
Margins

- Top margin: enter a value for the top margin of the element
- Bottom margin: enter a value for the bottom margin of the element.

Font

By clicking this button you open the Windows **Font** dialog box where you can specify the font attributes for the legend.

4.53 The "Licensing" Dialog Box



You can get to this dialog by the **General** property page.

Before licensing, the program is automatically licensed as a trial version. Compared to the full version, the trial version is subject to restrictions: The trial period for testing the product is limited to 30 days. After this period, all diagrams will show a "Demo" watermark.

Hardware identification

(cannot be edited) The number that is indicated here is calculated by your hardware configuration. NETRONIC needs it for the licensing procedure. When you modify your hardware, you have to renew your licence. Please don´t hesitate to contact the technical support team of NETRONIC.

Request license information from NETRONIC

For licensing, click on this button. Then the **Request License Information** dialog will open.

License number/Name/Company name

(cannot be edited) Indicates your license number, your name and the name of your company.

Licensed features

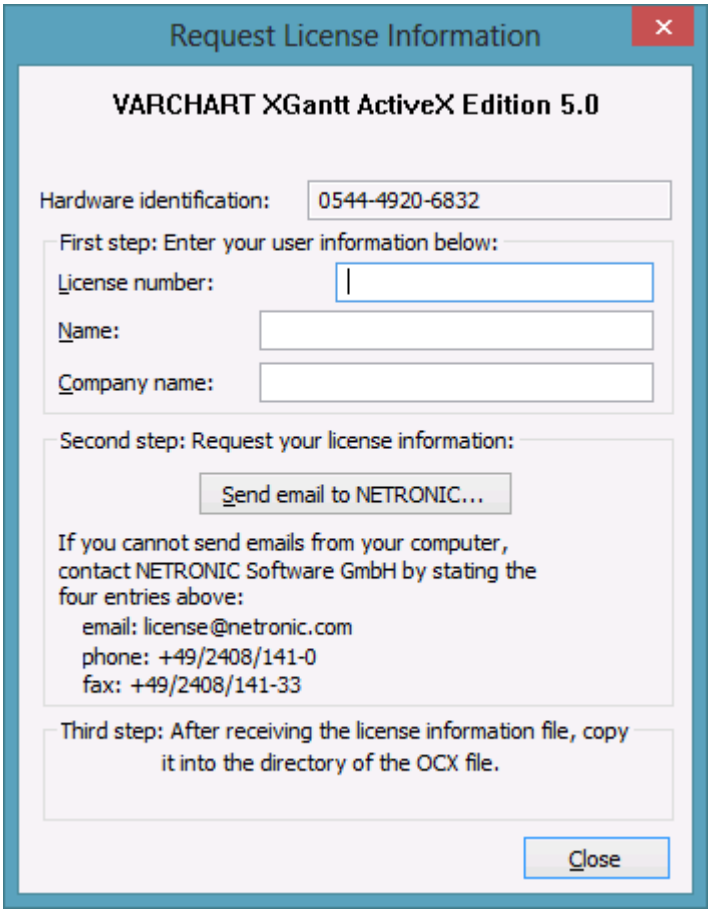
Indicates the modules that have been licensed. If the licensing procedure was successful, the licensed modules are activated.

- **Developer license**
- **Histogram**
- **Global runtime license** (VARCHART ActiveX runs in the runtime mode on each computer.)
- **Single-place runtime licenses** (VARCHART ActiveX has to be licensed individually for each computer to run on.)
- **Graphics export per API**
- **Interactivity**

Close

Quits the dialog box.

4.54 The "Request License Information" Dialog Box



Enter your license number, your name and the name of your company and click on **Send email to NETRONIC**. An email to NETRONIC will be generated automatically. As soon as we have received it, we will generate your license information file (**vcgantt.lic**) and mail it back to you.

After having received the file, please copy it to the directory in which the file vcgantt.ocx is stored.

After licensing, you need to activate the new license in each of your projects. So please open a property page in each of your projects, make some change and store it. Then the new license will be activated.

5 User Interface

5.1 Overview

The following list gives an overview of possible user interactions.

- Navigation in the diagram and in the table
- Zooming
- Marking nodes or layers
- Creating nodes
- Moving nodes
- Moving layers
- Change start/end date
- Delete, cut, copy and paste nodes
- Editing node data
- Editing links
- Anchor boxes to nodes
- Editing group data
- Expanding/collapsing groups
- Moving groups
- Modifying table/diagram ratio
- Modifying table column width
- Editing fields in the table
- Inserting table rows
- Editing the timescale
- Modifying the scaling and the frontiers of sections
- Moving the date line
- Editing the legend
- Setting up pages
- Use the print preview

Context menus (right mouse key):

- for the diagram
- for nodes
- for links
- for groups
- for the timescale
- for the histogram
- for the legend
- for boxes

For further information on user interactions in grouped diagrams or in hierarchically sorted diagrams please read the chapters "Important Concepts: Grouping" or "Hierarchy" respectively.

All these interactions trigger an event so that you will be informed about it and will be able to react to it.

5.2 Navigation in the Diagram and in the Table

Use the following keys and shortcuts for navigating in diagram and table:

- The arrow keys move the marking from one node to the other in the selected direction (for further information, in particular concerning the marking in groups, please see chapter 5.4 "Marking Nodes and Layers".
- **Pos1:** scrolling to the left diagram border
- **Ctrl** + **Pos1:** scrolling to the left upper diagram corner
- **End:** scrolling to the right diagram border
- **Ctrl** + **End:** scrolling to the right lower diagram corner
- **Page up/down:** scrolling one screen page up/down
- **Ctrl** + **Shift J:** scroll to the next date line
- **Ctrl** + ***** (NUM key): the screen section is shifted so that the start of the node is visible

The mouse can also be used for navigating:

- Turn the mouse wheel for scrolling vertically in the diagram or in the histogram (depending on the cursor position)
- By holding down the mouse wheel (or the middle mouse key) and moving the mouse you can scroll in any direction wanted.

5.3 Zooming

The following shortcuts can be used for zooming:

- **Ctrl** + **Num -**: zoom out
- **Ctrl** + **Num +**: zoom in

You can also use the mouse for zooming:

- Turn the mouse wheel while holding down the Ctrl key. For that purpose the usage of the mouse wheel for zooming has to be permitted. This can be done by ticking the **AllowZoomingByMouseWheel** box on the **General** property page or by setting the property **VcGantt1.Zooming-PerMouseWheelAllowed** to **True**. This property is set to **False** by default.

For further information about zoom settings for the output please see chapter 5.21 "Setting up pages".

5.4 Marking Nodes or Layers

To mark a node, click the left mouse key on the node. The first field of the corresponding table line will also be marked.

You can also click on a certain field in the table and with that mark the corresponding activity in the diagram area at the same time.

To mark several nodes which are situated above or below one another in the diagram area, keep the Shift key pressed while clicking on the nodes or on the corresponding table lines in the table area.

Alternatively, you can drag a rectangle around the nodes to be marked, using the left mouse key.

Several nodes which are not situated above or below one another in the diagram area can be marked by keeping the Ctrl key pressed and clicking on the nodes or on the corresponding table lines in the table area.

For groups of the mode **All nodes in one row** and **optimized**: If you navigate downwards, the first node of a group will be marked at first.

If you navigate upwards, the last node of a group will be marked at first.

Within these groups you can use the arrow buttons left/right to navigate to the left/right.

Note: The markings of nodes or table fields/lines are undone by clicking on them a second time or by pressing the ESC-key.

5.5 Creating Nodes

This mode is available only if the **Allow new nodes** option on the **Nodes** property page is activated.

In this mode, the cursor shape changes to a small cross. While in this mode, you can create a node by dragging the mouse and pressing the left mouse button. A little box will appear at the current position of the mouse which shows the current start and end date and the duration of the new node.

Create Activity	
Start:	07.09.2007
End:	09.09.2007
Duration:	2 days

If you are creating a node in a collapsed group in a multi-level grouped diagram, additionally to the small cross an arrow appears: It shows whether the new node will be the first node in the group (arrow up) or the last one (arrow down).

In expanded groups the new node always will be inserted as the first node, on condition hat the cursor is placed in a group title row.

In hierarchically grouped diagrams you always can insert the new node above or below the reference node (dependent on the arrow direction).

If the **Edit new node** option on the **Nodes** property page is activated, the **Edit Data** dialog box will appear, as soon as you release the mouse button. In the **Edit Data** dialog box you can edit all data of the new node.

If you have not defined anything else in your settings, the node just created will appear at the current position of the mouse.

The **Mode: Create Node** can also be activated by setting the property **InteractionMode** to the value **VcCreateNode**.

The event **OnNodeCreate** occurs when the user creates a node. The node object is captured, so that a validation can be made. For the validation, the **Edit Data** dialog box has to be activated. If you set the returnStatus to **vcRetStatFalse**, the node will will be deleted.

5.6 Moving Nodes by Mouse

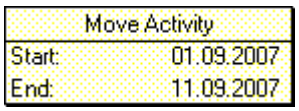
Moving nodes in the diagram

The possibilities of moving a node vary in dependence on the settings on the **Nodes** property page. Find below the description of how to move nodes when the following default settings on the **Nodes** property page are valid (for information about further possible settings please see chapter 4.4 "The Nodes Property Page"):

- Move node when marked
- Move layers as node when shift key pressed

When you position the mouse on a node, the mouse pointer takes the shape of a small square with an arrow pointing left and right (or with four arrows, when the node consists of one layer only). Now you can move the layer by dragging it with the mouse.

If you want to move the complete node (with all layers) press the Shift key while pointing on the node. Now the cursor takes the shape of a small square with four arrows . Hold the Shift key down while dragging the node to a different position. An info box will display the current start and end dates of the node. As soon as you release the mouse key, the node will be dropped at the current position and the box will be closed.



Note: The Shift key has to be pressed only if you want to move a node that consists of more than one layer.

If the **Allow vertical node movement via diagram** box is ticked on the **Nodes** property page, nodes can also be moved in vertical direction.

When a node is being moved vertically in the diagram, a cursor with corresponding arrows indicates in which way the node will be positioned relatively to the other nodes: .

Moving nodes in the table

If the check box **Allow vertical node movement via table** has been ticked, you can also move nodes in the table. Up to now, however, it is only possible to move complete nodes only vertically. When a node is being moved vertically in the table, a cursor with corresponding arrows indicates in which way the node will be positioned relatively to the other nodes: .

The event **OnNodeModifyEx** occurs when the user has modified the length or the position of a node or a value in the **Edit Data** dialog. By the **modificationType** parameter you get further information of the kind of modification. If you set the returnStatus to **vcRetStatFalse**, the modification will be revoked.

5.7 Moving Nodes and Modify Duration by Keys

Usually, the arrow keys <left> and <right> are reserved for various navigating interactions, such as scrolling the diagram, moving a marked field within a node or within the table. These functions can be changed into modifying functions by the **VcGantt.ArrowKeyMode** property so that the user can move, enlarge or reduce the size of a node by them.

- Move nodes

  By simply striking the arrow keys, a node will move; the smallest step size being the same as when moving the node by mouse. The step size can be enlarged by the property **VcGantt.ArowKeyStepMultiplier** and activated by holding the <Ctrl> key down in addition.

  Key functions:

  Arrow key <left/right>: move node

  <Ctrl> + **<Arrow key left/right>**: modify step size

- Modify Duration

  The duration can only be modified for all **visible** layers and only, if only one node ist marked. The above mentioned multiplier for the step size can be used as well.

  Key functions:

  <Shift> + **arrow key <left/right>**: change size of the node and thus modify its duration

  <Shift> +**<Ctrl>** + **arrow key <left/right>**: modify step size

A window displaying information on the position will remain on the screen for a few more seconds after the interaction is finished to let the user read its content.

For further information about the corresponding API properties please see the API reference guide.

The event **OnNodeModifyEx** occurs when the user has modified the length or the position of a node or a value in the **Edit Data** dialog. By the **modificationType** parameter you get further information of the kind of modification. If you set the returnStatus to **vcRetStatFalse**, the modification will be revoked

5.8 Moving Layers

Press the left mouse key to mark a layer and then move the mouse to shift the layer until releasing the mouse button again. When moving the layer horizontally, the **Move Layer** box continuously displays the current start and end dates of the layer, while the duration remains constant. Layers can only be moved within a row; there is no way to move a layer to a different row.

Move Layer	
Start:	09.09.2007
End:	23.09.2007

If you move a symbol layer, the **Move Layer** box will look like this:

Move Layer	
Date:	09.09.2007

5.9 Change Start/End Date

In a similar way you can modify just the start or the end date of a layer if you position the cursor on the outer left or right edge of the layer. The **Change Start Date** or **Change End Date** box (as appropriate) will appear that continuously displays the current start or end date. The duration will change.

The event **OnNodeModify** occurs when the user modifies the length or the position of a node or a value in the **Edit Data** dialog. By the parameter **modificationType** you can obtain more information on the kind of modification. If you set the returnStatus to **vcRetStatFalse**, the modification will be revoked.

5.10 Delete, Cut, Copy and Paste Nodes

By using the Del key you can delete marked nodes.

Ctrl-X lets you cut marked nodes, by Ctrl-C you can copy nodes.

With the Shift key pressed, you can use the arrow up/down buttons to mark several nodes.

If the area of marked nodes contains a group in the mode **All nodes in one row** and **optimized**, all nodes of this group will be marked.

If the first node of such a group is marked and if you move the cursor with pressed Shift key into another row, all nodes of the target and of the start row will be marked.

You can insert copied or cut nodes via Ctrl-V above the target row (the row in which a node is marked).

You can insert copied or cut nodes via Ctrl-Shift-V below the target row.

The insertion position relative to the reference node will be indicated by appropriate arrows at the cursor symbol.

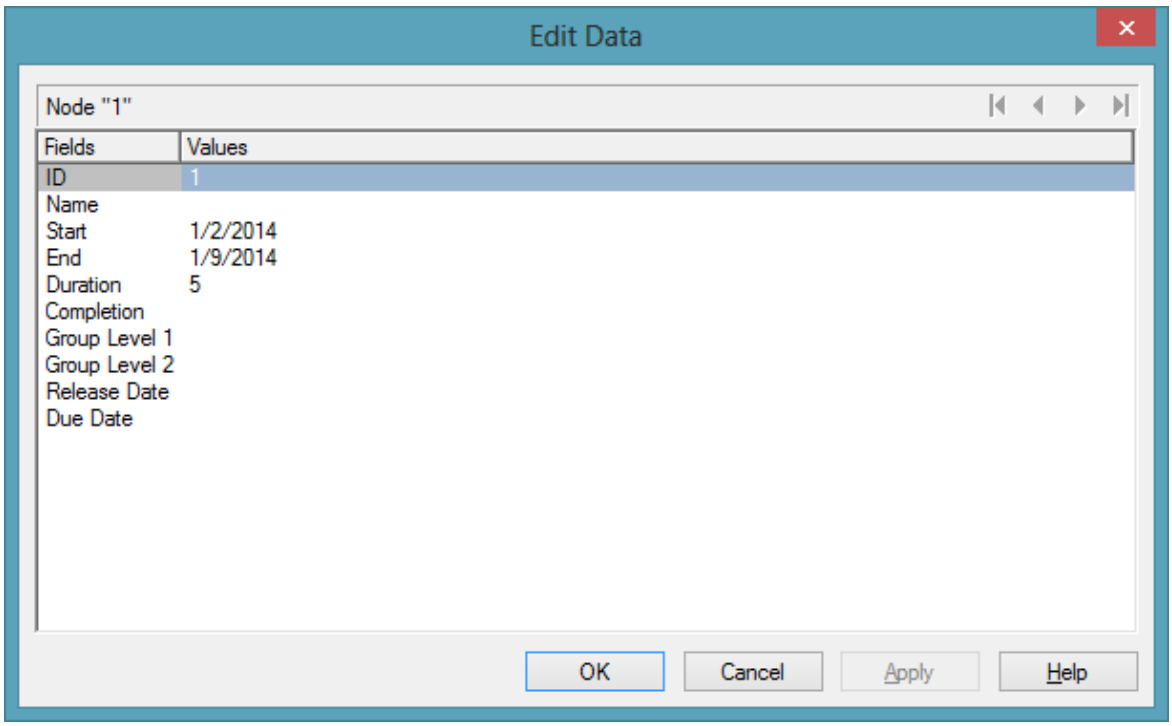
When you insert nodes, the order of grouped nodes will not be changed.

Nodes cannot be inserted in empty groups in the mode **All nodes in one row** and **optimized**.

5.11 Editing Node Data

In the dialog "Edit data" you can edit all node data. You open this dialog by either clicking on the **Edit** item of the corresponding context menu or by double-clicking on the node.

To edit several nodes, you mark the desired nodes and then click the **Edit** item of the context menu of one of the marked nodes to pop up the **Edit Data** dialog. Now you can edit the data of the marked nodes one after another



By double-clicking on a node, the event **OnNodeLDblClick** is triggered.

Modifiying a node interactively, e.g. by the **Edit Data** dialog, triggers the event **OnNodeModify**. By the **modificationType** parameter you get further information of the kind of modification. If you set the returnStatus to **vcRet-StatFalse**, the modification will be revoked.

Fields

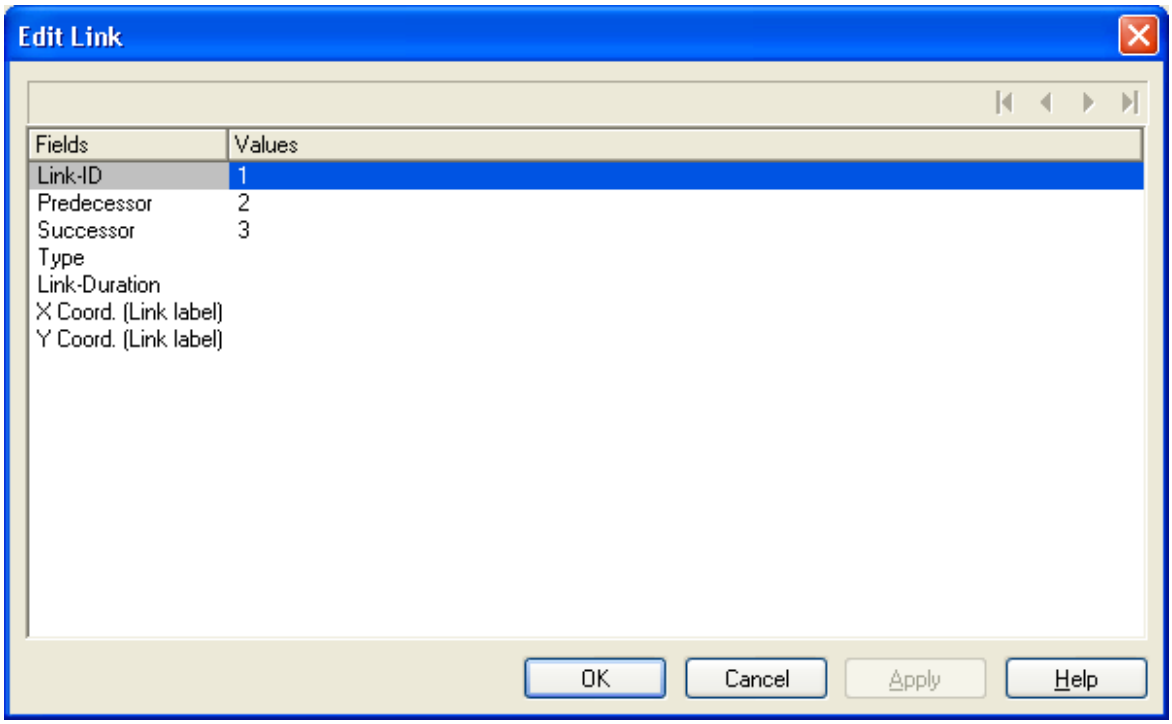
This column displays the data fields that define the marked node. The data fields available are the ones defined by the data definition in the **Administrate data tables** dialog. Only data fields that are **not** defined as **hidden** are displayed.

Values

This column lets you edit the values of the nodes marked, but only if they have been defined to be **Editable> in the Administrate Data Tables** dialog. If you edit a data field of the **Date/Time** type, a Date dialog will appear that you can select a date from.

The **Date Output Format** is defined on the **General** property page. When editing a field of the type **Integer** you can modify the value by a spin control that delivers the desired values via up and down arrows.

5.12 Edit Links

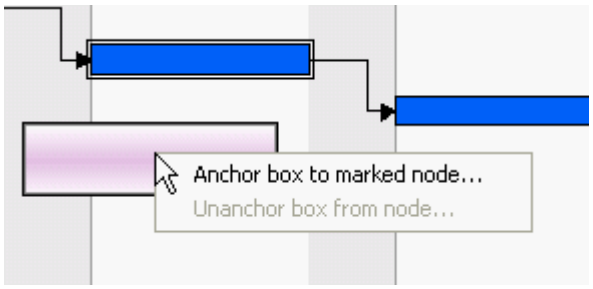


This dialog can be invoked by the method **VcGantt.EditLink**. Here you can view and edit the data of the marked link. The ID of the link is indicated at the first position of the list.

5.13 Anchor Box to Node

Boxes can be anchored to nodes either interactively (mouse + Shift key or context menu) or by using the corresponding API properties and methods.

- **Anchoring by mouse:** Point with the mouse to the box you want to tie to a node and press the Shift key. A little anchor appears. Keep the Shift key pressed and draw a line between the box and the desired node. The box is now anchored to the node. If you have ticked the check box **Anchoring line visible** in the **Administrate boxes** dialog, a line is displayed. Follow the same steps to untie the box again.
- **Anchoring over contextmenu:** Mark the node to which you want to anchor the box and select **Anchor box to marked node** from the context menu of the box. If the context menu does not pop up, you have to tick **Show context menu for the box** on the **General** property page.



Select **Unanchor box from node** to untie the box again.

If you want to tie the box to another node, carry out the same steps as described above, either by mouse or over context menu.

- **Anchoring via API:** Please see the API Reference Guide for a detailed description of the property **AnchoringInteractionsAllowed** and the method **AnchorToNode** of the object **VcBox**

A box which was anchored can be still moved interactively (provided that you have ticked the check box **Moveable** in the **Administrate boxes** dialog).

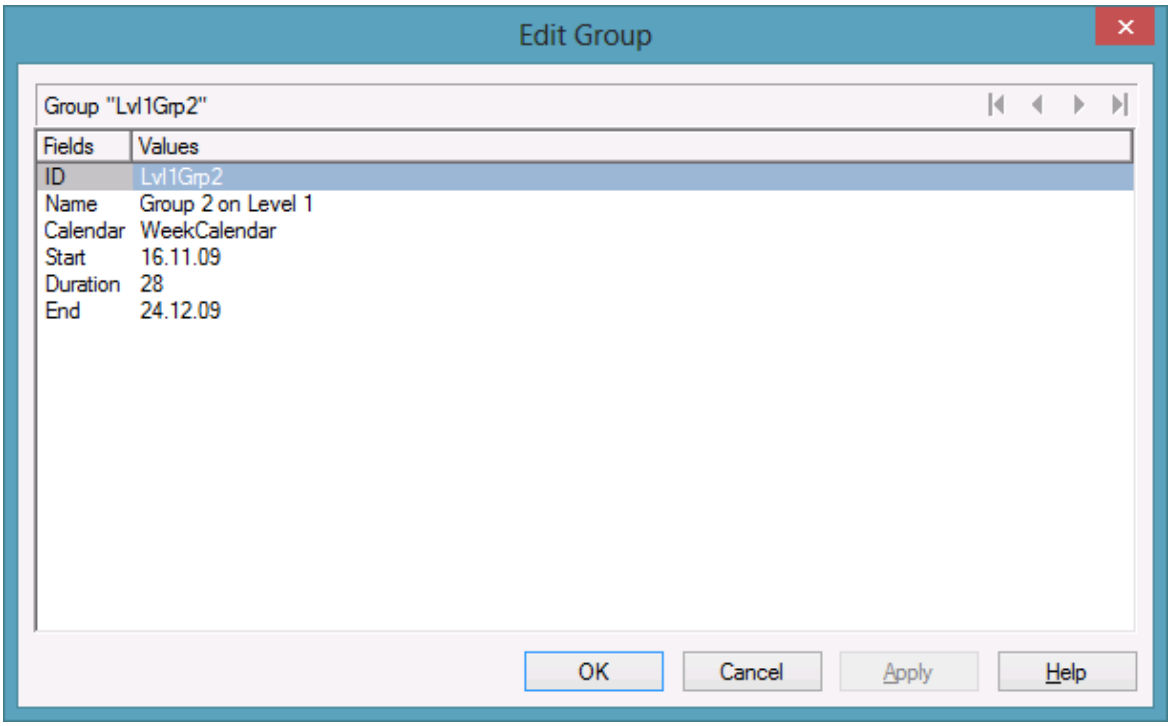
If you move a node which is anchored to a box, the box is moved as well. If the node is collapsed, the box is collapsed as well, thus becoming invisible. When the node is expanded the box is visible again.

If a box is tied interactively to a node, its position on the screen will be maintained. The offset values which are used as basis are converted according to the reference points (Origin, ReferencePoint). If, for example, a box with a certain offset refers to a chart at the top left (origin) and then is anchored to a node, an offset to the the top left node is calculated automatically. This makes sure that the position on the screen will not be

altered. If the box is untied from the node the calculation is carried out backwards.

This method is applied as well when using the API property **AnchorToNode** but not when setting the property **NodeID**.

5.14 Edit Group data



You can get to this dialog by the context menu of the group or by double-clicking a group layer (which will only be displayed if in the **Grouping dialog** the box **Group node visible** has been ticked).

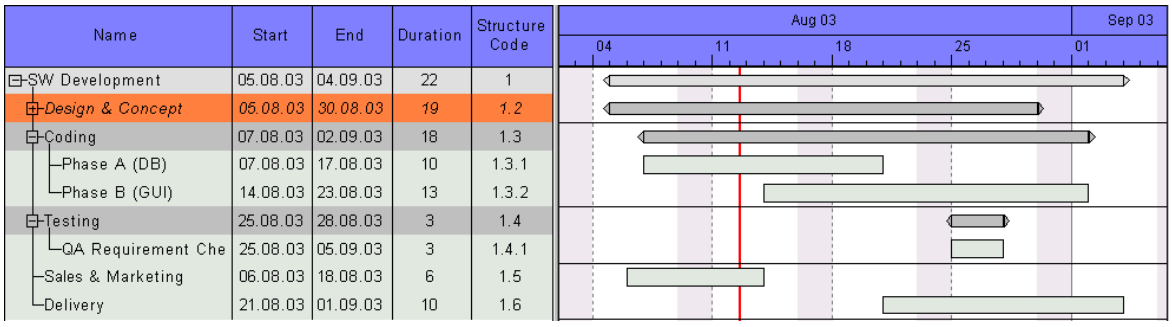
The dialog lets you edit the data of one group or, if more than one group has been marked, the data of every marked group one after the other.

The number of the current group out of the total number of marked groups is indicated above the list.

The arrow buttons above the list allow to navigate to the previous or next (or first or last) marked node.

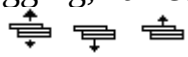
5.15 Collapsing/Expanding Groups

If a grouping is specified and the **Modifications allowed** box in the **Grouping** dialog is ticked, you can expand a collapsed group/collapse an expanded group by double-clicking on the group heading or by clicking on the **plus** or **minus** symbol of the group heading.



The event **OnGroupModify** occurs when a user interactively modifies a group. The group object, the type of modification and the return status are returned. If you set the returnStatus to **vcRetStatFalse**, the modification will be revoked.

5.16 Moving Groups

Groups can be moved vertically in the table as well as in the diagram (by dragging the summary bar) when the checkboxes **Moving groups vertically via table** and/or **Moving groups vertically via diagram** in the dialog **Grouping** have been ticked. While dragging, a corresponding cursor indicates where the group will be positioned  .

Tip: Groups can only be moved within a parentgroup.

5.17 Editing Fields in the Table

To edit the contents of a table field click on it and either enter new contents or modify the current one.

There are further ways of editing the field contents in the table which are only available after having ticked the **Extended Editing behavior** box on the **General** property page.

You can then modify date and time fields by clicking on the arrow button. For further information about the usage of the date dialog box see chapter 4.40 The "Specify Date Lines" Dialog.

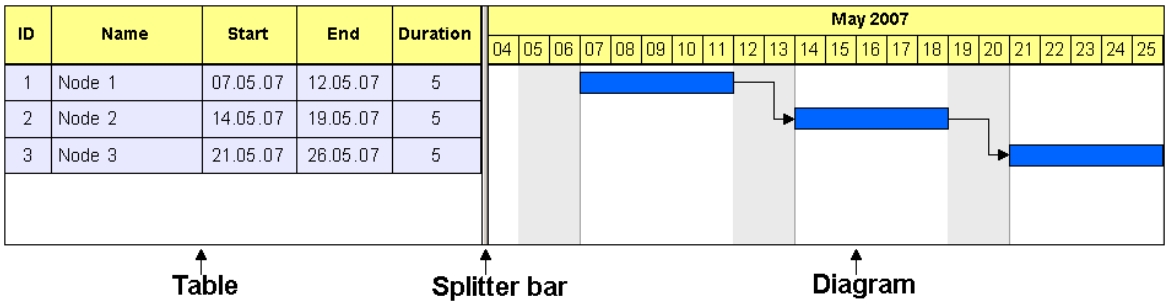
The value of numeric data fields may be increased or decreased by clicking on the corresponding arrow buttons.

For further information about extended editing see chapter 4.2 "The General Property Page".

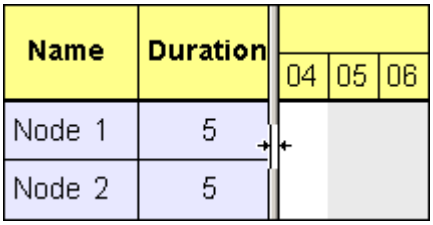
Note: By pressing the Esc-key you leave the edited fields without saving the changes.

5.18 Modifying Table/Diagram Ratio

The table and the diagram are separated from each another by a splitter bar.



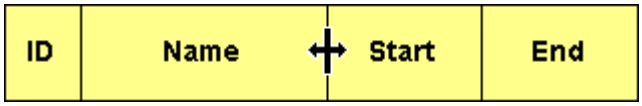
When you move the mouse over the splitter bar, the pointer shape changes to a double vertical line with an arrow to the left and right.



By dragging the mouse, you can now change the width ratio of the table to the diagram. (The maximum table width is limited by the total of column widths specified in the **Edit Table** dialog.)

5.19 Modifying the Table Column Width

You can change the width of a column in the table interactively by moving the separation line between the columns in the table caption.



You can change the width of a table column in the table caption only.

The event **VcTableWidthChanging** occurs when the user modifies the width of the table. The table and the modified diagram aspect ratio are returned. If you set the returnStatus to **vcRetStatFalse**, the modification will be revoked.

The event **VcTableColumnWidthChanging** occurs when the user modifies the width of a table column. The table, the index and the current width (as 1/100 mm) of the modified column are returned. If you set the returnStatus to **vcRetStatFalse**, the modification will be revoked.

The column width can be calculated automatically, too. For that, on the **General** property page the **Allow table column width optimization** check box has to be activated. Then at run time, a double-click on a column separation line will cause that the width of the column on the left will be adapted automatically to the length of the texts which it contains. This will trigger the **VcTableColumnWidthOptimizing** event. If the optimization has occured, the event **VcTableColumnWidthChanging** will be triggered.

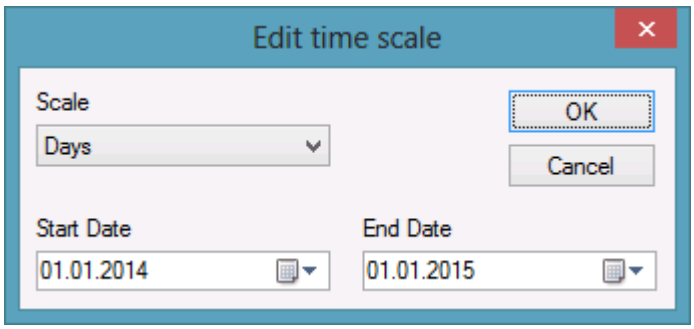
5.20 Inserting table rows

If the check box **Extended editing behavior** on the **General** property page was ticked, the Ins-key can be used for inserting a table row above the current one. If no row was marked, the new line is inserted at the end of the table.

5.21 Editing the Time scale

In the **Edit Time scale** dialog box you can set the time scale type (minutes, hours, days, weeks, months) and the start and end of the time scale.

You can open this dialog by double-clicking on the time scale or selecting the corresponding context menu item. When shifting the beginning of the time scale, the beginning must not be shifted beyond the end of the first section, if more than a single section was defined.



By double-clicking on the time scale, the event **OnTimeScaleLDblClick** is triggered. The TimeScale object and the mouse position (x,y-coordinates) are returned. If you set the returnStatus to **vcRetStatFalse**, the integrated **Edit Time scale** dialog box will be revoked.

The "Edit time scale" dialog

Scale

Select the time scale. Choose between minutes, hours, days, weeks and months.

Start Date

Specify the start date of the time scale. If you click on the arrow button, a Date dialog will appear that you can select a date from.

The date output format is defined on the **General** property page.

End Date

Specify the end date of the time scale. If you click on the arrow button, a Date dialog will appear that you can select a date from.

The date output format is defined on the **General** property page.

5.22 Modifying the Scaling and the Frontiers of Sections

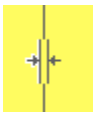
Scaling Time scale Sections

You can rescale a time scale section interactively by positioning the mouse cursor onto the section, pressing the left mouse key and dragging the mouse towards the left or right. The shape of the cursor will change to a vertical line with an arrow to the left and right. Dragging the cursor towards the left will downsize the width of the time scale units, dragging it to the right will blow them up. While dragging an info box will pop up to inform you about the percentage by which the time scale section is altered.

Note: The closer you place the cursor to the beginning of a section, the stronger the enlargement/downsizing will be. If you intend to enlarge/downsize a lot, you are suggested to place the cursor close to the beginning on the left, while for smaller adjustments placing the cursor towards the end on the right is suggested.

The event **OnTimeScaleSectionRescale** occurs when the user rescales a section of the time scale. The TimeScale object, the section index and the current **BasicUnitWidth** are returned. If you set the return status to **vcRet-StatFalse**, the modification will be revoked.

Moving the Limits of a Time scale Section



You can move the limits between two time scale sections by shifting the separating line between them. The shape of the cursor will change to a vertical double-line with an arrow to the left and right.

The event **OnTimeScaleSectionStartModify** occurs when the user modifies the start date of a section interactively. The TimeScale object, the section index and the current start date are returned. If you set the returnStatus to **vcRetStatFalse**, the modification will be revoked.

5.23 Moving the Date Line

You can modify the date of a date line by moving it via the mouse.

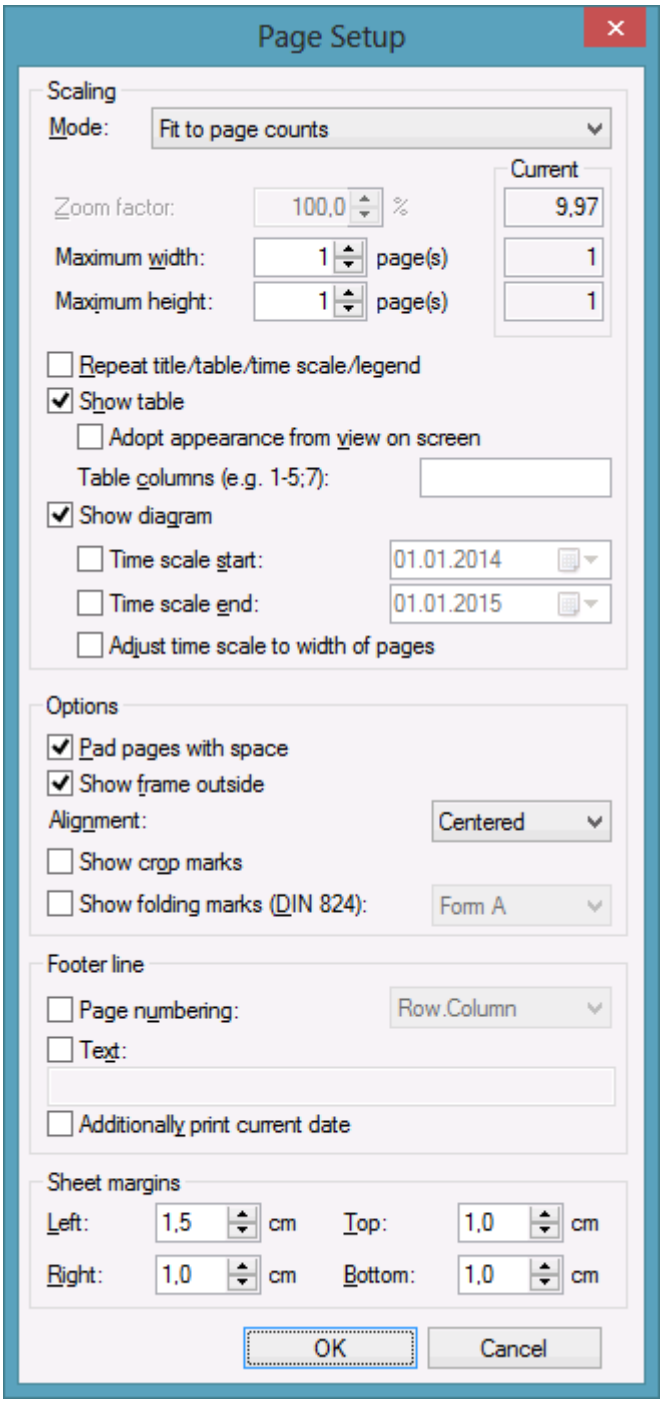
Before, on the **Specify Date Line** dialog the **Moveable** check box of the corresponding date line has to be activated for the relevant date line.

Beside, you can generate date lines via the API.

The event **OnDateLineModify** occurs when the user has moved a date line. The modified date line object is captured and returned so that you receive the new values. If you set the return status to **vcRetStatFalse**, the modification will be revoked.

5.24 Setting up Pages

All settings concerning the page layout can be made in the corresponding dialog which can be opened either by clicking the **Page setup** item of the diagram contextmenu or by clicking the corresponding button in the **Print preview**.



Mode

By selecting a scaling mode from the drop down list and setting the corresponding values **Zoom factor** and **Maximum width/height** you specify a zoom factor for your output. After having clicked the **Apply** button, the values which result from your settings are shown under **Current**.

Zoom factor

100% is equivalent to the original size; a smaller value correspondingly reduces the size of the diagram, a greater value increases it.

Fit to page counts

By selecting this option you can specify the maximum number of pages, both heightwise and widthwise, into which the diagram may be split for the output (**Maximum width, Maximum height**. If necessary, one of the two values may be ignored in order to print the diagram as large as possible while preventing it from being distorted.

Zoom with horizontal fit

This option lets you regulate the pagination by selecting a zoom factor as well as a fixed number of pages in width. This number of pages is reached by downsizing or expanding the time scale.

Repeat title/table/timescale/legend

By ticking this check box title, table, timescale and leged of a diagram that was partitioned into pages will be added to each page.

Show table

Specify whether the table is to be printed or not. If you don't tick the check box, the table will not be printed.

Adopt appearance from view on screen

This option lets you specify whether the table width that is currently shown on the screen is to be adopted for the print preview and for the output.

This feature can also be set by the property **VcPrinter.TableWidth-AdoptionFromViewOnScreen**.

Show table columns

Here you can set the number of table columns to be printed. Specify single columns or ranges of columns, that are to be separated by commas or semicolons. Example: "1;5-7;3" specifies the columns 1 and 3 and the range from 5 to 7.

Show diagram

Specify whether the diagram (timescale and layers) shall be also printed or not.

Time scale start

This option lets you specify the start date of the time range to be used for the output. The time range can only be restriced in comparison to the time range displayed on the screen. Hence only a later start date than that having been set by the VcGantt property **TimeScaleStart** leads to a modified output.

This feature can also be set by the property **VcPrinter.TimeColumnStart-Date**.

Time scale end

This option lets you specify the end date of the time range to be used for the output. The time range can only be restriced in comparison to the time range displayed on the screen. Hence only an end date prior to that having been set by the VcGantt property **TimeScaleEnd** leads to a modified output.

This feature can also be set by the property **VcPrinter.TimeColumnEnd-Date**.

Adjust time scale to width of pages

This option leads to a better utilization of the printing pages:

- If scaling fit to page is selected: The zoom factor is calculated in such a way that the space of the selected number of pages is fully used for printing into the height while the time scale gets downsized or enlarged so that the selected number of pages is used to full capacity into the width.
- If a scaling via zoom factor is selected: The time scale gets downsized or enlarged so that the selected number of pages is being used to full capacity into the width.

Pad pages with space

This option lets you specify whether enough space is to be left between the diagram and the boxes of the title and legend area so that the boxes are always printed in full width and are fixed to the margin. If the option is not selected, there will be no space left between the diagram and the boxes and their width may vary on the different pages depending on the diagram.

Frame outside

If you tick this box, each page will be given a frame, otherwise a frame will be drawn around the whole of the diagram. When the **Repeat title/table/time scale legend** check box has been ticked, a frame will be drawn around the whole diagram

Alignment

Select one of the possible alignments for the diagram from the list.

Show crop marks

If you tick this check box, crop marks will be printed on the edges of the diagram that help gluing together the single pages to get a complete chart.

Show folding marks (DIN 824)

Specify folding marks to fold your drawing according to DIN standard 824 (current version from 1981) for the folding of constructional drawings. The following formats are available:

- **Form A:** includes a filing margin on the left side so that the folded drawing can be punched and filed away without flexi filing fastener
- **Form B:** slightly smaller so that a flexi filing fastener can be applied and together with the fastener the drawing corresponds to the width of DIN A4.
- **Form C:** the folded drawing is not to be punched but to be put in a sheet protector

The available folding marks can be displayed for every format, whereas the DIN 824 only mentions the formats DIN A0 to A3 explicitly.

Page numbers

If you tick this check box, a page number will be displayed in the bottom left-hand corner of each page. The following options are available:

- **Row.Column**: Useful for charts stretching across more than one page both heigthwise and widthwise. The vertical position of the page is displayed before the dot, the horizontal position after it.
- **Column:Row**: Useful for charts stretching across more than one page both heigthwise and widthwise. The horizontal position of the page is displayed before the dot, the vertical position after it.
- **Page/Count**: The current page number is displayed before the slash and after it the total number of pages: 1/6, 2/6 etc.

Text

Please tick this check box to set a text into the bottom left-hand corner of each page. If there is a page number, the additional text will be placed right of it.

For numbering the pages you may enter in **Additional text** the following place holders which will be replaced with the appropriate contents on the printout:

{PAGE}	= consecutive numbering of pages
{NUMPAGES}	= total number of pages
{ROW}	= line position of the section in the complete chart
{COLUMN}	= column position of the section in the complete chart

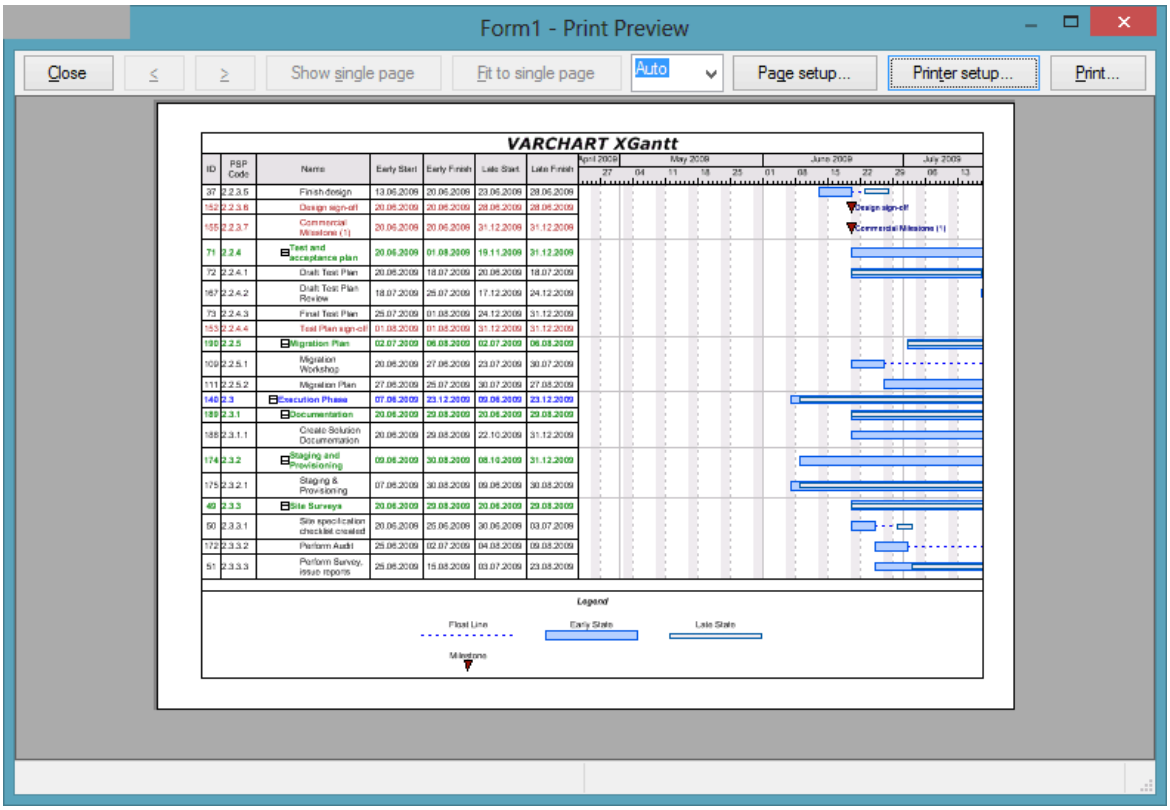
Additionally print current date

If you tick this check box, the printing date of will be displayed in the bottom left corner. If there is a page number or an additional text, the print date will be placed right of them.

Sheet margins

The fields **Top, Botttom, Left** and **Right** let you set the margin between the diagram and the edge of the paper sheet (unit: cm).

5.25 Print Preview



Before printing, you can view the diagram in the print preview where it will be displayed as defined by the settings of the **Page Setup** dialog and as it will be printed.

You can view single pages or an overview of all pages or you can zoom and print a certain section of your diagram interactively.

The status bar shows the total number of pages and their horizontal and vertical spreading. In the **Single Page** mode, also the number of the current page is shown.

Close

By clicking on this button, you will leave the page preview and return to your diagram.

<

*Only activated when the **Single** button has been pressed*. If the diagram consists of more than one page, you can click this button to view the previous page. You traverse the pages horizontally starting from the bottom right and finishing at the top left page.

>

*Only activated when the **Single** button has been pressed*. If the diagram consists of more than one page, you can press this button to view the next page. You traverse the pages horizontally starting from the top left and finishing at the bottom right page.

Show Single Page/Overview

If the diagram consists of more than one page you can either view the pages one by one or in the overview. The overview shows all pages, their size depending on the total number of pages. The **Single Page** mode inititally shows the first page in full size, the buttons [≥] and [≤] allowing to browse through the pages. By double-clicking a page you can easily switch between the two modes **Single Page** and **Overview**.

If you want to zoom a certain section of your diagram, switch to the **Single Page** mode and with the mouse draw a rectangle around the desired section while holding down the left mouse button. As soon as you release the button, the selected section will be enlarged and can be printed by clicking the [Print area...] button that appears in place of the **Print** button. Please note that the zooming factor will not influence the scaling factor set in the **Page Setup** dialog.

Fit To Single Page

This button lets you scale down a multiple-page diagram to one page. The **Fit To Single Page** mode also allows to zoom a certain section as described under **Show Single Page/Overview**

Zoom factor

You can modify the size of the diagram by selecting a zoom factor from the list or by defining an individual one. This is only possible in the "Show Single Page" mode. To modify the zoom factor you can also use the scroll-wheel while holding down the <CTRL> key. The zoom factor it will not modify the size of the output. Depending on the selected zoom factor, vertical and/or horizontal scroll bars will be displayed. You can also use the mouse wheel to scroll vertically, holding down <Shift> to scroll horizontally.

The zoom factor **Auto** is the pre-set default and will always enlarge or downsize the sheet to the full size of the screen.

Page Setup

When clicking on this button, you will get to the dialog **Page Setup** to modify page settings.

Printer Setup

*Only visible if the check box **Use PrintDlgEx dialog** on the **General** property page has not been ticked*.

When clicking on this button, you will get to the Windows dialog **Printer Setup**, where you can modify printer settings.

Print/Print Area

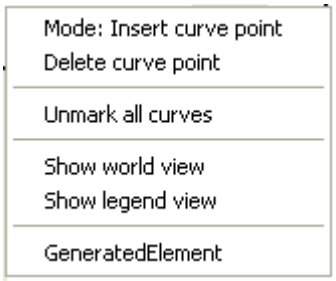
Click on this button to reach the Windows **Print** dialog box to start the print procedure.

If you have zoomed a section in the page preview, the button's label will change to **Print Area** and when you click it, the **Selection** radio button in the Windows **Print** dialog box will already be selected. If you click on **OK** the section displayed on the screen will be printed.

Please note that the zooming factor will not influence the scaling factor set in the **Page Setup** dialog.

5.26 Context Menu of the Curve

If you press the right mouse button in an empty section of the histogram or on a curve, the below context menu will occur:



If the user presses the right mouse button on an empty section of the histogram or on a curve, the event **OnHistogramRClick** or **OnCurveRClick** is triggered, respectively, if the user presses the right mouse button on an empty section of the histogram or on a curve. The histogram or curve object and the mouse position (x,y-coordinates) are returned. You can suppress the integrated context menu at the given position by setting the returnStatus to **vcRetStatNoPopup** and pop up your own context menu.

Mode: Insert curve point

In this mode you can add a curve point by pressing the left mouse button.

Delete curve point

To delete a curve point, click on it with the right mouse button and select the option **Delete curve point** in the context menu.

Unmark all curves

All curves will be unmarked.

Show world view

This menu item lets you switch on or off the world view. The world view is an additional window that shows the complete diagram including the histogram. A frame points out the section currently displayed in the main window.

Show legend view

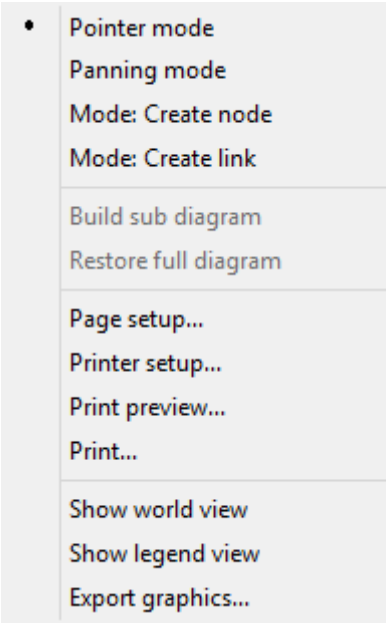
This menu item lets you switch on or off the legend view which is an additional window for showing the legend.

Curves

If available, the API curves are indicated in this context menu, where they can be marked.

5.27 Context Menu of the Diagram

If you press the right mouse key when the cursor is positioned in the diagram area (but not on a node), the following context menu will appear:



The event **OnDiagramRClick** occurs when the user clicks the right mouse key on the diagram, not hitting a node. The position of the mouse (x,y-coordinates) is captured, so that you can for example display your own context menu at the appropriate location. If you set the returnStatus to **vcRetStatNoPopup**, the integrated context menu will be revoked.

Pointer mode

The pointer mode is the default mode. It allows all types of interactions except for generating nodes and links.

Panning mode

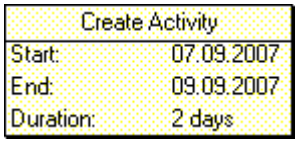
In the panning mode you can move certain screen sections by way of a cursor shaped like a hand.

The panning mode has to be activated on the **General** property page.

Mode: Create node

This mode is available only if the **Allow new nodes** option on the **Nodes** property page is activated.

In this mode, the cursor shape changes to a small cross. While in this mode, you can create a node by dragging the mouse and pressing the left mouse button. A little box will appear at the current position of the mouse which shows the current start and end date and the duration of the new node.



If you are creating a node in a collapsed group, additionally to the small cross an arrow appears: It shows whether the new node will be the first node in the group (arrow up) or the last one (arrow down).

If the **Edit new node** option on the **Nodes** property page is activated, the **Edit Data** dialog box will appear, as soon as you release the mouse button. In the **Edit Data** dialog box you can edit all data of the new node.

If you have not defined anything else in your settings, the node just created will appear at the current position of the mouse.

The **Mode: Create Node** can also be activated by setting the property **InteractionMode** to the value **VcCreateNode**.

The event **OnNodeCreate** occurs when the user creates a node. The node object is captured, so that a validation can be made. For the validation, the **Edit Data** dialog box has to be activated. If you set the returnStatus to **vcRetStatFalse**, the node will will be deleted.

Mode: Create link

The cursor shape changes into a pencil. Use the mouse to draw a link between two nodes and create a finish-start link.

This mode is available only if the **Show Links** option on the **Links** property page is activated.

The event **OnLinkCreate** occurs when the user creates a link between two nodes. The generated link object is returned, so that a validation and if necessary a data base entry can be made. If you set the returnStatus to **vcRet-StatFalse**, the link will be deleted again.

Mode: Create box

This mode is available only if the **Allow new boxes** option on the **General** property page is activated.

While in this mode, you can create a box by dragging the mouse and pressing the left mouse button.

The **Mode: Create box** can also be activated by setting the property **InteractionMode** to the value **VcCreateBox**.

Also see the events **OnBoxCreate** and **OnBoxCreateComplete**.

Build sub diagram

(only active if nodes are marked) Select this item to display a subdiagram of the marked nodes.

Restore full diagram

*(only active if the option **Build sub diagram** has been selected before)* Select this item to restore the full diagram.

Page setup

The **Page Setup** dialog box appears.

The **Page Setup** dialog box also can be invoked by the VcGantt method **PageLayout**.

Print setup

*Only selectable if the check box **Use PrintDlgEx dialog** on the* <!eGeneral *property page has not been ticked*.

The Windows **Print Setup** dialog box appears. This dialog box also can be invoked by the VcGantt method **PrinterSetup**.

Print preview

The **Page Preview** dialog box appears. This dialog box also can be invoked by the VcGantt method **PrintPreview**.

Print

Select the **Print** option to reach the Windows **Print** dialog box. This dialog box also can be invoked by the VcGantt method **PrintIt**.

Show world view

This menu item lets you switch on/off the world view. The world view is an additional window that shows the complete diagram. A frame marks the diagram section currently displayed in the main window. If you move this frame with the mouse, the according diagram section is displayed in the main window.

The world view also can be displayed oder hidden by the property **VcWorldView.Visible**.

Show legend view

This menu item lets you switch on or off the legend view. The legend will appear in a separate window.

The legend view also can be displayed oder hidden by the property **VcLegendView.Visible**.

Export Diagram

When selecting this menu item, you will get to the Windows dialog box **Save as**, that lets you save the diagram as a graphics file.

This dialog box also can be invoked by the VcGantt method **ShowExport-GraphicsDialog**.

When exporting, the size of the exported diagram will be calculated this way:

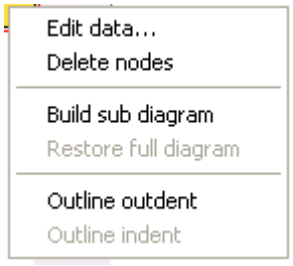
- PNG: a resolution of 100 dpi and a zoom factor of 100% are assumed. If alternatively a value of <= -50 is specified in the parameter SizeX, the absolute number will be used as DPI input.
- GIF, TIFF, BMP, JPEG: a resolution of 100 dpi and a zoom factor of 100% are assumed. If alternatively a value of <= -50 is specified in the parameter SizeX, the absolute number will be used as DPI input. In addition, an internal limit of 50 MBs of memory size is required for the uncompressed source bit map in the memory; so larger diagrams may have a smaller resolution than expected.

- WMF: A fixed resolution is assumed where the longer side uses coordinates between 0 and 10,000 while the shorter side uses correspondingly smaller values to keep the aspect ratio.
- EMF/EMF+: The total resolution is adopted, using coordinates scaled by 1/100 mm.

For further details on the different formats please read the chapter "Important Concepts: Graphics Formats".

5.28 Context Menu of Nodes

If you click the right mouse button on one or several marked nodes, the below menu will appear:



The event **OnNodeRClick** occurs when the user clicks the right mouse button on a node (location = vcInDiagram) or on a table entry related to an activity (location = vcInTable). The node object hit and the mouse position (x,y-coordinates) are returned, so that you can display a context menu at the appropriate position. If you set the returnStatus to **vcRetStatNoPopup**, the integrated context menu will be revoked.

Edit Data

Opens the **Edit Data** dialog box. If you marked more than a single node, you can edit them right away.

Delete Nodes

Select this option to delete the marked node(s).

Build sub diagram

Select this item to display a subdiagram of the marked nodes.

Restore full diagram

*(only active if the option **Build sub diagram** has been selected before)* Select this item to restore the full diagram.

Outline outdent

(only for hierarchy) The position of the marked node in the hierarchy will be increased.

Outline indent

(only for hierarchy) The position of the marked node in the hierarchy will be decreased.

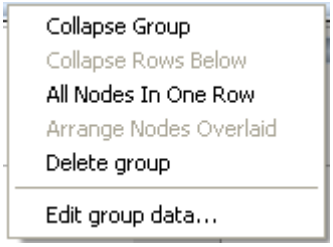
5.29 Context Menu of Links

If you click the right mouse key on a link, the **Delete Link** context menu will appear. To delete the marked link, please click the left mouse key to confirm.

The event **OnLinkRClickCltn** occurs when the user clicks the right mouse key on a link or on several overlapping links. The LinkCollection object and the mouse position (x,y-coordinates) are captured and passed, so that you can display your own context menu at the appropriate position. If you set the returnStatus to **vcRetStatNoPopup**, the integrated context menu will be revoked.

5.30 Context Menu of Groups

If you right-click on a group title in the table or a group layer in the diagram (which will only be displayed if in the **Grouping** dialog the checkbox **Group node visible** has been ticked), a context menu will appear that offers basic options on groups:



The event **OnGroupRClick** occurs when the user clicks the right mouse key on a group heading in the table. The group object and the mouse position (x,y-coordinates) are returned, so that you can display a context menu at the appropriate position. If you set the returnStatus to **vcRetStatNoPopup**, the integrated context menu will be revoked.

Collapse/Expand Group

This menu item lets you expand a collapsed group or collapse an expanded one.

Expand/Collapse Rows Below

This menu item lets you expand the rows of a collapsed group or collapse the rows below an expanded group respectively.

If you have chosen for the group **All Nodes In One Row**, this option will collapse only the subgroups of the selected group.

All Nodes In One Row/Nodes In Separate Rows

If you choose the option **All Nodes In One Row** all activities in a group will be displayed in one row. If the activities in the group coincide, they will be automatically displayed underneath one another in expanded mode to prevent overlapping. If the group is collapsed, the activities may overlap.

With this type of arrangement, the table section for the activities is suppressed, so you will need to utilise the layer annotation or tooltip to identify the activities for the user.

The option **Nodes In Separate Rows** lets you display each node in its own row.

Arrange Nodes Optimized/Arrange Nodes Overlaid

*(Selectable only, if **All Nodes In One Row** was selected.)*

If you select **Arange Nodes Overlaid**, the nodes are displayed in one row, even if they are overlapping each other.

If you select **Arrange Nodes Optimized**, the layout of the nodes will be optimized to avoid overlapping, even if they require more space than a single row.

Delete group

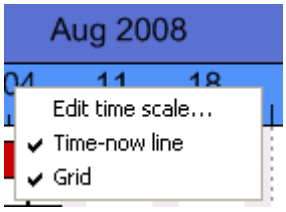
This menu item lets you delete an empty marked group.

Edit group data

The corresponding dialog will appear.

5.31 Context Menu of the Time scale

If you click the right mouse key on the time scale, the below menu will appear:



The event **OnTimeScaleRClick** occurs when the user clicks the right mouse key on the time scale. The TimeScale object and the mouse position (x,y-coordinates) are returned. At this position you can show your customized context menu. If you set the returnStatus to vcRetStatNoPopup, the integrated context menu will be revoked.

Edit Time scale

Select this option to reach the **Edit Time scale** dialog box.

Timenow Line

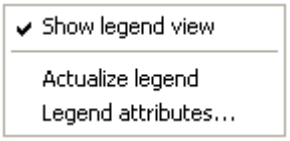
Specify whether your diagram should contain a timenow line (date line).

Grid

Specify whether your diagram should contain grid lines.

5.32 Context Menu of the Legend

A right mouse button click on the legend will open the below menu:



Show legend view

This menu item lets you switch on or off the legend view.

Actualize legend

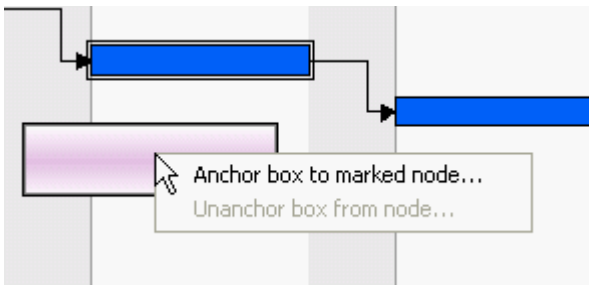
This menu item lets you refreshing the legend which is needed after modifications in the chart, such as adding or deleting nodes, because they are not displayed automatically in the legend. The refreshing can also be carried out by switching off and on the legend view. This concerns the loading of nodes as well. If on the property page **Additional views** the attribute **Initially visible** was selected for the legend view and no nodes have been loaded when running the program, the legend stays empty until it was refreshed.

Legend attributes

With this item you open the corresponding dialog where you can specify the settings concerning legend title, legend elements and margins. For further information about this dialog please see chapter 4.44 "The Legend Attributes Dialog Box".

5.33 Context Menu of Boxes

A right mouse click on a box will open the below menu:



If the context menu does not pop up, you have to activate the option **Show context menu for the box** on the **General** property page.

Anchor box to marked node

This item lets you anchor a box to the marked node. This is only possible if you have selected the option **Anchoring interactions allowed** in the **Administrate boxes** dialog.

Unanchor box from node

This item lets you anchor a box to the marked node.

6 Frequently Asked Questions

6.1 How can I Activate the License File?

6.2 What can I do if Problems Occur during Licensing?

When you license a module for the first time or when you continue an expired license, please open the **Licensing** dialog box which you reach via the **General** property page. Click on the **Request** button. Then the **Request License Information** dialog will open.

Enter your license number, your name and the name of your company and click on **Send email to NETRONIC**. An email to NETRONIC will be generated automatically. As soon as we have received it, we will generate your license information file (vcgantt.lic) and send it back to you. After having received this file, please copy it to the directory in which the file **vcgantt.ocx** is stored.

After licensing, you need to activate the new license. Please open a property page and make the system store it by making some change. This will activate the new license.

If during licensing of the VARCHART ActiveX control you receive an error message "REGSVR32 Error Return: 0X0000007e", the file *vcwin32u.dll* does not exist or is not stored in a directory indicated in the PATH. If the file does not exist, please contact the support of NETRONIC Software GmbH.

6.3 How can I Make the VARCHART ActiveX Control Use a Modified .INI File?

Some of the VARCHART ActiveX control's settings cannot be modified on the property pages. Still, you can adjust them via the *.ini file:

1. Open the **General** property page. The **Configuration file** field shows the current configuration file (for example *project.ini*).
2. Click on the **Browse** button. The dialog **Load/Save** will open. Please enter a file name into the **Temporary data file** field to be used as a temporary dummy configuration file, such as *dummy.ini*. Click on **Save**.
3. Now click on the **OK** or **Apply** button of the **General** property page. The configuration file *dummy.ini* will automatically be generated and applied.
4. Now you can edit your *.ini file (e.g. *project.ini*) in a text file editor and save your changes.
5. Then reset the true configuration file by selecting the former file (*project.ini*) on the **General** property page in the **Configuration file** field and click on **OK**. Your modified *.ini file is being used from now on.

6.4 What Borland Delphi Users Need to do on Upgrading a New VARCHART XGantt Version.

After the upgrade or update of the VARCHART XGantt to a higher version it is necessary to install the new version to the Delphi Package Borland User Components. Please proceed as described below:

1. Start Borland Delphi.
2. Click onto **Components** and **ActiveX import**.
3. Select *NETRONIC VARCHART XGantt* from the ActiveX Controls list and click onto the **Remove** button to remove the registration. Quit the dialog by **Cancel**.
4. Now open the **Components** > **Install packages** dialog. Select the package *Borland User Components*. (This package is stored in the file *dclusr*0.bpl*. The '*' in the file name depends on your Delphi version: 5, 6 or 7.) .
5. Click on **Edit**. The file *dclusrX0.dpk* will open.
6. Select *VcGanttLib_TLB.pas* and *VcGanttLib_TLB.dcr* succeedingly and remove them from the project by clicking the right mouse button.
7. Compile the package and close the dialog. This way the changes will be saved to the project *dclusrX0*.
8. Now re-open the dialog **Components** > **ActiveX import**.
9. Click on **Add**, select *vcgantt.ocx*, and click on **Open**. Now *NETRONIC VARCHART XGantt* re-appears in the list of the registrated ActiveX controls.
10. Click on **Install...** to re-compile the package *dclusrX0.bpl*.
11. Close the dialog to save the project *dclusrX0*.

6.5 How can I Activate the XP Visual Style in VARCHART XGantt?

The XP Visual Style is dependent on the used version of the common controls of windows. In Windows XP the versions 5 and 6 are included in delivery but due to reasons of compatibility, version 5 is used by default. The switching over of an application to version 6 is done by an entry in a so-called manifest which is supported by Windows from XP onwards. This manifest can either be compiled in a project as a resource with ID 24 resp. RT_MANIFEST or stored as file in the same directory as the EXE file (the name is then made up of the name of the EXE file + ".manifest", e.g. notepad.exe.manifest). The contents of the manifest is in XML format and looks as follows:

Example Code

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<assembly xmlns="urn:schemas-microsoft-com:asm.v1"
manifestVersion="1.0">
   <assemblyIdentity version="1.0.0.0" processorArchitecture="X86"
                     name="appname.exe" type="win32" />
   <description>Enter application description here</description>
   <dependency>
      <dependentAssembly>
         <assemblyIdentity type="win32" name="Microsoft.Windows.Common-
Controls" version="6.0.0.0"
                           processorArchitecture="X86"
publicKeyToken="6595b64144ccf1df"
                           language="*" />
      </dependentAssembly>
   </dependency>
</assembly>
```

You should make sure that the Win 32 API command **InitCommonControls** is invoked at the beginning of the application. Otherwise it may happen that the old Common controls of version 5.0 are loaded before the manifest is interpreted. In VB 6 e.g. you can set the command in the event **Form_Initialize** of the start form. Before doing so, the below code line is required:

Example Code

```
Private Declare Sub InitCommonControls Lib "comctl32.dll" ().
```

The manifest itself should be customized to the name and version of the application (<assemblyIdentity> and <description>).

For further information look on the following Internet pages:

http://www.activevb.de/tutorials/tut_xpstyles/xpstyles.html

http://support.microsoft.com/?id=309366

http://support.microsoft.com/?id=303636

6.6 What to do if the Control Does Not Work With a User Account of a Computer

If you find that the control does not react when two users invoke the same application that uses the control, the reason for this may be that the control was not installed for both users. When generating the setup program by which the control is installed on the computer of your customer, the option "install for all users" needs to be selected.

An installation for several users can be activated at a later time by extending the safety settings of the files that belong to the control, allowing different accounts to access the files. The safety settings you can modify by the menu item "properties" of the context menu of the affected file or by the command line using the command 'cacls'. You can find a list of the files that belong to the control in the chapter "Shipping the Application" at the beginning of this book.

6.7 How can I Limit the Timescale Width?

If you touch the timescale on the extreme left side of the visible area keeping the left mouse button pressed to widen the timescale, you can easily reach a factor far in excess of 1000%. To control this, use the **OnTimeScaleSectionRescale** event. The below example shows how to allow for a twofold enlargement at maximum.

Example Code

```
Private Sub VcGantt1_OnTimeScaleSectionRescale(ByVal timeScale As _
               VcGanttLib.VcTimeScale, ByVal sectionIndex As _
               Integer, ByVal newBasicUnitWidth As Long, _
               returnStatus As Variant)

    Dim nOldUnitWidth As Long

    nOldUnitWidth = timeScale.Section(sectionIndex).UnitWidth

    If newBasicUnitWidth > (2 * nOldUnitWidth) Then
        timeScale.Section(sectionIndex).UnitWidth = 2 * nOldUnitWidth
        returnStatus = vcRetStatFalse
    End If

End Sub
```

6.8 How can I Move a Bar into the Visible Area by Clicking on the Table?

The event **OnNodeLClick** captures both the node and the information **InTable** or **InDiagram**. If the table was clicked on (**InTable**), the relevant date of the node is retrieved and transferred to the VARCHART ActiveX object using the **ScrollToDate** method.

Example Code

```
Private Sub VcGantt1_OnNodeLClick(ByVal node As VcGanttLib.VcNode, _
                  ByVal location As VcGanttLib.LocationEnum, _
                  ByVal x As Long, ByVal y As Long, _
                  returnStatus As Variant)
   Dim myDataDef As VcDataDefinition
   Dim myDataDefTable As VcDataDefinitionTable
   Dim myDataField As VcDefinitionField
   Dim myIndex As Integer

   If location = vcInTable Then
      ' if the index of the "Start" field is not known
       Set myDataDef = VcGantt1.DataDefinition
       Set myDataDefTable = myDataDef.DefinitionTable(vcMaindata)
       Set myDataField = myDataDefTable.FieldByName("Start")
       myIndex = myDataField.ID
       VcGantt1.ScrollToDate node.DataField(myIndex), vcLeftAligned, 2
   End If
End Sub
```

6.9 How can I Make Overlapping Activities in a Group Visible?

To avoid bottlenecks in holiday rosters or machine allocations, overlapping activities in a group can be made visible.

Activities can overlap if the activities have been grouped and in the **Grouping** dialog the **Nodes in separate rows** option is **not**selected. With the **Nodes in separate rows** option, the activity groups can be collapsed and expanded. When a group is collapsed, overlapping activities cannot be detected. When a group is expanded, the activities are staggered so that overlapping activities become apparent.

To make overlapping activities in a group visible, deactivate the **Nodes in separate rows** option in the **Grouping** dialog to display the activities of a group in one line. If the activities of a group overlap, they will be displayed in different lines even when the option is deactivated allowing you to see any collisions at a glance.

When the activities are collapsed, overlapping activities cannot be detected. Therefore you should deactivate the **Modifications allowed** option to prohibit the user from switching between these two types of display. When the **Initially collapsed** option is *not* activated, the groups will be displayed in their expanded states, i.e. overlapping activities can be instantly recognised as they are displayed beneath each other in separate lines.

6.10 How can I Save and Reload theOrder of Activities?

On condition that the activities are loaded from a file, you can save and reload the activities.

In order to save and reload the order of activities, open the **Sorting** property page and select a data field from **Row number field**. The VARCHART ActiveX control will store the identification to this data field. If the order of the nodes was modified interactively, you can update it using the method **UpdateRowNumberField**. Groups and hierarchy must not be activated at that time.

Finally, please add the following code:

Example Code

```
Private Sub Form_Unload ()
   VcGantt1.UpdateRowNumberField
   VcGantt1.SaveAs (" ")
End Sub
```

6.11 Why can I not Create Nodes Interactively at Times?

If you cannot create nodes with the mouse at runtime, please tick the check box **Allow new nodes** on the **Nodes** property page.

If in addition you tick **New nodes via double-click** you can generate nodes by double-clicking on the mouse.

Beside, if a calendar is activated, nodes cannot be generated in workfree periods.

Check if the property **AllowNewNodes** has not been set to **False**.

6.12 How can I Disable the Default Context Menus?

You can disable a predefined context menu to occur by setting the returnStatus to **vcRetStatNoPopup**.

Example Code

```
'switching off the context menu of diagram
Private Sub VcGantt1_OnDiagramRClick(ByVal x As Long, _
                                ByVal y As Long, returnStatus As
Variant)
    returnStatus = vcRetStatNoPopup
End Sub

'switching off the context menu of links
Private Sub VcGantt1_OnLinkRClickCltn(ByVal linkCltn As _
                                VcGanttLib.VcLinkCollection, _
                                ByVal x As Long, _
                                ByVal y As Long, _
                                returnStatus As Variant)
    returnStatus = vcRetStatNoPopup
End Sub

'switching off the context menu of nodes
Private Sub VcGantt1_OnNodeRClick(ByVal node As VcGanttLib.VcNode, _
                                ByVal location As
VcGanttLib.LocationEnum, _
                                ByVal x As Long, ByVal y As Long, _
                                returnStatus As Variant)
   returnStatus = vcRetStatNoPopup
End Sub
```

6.13 What can I do if Problems Occur during Printing?

If printing of your diagram is impossible or if you cannot set up the printer, please verify whether the file *vcprct32.dll* exists. Also, please verify if the file can be located by the PATH settings, and if the Windows default printer has been set up.

If the file *vcprct32.dll* does not exist, please contact the support of NETRONIC Software GmbH.

6.14 How can I Improve the Performance?

> SuspendUpdate

Projects that include a large number of nodes may take too long if updating actions are repeated for each node. Not every automatic update procedure is necessary; in those cases you can suspend single updates, work off a sequence of code and then do a final update. Suspending and re-activating updates both can be done by the method **SuspendUpdate**, which is set to **True** at the beginning of the code sequence and to **False** at its end. Using this method can im improve the overall performance considerably.

Example Code

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

Set dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
Set dataRecordCltn = dataTable.DataRecordCollection

VcGantt1.SuspendUpdate (True)

For Each dataRecord In dataRecordCltn
    dataRecord.DataField(4) = "10"
    dataRecord.UpdateDataRecord
Next

VcGantt1.SuspendUpdate (False)
```

You can also accelerate the updating procedure of links via the **Suspend-Update** method.

If you modify table formats in large projects, you also should use the **SuspendUpdate** method.

Example Code

```
Private Sub ModifyTable_Click()
    Dim formatCol As VcTableFormatCollection
    Dim aFormat As VcTableFormat
    Dim index As Integer
    VcGantt1.SuspendUpdate True
    Set formatCol = VcGantt1.Table.TableFormatCollection
    For Each aFormat In formatCol
        For index = 1 To aFormat.NoOfColumns
            aFormat.FieldBackgroundColor(index) = vbGreen
            aFormat.FieldFontBody(index) = vcBold
            aFormat.FieldFontColor(index) = vbRed
            aFormat.FieldFontName(index) = "Arial"
            aFormat.FieldFontSize(index) = 14
            aFormat.FieldHorAlignment(index) = vcHorCenterAligned
        Next
    Next
    VcGantt1.SuspendUpdate False
End Sub
```

This method also accelerates the updating procedure when you use not equidistant histogram curves.

Example Code

```
Private Sub CommandCreateCurve_Click()

Dim myCurve As VcCurve
Set myCurve =
VcGantt1.HistogramCollection.FirstHistogram.CurveCollection. _
              CurveByName("Curve1")
Dim index As Integer
Dim aDate As Date

'current date
aDate = Date
VcGantt1.SuspendUpdate True

For index = 1 To 3000
' move: 2h:24min
  aDate = aDate + 1 / 10
  myCurve.SetValues aDate, index
Next
VcGantt1.SuspendUpdate False

End Sub
```

The method also can accelerate the updating procedure when you use calendars because modifications of the calendars need a lot of time when the nodes have been loaded since then for all nodes the program has to check if they depend on a calendar.

> Graphics

Another reason for a low performance may be graphics in table, node or box fields that are too large or that have to many pixels.

6.15 Error Messages

> Error messages at runtime caused by the developer

Error Reason	Message
License failure	This is an unlicensed version of *. Please contact NETRONIC for a licensed version.
	The licensing failed. Please contact NETRONIC.
	The expiry date is exceeded. Please contact NETRONIC.
	Your identification has changed from * to *. Please contact NETRONIC!
	The ActiveX Control * used in this program has no runtime license!
ActiveX installation incomplete or older versions of a DLL in the system path	DLL * not found
	Loading the interface with identifier * failed
	The interface DLL (version *) is too old. This program needs version * or above.
Program installation incomplete or absolute path is erroneous	Group titles file not found
	The file * is not a valid graphics file.
	Graphics file not specified or not existent.
Error at assignment of a new INI file	The configuration file * was not found, program creates it using the default configuration.
INI file has errors	The highlight/table/layer * uses the non-existent filter *. The filter entry is corrected to <always>.
	The highlight/table * uses the non-existent node annotation *. The node annotation entry is corrected to *.
	Layer name * is not unique. Please check the configuration file.
	Highlight * non-existent
	The name * for link appearance is not unique. Please check the configuration file(s).
	Your configuration file * is corrupt. [*] must be unique.

> Error messages at runtime caused by the end user or by the developer

Error Reason	Message
Cycles detected in the method **ScheduleProject**	Project has cycled links!
Interactive moving of nodes	Cannot create new groups

6.16 Can All Fonts be Used?

Due to the support of GDI+ there are some cutbacks in terms of font display. GDI+ is unable to display postscript and bitmap fonts. The first group includes fonts that may be of the type **OpenType**, but being "classical fonts" they have some sort of internal postscript structure, such as "Warnock Pro". The second group includes the early Windows fonts "Courier", "Times", "System" and "MS Sans Serif".

For this reason, the above fonts are not offered by the font selection dialogs of VARCHART XGantt. If you set them via the API, an alternative font will be displayed. In terms of the early fonts, NETRONIC has put up a replacement rule that selects a similar "late" font; external fonts are replaced by "Arial" to ensure a display at all.

Probably or probably not future versions of GDI+ will support the fonts presently not supported. Unfortunately, more information on this subject can only be obtained in blogs and news groups, but not at MSDN.

7 API Reference

7.1 Object types

- DataObject
- DataObjectFiles
- VcBorderArea
- VcBorderBox
- VcBox
- VcBoxCollection
- VcBoxFormat
- VcBoxFormatCollection
- VcBoxFormatField
- VcCalendar
- VcCalendarCollection
- VcCalendarGrid
- VcCalendarGridCollection
- VcCalendarProfile
- VcCalendarProfileCollection
- VcCurve
- VcCurveCollection
- VcDataDefinition
- VcDataDefinitionTable
- VcDataDefinitionTable
- VcDataRecord
- VcDataRecordCollection
- VcDataTable
- VcDataTableCollection
- VcDataTableField
- VcDataTableFieldCollection
- VcDateLine
- VcDateLineCollection
- VcDateLineGrid
- VcDateLineGridCollection
- VcDefinitionField
- VcField

- VcFilter
- VcFilterCollection
- VcFilterSubCondition
- VcGantt
- VcGroup
- VcGroupCollection
- VcGroupLevelLayout
- VcGroupLevelLayoutCollection
- VcHierarchyLevelLayout
- VcHistogram
- VcHistogramCollection
- VcInfoWindow
- VcInterval
- VcIntervalCollection
- VcLayer
- VcLayerCollection
- VcLayerFormat
- VcLayerFormatField
- VcLegendView
- VcLineFormat
- VcLineFormatCollection
- VcLineFormatField
- VcLink
- VcLinkAppearance
- VcLinkAppearanceCollection
- VcLinkCollection
- VcMap
- VcMapCollection
- VcMapEntry
- VcNode
- VcNodeCollection
- VcNodeLevelLayout
- VcNumericScale
- VcNumericScaleCollection
- VcPrinter
- VcRect
- VcResourceScheduler2
- VcRibbon
- VcScheduler
- VcSection

- VcTable
- VcTableCollection
- VcTableFormat
- VcTableFormatCollection
- VcTableFormatField
- VcTimeScale
- VcTimeScaleCollection
- VcUpdateBehavior
- VcUpdateBehaviorCollection
- VcUpdateBehaviorContext
- VcWorldView

7.2 DataObject

DataObject

The OLE Drag & Drop technique allows to move selected nodes from an activeX source control to a target control. The container to transfer the corresponding data is the object **DataObject**. The object provides appropriate properties for the transfer: **Files**, **Clear**, **GetData**, **GetFormat** and **SetData**.

You can also exchange data with other controls capable of OLE-Drag&Drop. When doing so, please keep in mind that VARCHART-ActiveX controls store and interpret data in the CSV text format.

To make OLE Drag & Drop work, in the properties window the properties **OLEDragMode** and **OLEDropMode** need to be activated. On the **Nodes** property page by the option **Move all selected nodes** you can select whether just a single node or several marked nodes can be moved.

Please find detailed information in the chapter **Important Concepts** in the section **OLE-Drag&Drop**.

Properties

- DropEndDate
- DropStartDate
- Files

Methods

- Clear
- GetData
- GetFormat
- SetData

Properties

DropEndDate

Read Only Property of DataObject

This property indicates the end date of the dropping operation. If **OLEDrop-Mode** was set to **vcOLEDropManual**, this property can be used to retrieve the end date of the phantom in order to pass it on to a newly created node.

	Data Type	Explanation
Property value	Date	End date

DropStartDate

Read Only Property of DataObject

This property indicates the start date of the dropping operation. If **OLEDropMode** was set to **vcOLEDropManual**, this property can be used to retrieve the start date of the phantom in order to pass it on to a newly created node.

	Data Type	Explanation
Property value	Date	Start date

Files

Read Only Property of DataObject

This property returns a DataObjectFiles collection, which in turn contains a list of all file names used by a DataObject object (such as the names of files that a user drags to or from the Windows File Explorer.) This property can only be used if the DataObject contains Data of format **15** (**list of files**, please see property **GetFormat**).

	Data Type	Explanation
Property value	DataObjectFiles	List of available files

Methods

Clear

Method of DataObject

This method deletes the contents of the DataObject object. This method is available to drag operations only, i. e. **OLEStartDrag**, **OLESetData**, **OLEGiveFeedback** and **OLECompleteDrag**.

	Data Type	Explanation
Return value	Void	

GetData

Method of DataObject

This method returns data from a DataObject in the shape of the data type **Variant** and is available only to DataObject objects of the events **OLEDragOver** and **OLEDragDrop**.

It is possible for the **GetData** method to use data formats other than those listed below, including user-defined formats registered with Windows by the **RegisterClipboardFormat()** API function. However, there are a few caveats:

The **GetData** method always returns data in a byte array if it is in a format that it cannot recognize.

The byte array returned by **GetData** may be larger than the actual data, with arbitrary bytes at the end of the array. The reason for this is that VARCHART ActiveX does not know the format of the data, but merely has knowledge of the size of memory allocated for the data ba the operating system. The allocated size of memory often is larger than the one actually required for the data. Therefore, there may be an excess of bytes at the end of the allocated memory segment. As a result, you are supposed to use appropriate functions to interpret the data in a meaningful way (in Visual Basic e.g. truncating a string at a particular length by the **Left** function if the data is in a text format).

Note: Not all applications support the formats **2** (bitmap) or **9** (color palette), so it is recommended that you use **8** (device-independent bitmap) whenever possible.

	Data Type	Explanation
Parameter:		
⇨ format	Integer	Identification number of the format (plus examples from Visual Basic and C): 1 - text in ANSI-code (.txt files) VB: vcCFText; C: CF_TEXT 2 - bitmap (.bmp-files) VB: vbCFBitmap; C: CF_BITMAP 3 - metafile (.wmf-files) VB: vbCFMETAFILE; C: CF_MetaFile 8 - device-independent Bitmap (DIB) VB: vbCFDIB; C: CF_DIB 9 - color palette VB: vbCFPalette; C: CF_PALETTE 13 - text in unicode code (.txt-Dateien) VB: 13; C: CF_UNICODETEXT 14 - enhanced Metafile (.emf-files) VB: vbCFEMetaFile; C: CF_EMETAFILE 15 - list of files VB: vbCFFiles; C: CF_FILES -16639 - rich text format (.rtf files) VB: vbCFRTF; C: CF_RTF
	Possible Values:	Data field index
Return value	Variant	Data retrieved

GetFormat

Method of DataObject

This method returns a boolean value indicating whether data in the DataObject object match a specified format. It is available only to DataObject objects of the events **OLEDragOver** and **OLEDragDrop**.

	Data Type	Explanation
Parameter:		
⇨ format	Integer	Identification number of the format (plus examples from Visual Basic and C): 1 - text in ANSI code (.txt files) VB: vcCFText; C: CF_TEXT 2 - bitmap (.bmp-files) VB: vbCFBitmap; C: CF_BITMAP 3 - metafile (.wmf-files) VB: vbCFMETAFILE; C: CF_MetaFile 8 - device-independent Bitmap (DIB) VB: vbCFDIB; C: CF_DIB 9 - color palette VB: vbCFPalette; C: CF_PALETTE 13 - text in unicode code (.txt-Dateien) VB: 13; C: CF_UNICODETEXT 14 - enhanced Metafile (.emf-files) VB: vbCFEMetaFile; C: CF_EMETAFILE 15 - list of files VB: vbCFFiles; C: CF_FILES -16639 - rich text format (.rtf files) VB: vbCFRTF; C: CF_RTF
	Possible Values:	Data field index
Return value	Boolean	The **GetFormat** method returns **True** if an item in the DataObject object matches the specified format. Otherwise, it returns **False**.

SetData

Method of DataObject

This method inserts data into a DataObject using the specified data format. It is available only to DataObject objects of the events **OLEStartDrag**, **OLE-SetData**, **OLEGiveFeedback** and **OLECompleteDrag**.

It is possible for the **SetData** method to use data formats other than those listed below **format**, including user-defined formats registered with Windows by the **RegisterClipboardFormat()** API function. However, there are a few caveats:

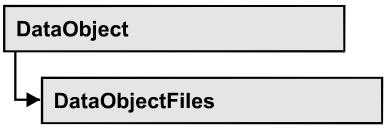
The **SetData** method requires the data to be in the form of a byte array if the data format specified could not be recognized.

Not all applications support **2** (bitmap) or **9** (palette), so it is recommended that you use **8** (device-independent bitmap) whenever possible.

	Data Type	Explanation
Parameter:		
⇨ data	Variant	Data to be set or **Empty** if you wish to transmit the format to be set on request by the event **OLESetData**.
⇨ format	Integer	Identification number of the format (plus examples from Visual Basic and C): 1 - text in ANSI code (.txt files) VB: vcCFText ; C: CF_TEXT 2 - bitmap (.bmp-files) VB: vbCFBitmap; C: CF_BITMAP 3 - metafile (.wmf-files) VB: vbCFMETAFILE; C: CF_MetaFile 8 - device-independent Bitmap (DIB) VB: vbCFDIB; C: CF_DIB 9 - color palette VB: vbCFPalette; C: CF_PALETTE 13 - text in unicode code (.txt-Dateien) VB: 13; C: CF_UNICODETEXT 14 - enhanced Metafile (.emf-files) VB: vbCFEMetaFile; C: CF_EMETAFILE 15 - list of files VB: vbCFFiles; C: CF_FILES -16639 - rich text format (.rtf files) VB: vbCFRTF; C: CF_RTF
	Possible Values:	Data field index

Return value	Void	

7.3 DataObjectFiles



This object keeps a list of all file names, that are stored in a DataObject, if it contains data of format **15** (list of files). By **For Each Item in DataObject-Files** you can access all file names in a loop.

Properties

- _NewEnum
- Count
- Item

Methods

- Add
- Clear
- Remove

Properties

_NewEnum

Read Only Property of DataObjectFiles

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all data object files. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method **GetEnumerator** is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Private Sub VcGantt1_OLEDragOver(ByVal data As VcGanttLib.DataObject, effect As
Long, ByVal button As Integer, ByVal Shift As Integer, ByVal x As Long, ByVal y
As Long, ByVal state As VcGanttLib.OLEDragStateEnum)

Dim fileName as String
```

```
For Each fileName In DataObject.DataObjectFiles
   Debug.Print fileName
Next

End Sub
```

Count

Read Only Property of DataObjectFiles

This property returns the number of file names available in the list.

	Data Type	Explanation
Property value	Long	Number of files

Item

Property of DataObjectFiles

By this property you can assign or retrieve a file name by the index passed. Because this is the default property of the object, in many programming environments (e.g. Visaul Basic) the property name can be dropped. Example: DataObjectFiles(0) will return the first file name.

	Data Type	Explanation
Parameter: ⇨ index	Long	Index of the file name {0...Count-1}
Property value	String **Possible Values:**	File name Name of the color map

Methods

Add

Method of DataObjectFiles

This method lets you add the file name specified to the list of file names. If an index (Integer, values: 0 to .Count-1) is specified, the file name will be inserted at the specified position. Otherwise it will be inserted at the end of the list.

	Data Type	Explanation
Parameter:		
⇨ index	Variant	Index of the position in the list that the file name is to be inserted at (optional)
⇨ fileName	String	Name of the file
	Possible Values:	Name of the color map
Return value	Void	

Clear

Method of DataObjectFiles

This method lets you delete all file names available in the list.

	Data Type	Explanation
Return value	Void	

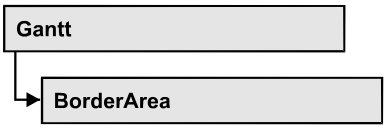
Remove

Method of DataObjectFiles

This method lets you remove the file name with the specified index (values: 0 to .Count-1).

	Data Type	Explanation
Parameter:		
⇨ index	Long	Index of the position in the list that the file name is to be removed from.
Return value	Void	

7.4 VcBorderArea



An object of the type **VcBorderArea** designates the title or legend area of the graphics.

Methods

- BorderBox

Methods

BorderBox

Method of VcBorderArea

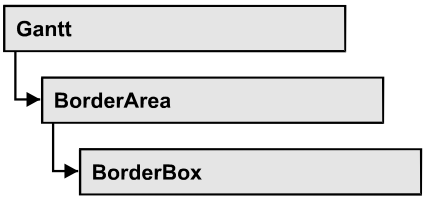
This method gives access to a BorderBox object.

	Data Type	Explanation
Parameter:		
boxPosition	BorderBoxPositionEnum	Box position
	Possible Values:	
	vcBBXPBottomBottomCentered 8	second line in the bottom area, centered
	vcBBXPBottomBottomLeft 7	second line in the bottom area, left
	vcBBXPBottomBottomRight 9	second line in the bottom area, right
	vcBBXPBottomTopCentered 5	first line in the bottom area, centered
	vcBBXPBottomTopLeft 4	first line in the bottom area, left
	vcBBXPBottomTopRight 6	first line in the bottom area, right
	vcBBXPLegend 51	legend
	vcBBXPTopCentered 2	top centered
	vcBBXPTopLeft 1	top left
	vcBBXPTopRight 3	top right
Return value	VcBorderBox	Box of the title and legend area

Example Code

```
Dim borderArea As VcBorderArea
Dim bBoxBBL As VcBorderBox

Set borderArea = VcGantt1.BorderArea
Set bBoxBBL = borderArea.BorderBox(vcBBXPBottomBottomLeft)
bBoxBBL.LegendTitle = "Explanation"
```

7.5 VcBorderBox



An object of the type **VcBorderBox** designates one of the boxes in the title or legend area of the graphics.

Properties

- Alignment
- GraphicsFileName
- LegendElementsArrangement
- LegendElementsBottomMargin
- LegendElementsMaximumColumnCount
- LegendElementsMaximumRowCount
- LegendElementsTopMargin
- LegendFont
- LegendTitle
- LegendTitleFont
- LegendTitleVisible
- Text
- TextFont
- Type

Properties

Alignment

Property of VcBorderBox

This property lets you set or retrieve the alignment of this BorderBox object.

	Data Type	Explanation
Property value	BorderBoxAlignmentEnum	Alignment of the border box
	Possible Values: vcBBXACentered -1 vcBBXALeft -3	 Center Left

vcBBXARight -2	Right

GraphicsFileName

Property of VcBorderBox

This property lets you set or retrieve the name of the graphics file used in the VcBorderBox object. *Available formats:*

- *.BMP (Microsoft Windows Bitmap)
- *.EMF (Enhanced Metafile or Enhanced Metafile Plus)
- *.GIF (Graphics Interchange Format)
- *.JPG (Joint Photographic Experts Group)
- *.PNG (Portable Network Graphics)
- *.TIF (Tagged Image File Format)
- *.VMF (Viewer Metafile)
- *.WMF (Microsoft Windows Metafile)
- *.WMF, with EMF included

EMF, EMF+, VMF and WMF are vector formats that allow to store a file independent of pixel resolution. All other formats are pixel-oriented and confined to a limited resolution.

The VMF format basically has been deprecated, but it will still be supported for some time to maintain compatibility with existing applications.

	Data Type	Explanation
Property value	String	Name of the graphics file
	Possible Values:	Name of the color map

Example Code

```
Dim borderArea As VcBorderArea
Dim bBoxTR As VcBorderBox
```

```
Set borderArea = VcGantt1.BorderArea
Set bBoxTR = borderArea.BorderBox(vcBBXPTopRight)
bBoxTR.Type = vcBBXTGraphics
bBoxTR.GraphicsFilename = "Asterix.jpg"
```

LegendElementsArrangement

Property of VcBorderBox

This property lets you set or retrieve the arrangement of the elements in the legend.

	Data Type	Explanation
Property value	LegendElementsArrangementEnum	Type of arrangement of the legend elements
	Possible Values:	
	vcLEAFixedToColumns 1	The legend elements are merely aligned along columns.
	vcLEAFixedToRows 0	The legend elements are merely aligned along rows.
	vcLEAFixedToRowsAndColumns 2	The legend elements are aligned along rows and columns.

LegendElementsBottomMargin

Property of VcBorderBox

This property lets you set or retrieve the width between the legend elements and the bottom of the border box (unit: mm).

	Data Type	Explanation
Property value	Integer	Width of bottom margin
	Possible Values:	Data field index

LegendElementsMaximumColumnCount

Property of VcBorderBox

This property lets you set or retrieve the number of columns to which the elements in the legend should disperse.

	Data Type	Explanation
Property value	Integer **Possible Values:**	Number of columns Data field index

LegendElementsMaximumRowCount

Property of VcBorderBox

This property lets you set or retrieve the number of rows to which the elements in the legend should disperse.

	Data Type	Explanation
Property value	Integer **Possible Values:**	Number of rows Data field index

LegendElementsTopMargin

Property of VcBorderBox

This property lets you set or retrieve the width between the legend elements and the top of the border box (unit: mm).

	Data Type	Explanation
Property value	Integer **Possible Values:**	Width of top margin Data field index

LegendFont

Property of VcBorderBox

This property lets you set or retrieve the font attributes of the legend.

	Data Type	Explanation
Property value	StdFont	Font attributes of the legend

Example Code

```
Dim borderArea As VcBorderArea
Dim bBoxBBL As VcBorderBox

Set borderArea = VcGantt1.BorderArea
Set bBoxBBL = borderArea.BorderBox(vcBBXPBottomBottomLeft)
bBoxBBL.Type = vcBBXTLegend
logThis (bBoxBBL.LegendFont.Name)
```

LegendTitle

Property of VcBorderBox

This property lets you set or retrieve the legend title.

	Data Type	Explanation
Property value	String **Possible Values:**	Legend title Name of the color map

Example Code

```
Dim borderArea As VcBorderArea
Dim bBoxBBL As VcBorderBox

Set borderArea = VcGantt1.BorderArea
Set bBoxBBL = borderArea.BorderBox(vcBBXPBottomBottomLeft)
bBoxBBL.LegendTitle = "Explanation"
```

LegendTitleFont

Property of VcBorderBox

This property lets you set or retrieve the font attributes of the legend title.

	Data Type	Explanation
Property value	StdFont	Font attributes of the legend title

Example Code

```
Dim borderArea As VcBorderArea
Dim bBoxBBL As VcBorderBox

Set borderArea = VcGantt1.BorderArea
Set bBoxBBL = borderArea.BorderBox(vcBBXPBottomBottomLeft)
bBoxBBL.Type = vcBBXTLegend
logThis (bBoxBBL.LegendTitleFont.Name)
```

LegendTitleVisible

Property of VcBorderBox

This property lets you set or retrieve whether the legend title is visible.

	Data Type	Explanation
Property value	Boolean	Legend title visible (True)/ not visible (False)
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
Dim borderArea As VcBorderArea
Dim bBoxBBL As VcBorderBox

Set borderArea = VcGantt1.BorderArea
Set bBoxBBL = borderArea.BorderBox(vcBBXPBottomBottomLeft)
bBoxBBL.LegendTitleVisible = False
```

Text

Property of VcBorderBox

This property lets you set or retrieve the text of a head line (above or below the diagram). For numbering the pages or displaying the system date you may enter the below wild cards which will be replaced by the appropriate contents on the printout:

{COLUMN} = page number wide (of a two-dimensional page layout)

{NUMPAGES} = total number of pages

{PAGE} = consecutive numbering of pages

{ROW} = page number high (of a two-dimensional page layout)

{SYSTEMDATE} = system date

	Data Type	Explanation
Parameter:		
rowIndex	Integer	row index {0...6}
	Possible Values:	Data field index
Property value	String	text in text boxes

Possible Values:	Name of the color map

Example Code

```
Dim borderArea As VcBorderArea
Dim bBoxBBL As VcBorderBox

Set borderArea = VcGantt1.BorderArea
Set bBoxBBL = borderArea.BorderBox(vcBBXPBottomBottomLeft)
bBoxBBL.Type = vcBBXTText
bBoxBBL.Text(index) = "Department A"
```

TextFont

Property of VcBorderBox

This property lets you set or retrieve the font attributes of a title line (above or below the diagram).

This property is an indexed property, which in C# is referred to by one of the methods **set_TextFont (rowIndex, pvn)** and **get_TextFont (row-Index)**.

	Data Type	Explanation
Parameter:		
rowIndex	Integer	Row index {0...6}
	Possible Values:	Data field index
Property value	StdFont	font attributes of the text

Example Code

```
Dim borderArea As VcBorderArea
Dim bBoxTL As VcBorderBox

Set borderArea = VcGantt1.BorderArea
Set bBoxBBL = borderArea.BorderBox(vcBBXPBottomBottomLeft)

bBoxTL.TextFont(i).Bold = False
bBoxTL.TextFont(i).Italic = False
bBoxTL.TextFont(i).Name = "Symbol"
```

Code Sample in C#

```
/ Text for Title
VcBorderBox borderBox =
VcGantt1.BorderArea.BorderBox(VcBorderBoxPosition.vcBBXPTopCentered);
borderBox.Type = VcBorderBoxType.vcBBXTText;

Font titleFont1 = new Font("Arial", 20, FontStyle.Bold);

borderBox.set_Text(1, "Time Scheduler");
borderBox.set_TextFont(1, titleFont1);
```

Type

Property of VcBorderBox

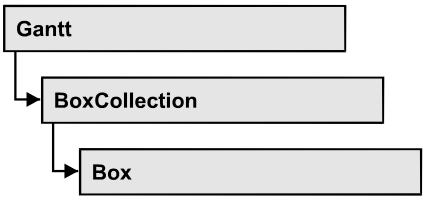
This property lets you set or retrieve the type of the BorderBox object.

	Data Type	Explanation
Property value	BorderBoxTypeEnum	box type
	Possible Values:	
	vcBBXTGraphics 3	graphics
	vcBBXTLegend 4	legend
	vcBBXTNothing 0	nothing
	vcBBXTText 1	text
	vcBBXTTextWithGraphics 2	text and graphics

Example Code

```
Dim borderArea As VcBorderArea
Dim bBoxBBL As VcBorderBox

Set borderArea = VcGantt1.BorderArea
Set bBoxBBL = borderArea.BorderBox(vcBBXPBottomBottomLeft)
bBoxBBL.Type = vcBBXTGraphics
```

7.6 VcBox



An object of the type **VcBox** designates a box to display texts or graphics.

Properties

- AnchoringInteractionsAllowed
- AnchoringLineVisible
- FieldText
- FormatName
- LineColor
- LineThickness
- LineType
- MarkBox
- Moveable
- Name
- NodeID
- Origin
- Priority
- ReferencePoint
- Resizing
- Specification
- UpdateBehaviorName
- Visible

Methods

- AnchorToNode
- GetActualExtent
- GetTopLeftPixel
- GetXYOffset
- GetXYOffsetAsVariant
- IdentifyFormatField
- SetXYOffset
- SetXYOffsetByTopLeftPixel

Properties

AnchoringInteractionsAllowed

Property of VcBox

This property lets you set or retrieve whether a box can be tied to a node interactively .

	Data Type	Explanation
Property value	Boolean	Box can/cannot be tied to a node interactively **Default value:** False
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

Set boxCltn = VcGantt1.BoxCollection
Set box = boxCltn.BoxByIndex(0)
box.AnchoringInteractionsAllowed = False
```

AnchoringLineVisible

Property of VcBox

This property lets you set or retrieve whether the specified reference points shall be linked by a line if the box is tied to a node.

	Data Type	Explanation
Property value	Boolean	Anchoring line between node and box is/is not shown **Default value:** False
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

Set boxCltn = VcGantt1.BoxCollection
Set box = boxCltn.BoxByIndex(0)
box.AnchoringLineVisible = False
```

FieldText

Property of VcBox

This property lets you set or retrieve the contents of a box field. You also can specify the offset in the **Edit Box** dialog box.

If a text field contains more than one line, you can use "\n" in the text string to separate two lines of the text field (Example: "Line1\nLine2"). Otherwise the lines will be separated at blanks.

	Data Type	Explanation
Parameter:		
⇨ fieldIndex	Integer	Field index
	Possible Values:	Data field index
Property value	String	Field content
	Possible Values:	Name of the color map

Example Code

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

Set boxCltn = VcGantt1.boxCollection
Set box = boxCltn.FirstBox
box.FieldText(0) = "User: "
```

FormatName

Property of VcBox

This property lets you set or retrieve the name of the box format.

	Data Type	Explanation
Property value	VcBoxFormat	BoxFormat object or **Nothing**

Example Code

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcGantt1.BoxCollection
box = boxCltn.FirstBox
box.FormatName = "Standard"
```

LineColor

Property of VcBox

This property lets you set or retrieve the color of the border line of the box.

	Data Type	Explanation
Property value	Color	RGB color values ({0...255},{0...255},{0...255})

Example Code

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

Set boxCltn = VcGantt1.BoxCollection
Set box = boxCltn.BoxByIndex(0)
box.LineColor = RGB(255, 0, 0)
```

LineThickness

Property of VcBox

This property lets you set or retrieve the line thickness of the border line of the box.

If you set this property to values between 1 and 4, an absolute line thickness is defined in pixels. Irrespective of the zoom factor a line will always show the same line thickness in pixels. When printing though, the line thickness is adapted for the sake of legibility and becomes dependent of the zoom factor:

Value	Points	mm
1	1/2 point	0.09 mm
2	1 point	0.18 mm
3	3/2 points	0.26 mm
4	2 points	0.35 mm

A point equals 1/72 inch and represents the unit of the font size.

If you set this property to values between 5 and 1,000, the line thickness is defined in 1/100 mm, so the lines will be displayed in a true thickness in pixels that depends on the zoom factor.

	Data Type	Explanation
Property value	Integer	Line thickness LineType {1...4}: line thickness in pixels LineType {5...1000}: line thickness in 1/100 mm **Default value:** As defined in the dialog
	Possible Values:	Data field index

Example Code

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

Set boxCltn = VcGantt1.BoxCollection
Set box = boxCltn.BoxByIndex(0)
box.LineThickness = 2
```

LineType

Property of VcBox

This property lets you set or retrieve the type of the border line of the box.

	Data Type	Explanation
Property value	LineTypeEnum	Line type **Default value:** vcSolid
	Possible Values:	
	vcDashed 4	Line dashed
	vcDashedDotted 5	Line dashed-dotted
	vcDotted 3	Line dotted
	vcLineType0 100	Line Type 0
	vcLineType1 101	Line Type 1
	vcLineType10 110	Line Type 10
	vcLineType11 111	Line Type 11
	vcLineType12 112	Line Type 12
	vcLineType13 113	Line Type 13
	vcLineType14 114	Line Type 14
	vcLineType15 115	Line Type 15
	vcLineType16 116	Line Type 16
	vcLineType17 117	Line Type 17
	vcLineType18 118	Line Type 18
	vcLineType2 102	Line Type 2

vcLineType3 103	Line Type 3
vcLineType4 104	Line Type 4
vcLineType5 105	Line Type 5
vcLineType6 106	Line Type 6
vcLineType7 107	Line Type 7
vcLineType8 108	Line Type 8
vcLineType9 109	Line Type 9
vcNone 1	No line type
vcNotSet -1	No line type assigned
vcSolid 2	Line solid

Example Code

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

Set boxCltn = VcGantt1.BoxCollection
Set box = boxCltn.BoxByIndex(0)
box.LineType = vcDotted
```

MarkBox

Property of VcBox

By this property you can set or retrieve whether a box is marked.

	Data Type	Explanation
Property value	Boolean	**True**: box marked; **false**: box unmarked
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

Set boxCltn = VcGantt1.BoxCollection
Set box = boxCltn.BoxByIndex(0)
box.MarkBox = True
```

Moveable

Property of VcBox

This property lets you set or retrieve whether the box can be moved interactively.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	Moveable (True)/ not moveable (False) **Default value:** True Group invisible/visible group nodes are/are not visible

Example Code

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

Set boxCltn = VcGantt1.BoxCollection
Set box = boxCltn.BoxByIndex(0)
box.Moveable = False
```

Name

Property of VcBox

This property lets you retrieve/set the name of a box. You can specify the name in the **Administrate Boxes** dialog box.

	Data Type	Explanation
Property value	String **Possible Values:**	Box name Name of the color map

Example Code

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox
Dim boxName As String

Set boxCltn = VcGantt1.boxCollection
Set box = boxCltn.FirstBox
boxName = box.Name
MsgBox boxName
```

NodeID

Property of VcBox

This property lets you set or retrieve the node ID of the node which the box is tied to. You can also specify the Node-ID in the **Administrate Boxes** dialog box.

	Data Type	Explanation
Property value	String **Possible Values:**	ID of the node the box is tied to Name of the color map

Example Code

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox
Dim box.NodeID As String

Set boxCltn = VcGantt1.boxCollection
Set box = boxCltn.FirstBox
box.NodeID = 3
```

Origin

Property of VcBox

This property lets you set or retrieve the point of origin of the box, i. e. the point of the diagram from which the offset to the reference point of the box will be measured.

By using the properties **Origin**, **ReferencePoint** and the method **GetXYOffset** you can position boxes individually in the diagram area. The relative position of a box does not depend on the diagram size.

	Data Type	Explanation
Property value	BoxOriginEnum **Possible Values:** vcBOBottomCenter 28 vcBOBottomLeft 27 vcBOBottomRight 29 vcBOCenterCenter 25 vcBOCenterLeft 24 vcBOCenterRight 26 vcBOTopCenter 22 vcBOTopLeft 21 vcBOTopRight 23	origin of the box bottom center bottom left bottom right center center center left center right top center top left top right

Example Code

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

Set boxCltn = VcGantt1.BoxCollection
Set box = boxCltn.BoxByIndex(0)
box.Origin = vcBOTopCenter
```

Priority

Property of VcBox

This property lets you specify or enquire the priority of the box.

	Data Type	Explanation
Property value	Integer **Possible Values:**	Priority value Data field index

Example Code

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

Set boxCltn = VcGantt1.BoxCollection
Set box = boxCltn.BoxByIndex(0)
box.Priority = 3
```

ReferencePoint

Property of VcBox

This property lets you set or retrieve the reference point of the box, i. e. the point of the box from which the offset to the origin will be measured.

	Data Type	Explanation
Property value	BoxReferencePointEnum **Possible Values:** vcBRPBottomCenter 28 vcBRPBottomLeft 27 vcBRPBottomRight 29 vcBRPCenterCenter 25 vcBRPCenterLeft 24 vcBRPCenterRight 26 vcBRPTopCenter 22 vcBRPTopLeft 21 vcBRPTopRight 23	reference point of the box bottom center bottom left bottom right center center center left center right top center top left top right

Example Code

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

Set boxCltn = VcGantt1.BoxCollection
Set box = boxCltn.BoxByIndex(0)
box.ReferencePoint = vcBRPCenterRight
```

Resizing

Property of VcBox

This property lets you set or retrieve whether and how the size of a box can be modified.

	Data Type	Explanation
Property value	BoxResizingEnum	Interactive modification of the size of the box
	Possible Values:	
	vcBRHeight 23	The height of the box can be modified interactively.
	vcBRNo 0	The size of the box cannot be modified interactively.
	vcBRWidth 24	The width of the box can be modified interactively.
	vcBRWidth/Height 1050	Width and height of the box can be modified interactively.

Example Code

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

Set boxCltn = VcGantt1.BoxCollection
Set box = boxCltn.BoxByIndex(0)
box.Resizing = vcBRWidth
```

Specification

Read Only Property of VcBox

This property lets you retrieve the specification of a box. A specification is a string that contains legible ASCII characters from 32 to 127 only, so it can be stored without problems to text files or data bases. This allows for persistency. A specification can be used to create a box by the method **VcBoxCollection.AddBySpecification**.

	Data Type	Explanation
Property value	String	Specification of the box
	Possible Values:	
		Name of the color map

Example Code

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

Set boxCltn = VcGantt1.BoxCollection
Set box = boxCltn.BoxByIndex(0)
MsgBox box.Specification
```

UpdateBehaviorName

Property of VcBox

This property lets you set or retrieve the name of the UpdateBehavior.

	Data Type	Explanation
Property value	String	Name of the UpdateBehavior
	Possible Values:	Name of the color map

Visible

Property of VcBox

This property lets you set or retrieve whether a box is visible. You also can specify this property in the **Administrate Boxes** dialog box.

	Data Type	Explanation
Property value	Boolean	box visible/invisible **Default value:** True
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

Set boxCltn = VcGantt1.BoxCollection
Set box = boxCltn.FirstBox
box.Visible = False
```

Methods

AnchorToNode

Method of VcBox

This method lets you tie boxes to nodes or untie them again. An anchored box can be still moved (provided that you have set the property **Moveable**). To untie a box from the node, you have to pass "NULL" as parameter.

If you move a node which is anchored to a box, the box is moved as well. If the node is collapsed, the box is collapsed as well, thus becoming invisible. When the node is expanded the box is visible again.

If a box is tied to a node, its position on the screen will be maintained. The offset values which are used as basis are converted according to the reference points (Origin, ReferencePoint). If, for example, a box with a certain offset refers to a chart at the top left (origin) and then is anchored to a node, an offset to the the top left node is calculated automatically. This makes sure that the position on the screen will not be altered. If the box is untied from the node the calculation is carried out backwards.

	Data Type	Explanation
Parameter:		
⇨ node	VcNode	Node object to which the box is tied
Return value	Boolean	Box is anchored to node/untied from node

GetActualExtent

Method of VcBox

This method lets you retrieve the extent of the box (unit: 1/100 mm).

By regarding these values when setting the XY offset, you can modify the reference point of the anchoring line without changing the position of the box.

	Data Type	Explanation
Parameter:		
⇦ width	Integer **Possible Values:**	width of the box Data field index
⇦ height	Integer **Possible Values:**	height of the box Data field index
Return value	Boolean	Extent of the box is returned/not returned

GetTopLeftPixel

Method of VcBox

This method lets you convert to pixel and display the saved XY offset for the top left corner.

The x value can be further used with the method **VcGantt.GetDate** for instance to get a date.

	Data Type	Explanation
Parameter:		
⇦ x	Integer **Possible Values:**	X value of the offset Data field index
⇦ y	Integer **Possible Values:**	Y value of the offset Data field index
Return value	Boolean	Offset is returned/not returned

GetXYOffset

Method of VcBox

This method lets you enquire the distance between origin and reference point in x and y direction (unit: 1/100 mm).

Note: If you use VBScript, you can only use the analogous method **GetXYOffsetAsVariant** because of the parameters by Reference.

	Data Type	Explanation
Parameter:		
⇦ xOffset	Integer **Possible Values:**	X value of the offset Data field index
⇦ yOffset	Integer **Possible Values:**	Y value of the offset Data field index
Return value	Boolean	Offset is returned/not returned

GetXYOffsetAsVariant

Method of VcBox

This method is identical with the method **GetXYOffset** except for the parameters. It was necessary to implement this event because some languages (e.g. VBScript) can use parameters by Reference (indicated by ⇦) only if the type of these parameters is VARIANT.

IdentifyFormatField

Method of VcBox

This method lets you retrieve the index of the format field at the specified position. If there is a field at the position specified, **True** will be returned, if there isn't, the method will deliver **False**.

	Data Type	Explanation
Parameter:		
⇨ x	Long	X coordinate of the position
⇨ y	Long	Y coordinate of the position
⇦ format	VcBoxFormat	Identified format
⇦ formatFieldIndex	Integer	Index of the format field
	Possible Values:	Data field index
Return value	Boolean	A format field exists/does not exist at the position specified

SetXYOffset

Method of VcBox

This method lets you specify the distance between origin and reference point in x and y direction (unit: 1/100 mm).

You also can specify the offset in the **Administrate Boxes** dialog box.

Note: If you use VBScript, you can only use the analogous method **GetXYOffsetAsVariant** because of the parameters by Reference.

	Data Type	Explanation
Parameter:		
⇨ xOffset	Integer **Possible Values:**	X value of the offset Data field index
⇨ yOffset	Integer **Possible Values:**	Y value of the offset Data field index
Return value	Boolean	Offset is set (True) / not set (False)

Example Code

```
Dim OffsetSet As Boolean
OffsetSet = VcGantt1.boxCollection.FirstBox.SetXYOffset(100, 100)
```

SetXYOffsetByTopLeftPixel

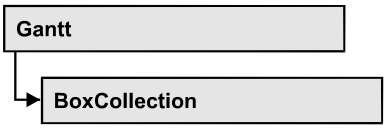
Method of VcBox

This method lets you internally convert the specified pixel value of the top left corner to an XY offset and then save the offset.

This enables you for instance to place a box at an XY coordinate from an event.

	Data Type	Explanation
Parameter:		
⇨ x	Integer **Possible Values:**	X value of the offset Data field index
⇨ y	Integer **Possible Values:**	Y value of the offset Data field index
Return value	Boolean	Offset is set (True) / not set (False)

7.7 VcBoxCollection



The VcBoxCollection object contains all boxes available. You can access all objects in an iterative loop by **For Each box In BoxCollection** or by the methods **First...** and **Next...**. You can access a single box by the method **BoxByName** and **BoxByIndex**. The number of boxes in the collection object can be retrieved by the property **Count**. The methods **Add**, **Copy** and **Remove** allow to handle the boxes in the corresponding way.

Properties

- _NewEnum
- Count

Methods

- Add
- AddBySpecification
- BoxByIndex
- BoxByName
- Copy
- FirstBox
- NextBox
- Remove
- Update

Properties

_NewEnum

Read Only Property of VcBoxCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all box objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method **GetEnumerator** is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim box As VcBox

For Each box In VcGantt1.BoxCollection
   Debug.Print box.Name
Next
```

Count

Read Only Property of VcBoxCollection

This property lets you retrieve the number of boxes in the box collection.

	Data Type	Explanation
Property value	Long	Number of boxes

Example Code

```
Dim boxCltn As VcBoxCollection
Dim numberOfBoxes As Long

Set boxCltnn = VcGantt1.BoxCollection
Dim numberOfBoxes = boxCltn.Count
```

Methods

Add

Method of VcBoxCollection

By this method you can create a box as a member of the BoxCollection. If the name was not used before, the new box object will be returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned. To make the new box visible in the diagram, the box collection needs to be updated by the **Update** call.

	Data Type	Explanation
Parameter:		
⇨ boxName	String **Possible Values:**	Box name Name of the color map
Return value	VcBox	New box object

Example Code

```
Set newBox = VcGantt1.BoxCollection.Add("box1")
```

AddBySpecification

Method of VcBoxCollection

This method lets you create a box by using by a box specification. This way you can keep a box persistent. This way of creating allows box objects to become persistent. The specification of a box can be saved and re-loaded (see VcBox property **Specification**). In a subsequent the box can be created can be created again from the specification and is identified by its name. To make the new box visible in the diagram, the box collection needs to be updated by the **Update** call.

	Data Type	Explanation
Parameter:		
⇨ Specification	String	Box specification
	Possible Values:	Name of the color map
Return value	VcBox	New box object

BoxByIndex

Method of VcBoxCollection

This method lets you access a box by its index. If a box of the specified index does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter:		
⇨ index	Integer	Index of the box
	Possible Values:	Data field index
Return value	VcBox	Box object returned

Example Code

```
Dim boxCltn As VcBoxCollection

Set boxCltn = VcGantt1.BoxCollection
Set box = boxCltn.BoxByIndex(2)
box.LineThickness = 2
```

BoxByName

Method of VcBoxCollection

By this method you can retrieve a box by its name. If a box of the specified name does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter:		
⇨ boxName	String **Possible Values:**	Box name Name of the color map
Return value	VcBox	Box

Example Code

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

Set boxCltn = VcGantt1.BoxCollection
Set box = boxCltn.BoxByName("Box 1")
```

Copy

Method of VcBoxCollection

By this method you can copy a box. If the box that is to be copied exists, and if the name for the new box does not yet exist, the new box object is returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned. To make the copied box visible in the diagram, the box collection needs to be updated by the **Update** call.

	Data Type	Explanation
Parameter:		
⇨ boxName	String **Possible Values:**	Name of the box to be copied Name of the color map
⇨ newBoxName	String **Possible Values:**	Name of the new box Name of the color map
Return value	VcBox	Box object

Example Code

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox
```

```
Set boxCltn = VcGantt1.BoxCollection
Set box = boxCltn.Copy("BoxOne", "NewBox")
boxCltn.Update
```

FirstBox

Method of VcBoxCollection

This method can be used to access the initial value, i.e. the first box of a box collection, and then to continue in a forward iteration loop by the method **NextBox** for the boxes following. If there is no box in the BoxCollection object, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcBox	First box

Example Code

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

Set boxCltn = VcGantt1.BoxCollection
Set box = boxCltn.FirstBox
```

NextBox

Method of VcBoxCollection

This method can be used in a forward iteration loop to retrieve subsequent boxes from a box collection after initializing the loop by the method **FirstBox**. If there is no box left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcBox	Subsequent box

Example Code

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

Set boxCltn = VcGantt1.BoxCollection
Set box = boxCltn.FirstBox

While Not box Is Nothing
    Listbox.AddItem box.Name
    Set box = boxCltn.NextBox
Wend
```

Remove

Method of VcBoxCollection

This method lets you delete a box. To make the deletion visible in the diagram, the box collection needs to be updated by the **Update** call.

	Data Type	Explanation
Parameter:		
⇨ boxName	String	Box name
	Possible Values:	Name of the color map
Return value	Boolean	Box deleted (True)/not deleted (False)

Example Code

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

Set boxCltn = VcGantt1.BoxCollection
Set box = boxCltn.BoxByIndex(2)
boxCltn.Remove (box.Name)
boxCltn.Update
```

Update

Method of VcBoxCollection

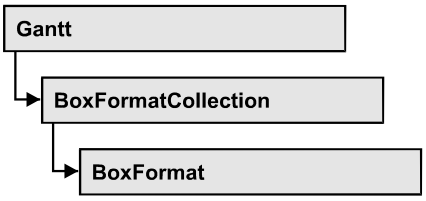
This method lets you update a box collection after having modified it.

	Data Type	Explanation
Return value	Boolean	update successful (True)/ not successful (False)

Example Code

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

Set boxCltn = VcGantt1.BoxCollection
Set box = boxCltn.BoxByIndex(2)
boxCltn.Remove (box.Name)
boxCltn.Update
```

7.8 VcBoxFormat



An object of the type **VcBoxFormat** defines the formats of boxes.

Properties

- _NewEnum
- FieldsSeparatedByLines
- FormatField
- FormatFieldCount
- Name
- Specification

Methods

- CopyFormatField
- RemoveFormatField

Properties

_NewEnum

Read Only Property of VcBoxFormat

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all box format field objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method **GetEnumerator** is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim formatField As VcBoxFormatField
```

```
For Each formatField In format
    Debug.Print formatField.Index
Next
```

FieldsSeparatedByLines

Property of VcBoxFormat

This property lets you set or retrieve whether fields are to be separated by lines.

	Data Type	Explanation
Property value	Boolean	Box fields separated by lines (True)/ not separated by lines (False).
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
Dim boxFormat As VcBoxFormat

Set boxFormat = VcGantt1.BoxFormatCollection.FormatByIndex(2)
boxFormat.FieldsSeparatedByLines = True
```

FormatField

Read Only Property of VcBoxFormat

This property lets you access a VcBoxFormatField object by its index. The index has to be in the range 0 to .FormatFieldCount-1.

Note for users of a version earlier than 3.0: The index does **not** count from 1 to .FormatFieldCount as (as did the field properties up to 3.0).

	Data Type	Explanation
Parameter:		
index	Integer	Index of the box format field 0 ... .FormatFieldCount-1
	Possible Values:	Data field index
Property value	VcBoxFormatField	Box format field

Example Code

```
Dim boxFormat As VcBoxFormat
Dim formatField As VcBoxFormatField
```

```
Set boxFormat = VcGantt1.BoxFormatCollection.FirstFormat
Set formatField = boxFormat.formatField(0)
MsgBox formatField.FormatName
```

FormatFieldCount

Read Only Property of VcBoxFormat

This property allows to determine the number of fields in a box format.

	Data Type	Explanation
Property value	Integer	Number of fields of the box format
	Possible Values:	Data field index

Example Code

```
Dim boxFormat As VcBoxFormat
Dim formatField As VcBoxFormatField

Set boxFormat = VcGantt1.BoxFormatCollection.FirstFormat
MsgBox boxFormat.FormatFieldCount
```

Name

Property of VcBoxFormat

This property lets you retrieve/set the name of a box format. You can also specify the name in the **Administrate Box Formats** dialog box.

	Data Type	Explanation
Property value	String	Box format name
	Possible Values:	Name of the color map

Example Code

```
Dim boxFormat As VcBoxFormat

For Each boxFormat In VcGantt1.BoxFormatCollection
   List1.AddItem (boxFormat.Name)
Next
```

Specification

Read Only Property of VcBoxFormat

This property lets you retrieve the specification of a box Format. A specification is a string that contains legible ASCII characters from 32 to 127

only, so it can be stored without problems to text files or data bases. This allows for persistency. A specification can be used to create a box format by the method **VcBoxFormatCollection.AddBySpecification**.

	Data Type	Explanation
Property value	String **Possible Values:**	Specification of the box format Name of the color map

Methods

CopyFormatField

Method of VcBoxFormat

This method allows to copy a box format field. The new VcBoxFormatField object is returned. It is given automatically the next index not used before.

	Data Type	Explanation
Parameter:		
⇨ position	FormatFieldInnerPositionEnum **Possible Values:** vcInnerAbove 1 vcInnerBelow 3 vcInnerLeftOf 0 vcInnerRightOf 4	Position of the new box format field above below left of right of
⇨ refIndex	Integer **Possible Values:**	Index of the reference box format field Data field index
Return value	VcBoxFormatField	Box format field object

Example Code

```
Dim boxFormat As VcBoxFormat
Dim formatField As VcBoxFormatField

Set boxFormat = VcGantt1.BoxFormatCollection.FormatByIndex(2)
Set formatField = boxFormat.CopyFormatField(vcInnerRightOf, 0)
```

RemoveFormatField

Method of VcBoxFormat

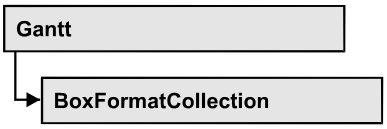
This method lets you remove a layer format field by its index. After that, the program will update all layer format field indexes so that they are consecutively numbered again.

	Data Type	Explanation
Parameter:		
⇨ index	Integer	index of the box format field to be deleted
	Possible Values:	Data field index

Example Code

```
Dim boxFormat As VcBoxFormat
Dim i As Integer

boxFormat = VcGantt1.BoxFormatCollection.FirstFormat

For i = 0 To boxFormat.FormatFieldCount - 1
   boxFormat.RemoveFormatField (i)
Next
```

7.9 VcBoxFormatCollection



The VcBoxFormatCollection object contains all box formats available. You can access all objects in an iterative loop by **For Each boxFormat In BoxFormatCollection** or by the methods **First...** and **Next...**. You can access a single box format by the methods **BoxFormatByName** and **BoxFormat-ByIndex**. The number of box formats in the collection object can be retrieved by the property **Count**. The methods **Add**, **Copy** and **Remove** allow to handle the box formats in the corresponding way.

Properties

- _NewEnum
- Count

Methods

- Add
- AddBySpecification
- Copy
- FirstFormat
- FormatByIndex
- FormatByName
- NextFormat
- Remove

Properties

_NewEnum

Read Only Property of VcBoxFormatCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all box format objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method **GetEnumerator** is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim format As VcBoxFormat

For Each format In VcGantt1.BoxCollection
   Debug.Print format.Name
Next
```

Count

Read Only Property of VcBoxFormatCollection

This property lets you retrieve the number of box formats in the box format collection.

	Data Type	Explanation
Property value	Long	Number of box formats

Example Code

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim numberOfBoxformats As Long

Set boxFormatCltn = VcGantt1.BoxFormatCollection
Dim numberOfBoxformats = boxFormatCltn.Count
```

Methods

Add

Method of VcBoxFormatCollection

By this method you can create a box format as a member of the BoxFormatCollection. If the name was not used before, the new box object will be returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
Parameter:		
⇨ FormatName	String	Name of the box format
	Possible Values:	Name of the color map
Return value	VcBoxFormat	New box format object

Example Code

```
Set newBoxFormat = VcGantt1.BoxFormatCollection.Add("boxFormat1")
```

AddBySpecification

Method of VcBoxFormatCollection

This method lets you create a box format by using a box format specification. This way of creating allows box format objects to become persistent. The specification of a box format can be saved and re-loaded (see VcBoxFormat property **Specification**). In a subsequent session the box format can be created again from the specification and is identified by its name.

	Data Type	Explanation
Parameter:		
⇨ formatSpecification	String **Possible Values:**	Box format specification Name of the color map
Return value	VcBoxFormat	New box format object

Copy

Method of VcBoxFormatCollection

By this method you can copy a box format. If the box format that is to be copied exists, and if the name for the new box format does not yet exist, the new box format object is returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
Parameter:		
⇨ FormatName	String **Possible Values:**	Name of the box format to be copied Name of the color map
⇨ newFormatName	String **Possible Values:**	Name of the new box format Name of the color map
Return value	VcBoxFormat	Box format object

Example Code

```
Dim boxFormatCltn As VcBoxFormatCollection
```

```
Dim boxFormat As VcBoxFormat

Set boxFormatCltn = VcGantt1.BoxFormatCollection
Set boxFormat = boxFormatCltn.Copy("CurrentBoxFormat", "NewBoxFormat")
```

FirstFormat

Method of VcBoxFormatCollection

This method can be used to access the initial value, i.e. the first box format of a box format collection and then to continue in a forward iteration loop by the method **NextFormat** for the box formats following. If there is no box format in the box format collection, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcBoxFormat	First box format

Example Code

```
Dim format As VcBoxFormat

Set format = VcGantt1.BoxFormatCollection.FirstFormat
```

FormatByIndex

Method of VcBoxFormatCollection

This method lets you access a box format by its index. If a box format of the specified index does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter:		
⇨ index	Integer **Possible Values:**	Index of the box format Data field index
Return value	VcBoxFormat	Box format object returned

Example Code

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim format As VcBoxFormat

Set boxFormatCltn = VcGantt1.BoxFormatCollection
Set format = boxFormatCltn.FormatByIndex(2)
```

FormatByName

Method of VcBoxFormatCollection

By this method you can retrieve a box format by its name. If a box format of the specified name does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter:		
⇨ formatName	String	Name of the box format
	Possible Values:	Name of the color map
Return value	VcBoxFormat	Box format

Example Code

```
Dim formatCltn As VcBoxFormatCollection
Dim format As VcBoxFormat

Set formatCltn = VcGantt1.BoxFormatCollection
Set format = formatCltn.FormatByName("Standard")
```

NextFormat

Method of VcBoxFormatCollection

This method can be used in a forward iteration loop to retrieve subsequent box formats from a box format collection after initializing the loop by the method **FirstFormat**. If there is no format left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcBoxFormat	Subsequent box format

Example Code

```
Dim formatCltn As VcBoxFormatCollection
Dim format As VcBoxFormat

Set formatCltn = VcGantt1.BoxFormatCollection
Set format = formatCltn.FirstFormat

While Not format Is Nothing
     List1.AddItem format.Name
     Set format = formatCltn.NextFormat
Wend
```

Remove

Method of VcBoxFormatCollection

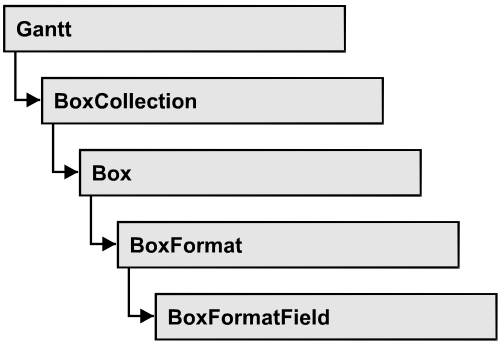
This method lets you delete a box format. If the box format is used in another object, it cannot be deleted. Then False will be returned, otherwise True.

	Data Type	Explanation
Parameter:		
⇨ FormatName	String	Box format name
	Possible Values:	Name of the color map
Return value	Boolean	Box format deleted (True)/not deleted (False)

Example Code

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormat As VcBoxFormat

Set boxFormatCltn = VcGantt1.BoxFormatCollection
Set boxFormat = boxFormatCltn.FormatByIndex(1)
boxFormatCltn.Remove (boxFormat.Name)
```

7.10 VcBoxFormatField



An object of the type **VcBoxFormat** represents a field of a VcBoxFormat object. A box format field does not have a name as many other objects, but it has an index that defines its position in the box format.

Properties

- Alignment
- FormatName
- GraphicsHeight
- Index
- MaximumTextLineCount
- MinimumTextLineCount
- MinimumWidth
- PatternBackgroundColorAsARGB
- PatternColorAsARGB
- PatternEx
- TextFont
- TextFontColor
- Type

Properties

Alignment

Property of VcBoxFormatField

This property lets you set or retrieve the alignment of the content of the box format field.

	Data Type	Explanation
Property value	FormatFieldAlignmentEnum **Possible Values:** vcFFABottom 28 vcFFABottomLeft 27 vcFFABottomRight 29 vcFFACenter 25 vcFFALeft 24 vcFFARight 26 vcFFATop 22 vcFFATopLeft 21 vcFFATopRight 23	Alignment of the field content bottom bottom left bottom right center left right top top left top right

Example Code

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

Set boxFormatCltn = VcGantt1.BoxFormatCollection
Set boxFormatField = boxFormatCltn.FirstFormat.formatField(0)
boxFormatField.Alignment = vcFFACenter
```

FormatName

Read Only Property of VcBoxFormatField

This property lets you retrieve the name of the box format to which this box format field belongs.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the box format Name of the color map

Example Code

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

Set boxFormatCltn = VcGantt1.BoxFormatCollection
Set boxFormatField = boxFormatCltn.FirstFormat.formatField(0)
MsgBox boxFormatField.FormatName
```

GraphicsHeight

Property of VcBoxFormatField

This property lets you set or retrieve for the type **vcFFTGraphics** the height of the graphics in the box format field.

	Data Type	Explanation
Property value	Integer	Height of the graphics in mm
	Possible Values:	0 ... 99 Data field index

Example Code

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

Set boxFormatCltn = VcGantt1.BoxFormatCollection
Set boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.Type = vcFFTGraphics
boxFormatField.GraphicsHeight = 150
```

Index

Read Only Property of VcBoxFormatField

This property lets you enquire the index of the box format field in the corresponding box format.

	Data Type	Explanation

Example Code

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

Set boxFormatCltn = VcGantt1.BoxFormatCollection
Set boxFormatField = boxFormatCltn.FirstFormat.formatField(0)
MsgBox boxFormatField.Index
```

MaximumTextLineCount

Property of VcBoxFormatField

This property lets you set or retrieve the maximum number of lines in the box format field, if the box format field is of the type **vcFFTText**. Also see the property **MinimumTextLineCount**.

	Data Type	Explanation
Property value	Integer	Maximum number of lines
	Possible Values:	0 ... 9 Data field index

Example Code

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

Set boxFormatCltn = VcGantt1.BoxFormatCollection
Set boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.Type = vcFFTText
boxFormatField.MaximumTextLineCount = 5
```

MinimumTextLineCount

Property of VcBoxFormatField

This property lets you set or retrieve the minimum number of lines in the box format field, if it is of the type **vcFFTText**. If there is more text than can be taken by the lines, the format field will be enlarged dynamically up to the maximum number of lines. When assigning a value by this property, please also remember to set the **MaximumTextLineCount** value anew, since otherwise the minimum value might overwrite the maximum value.

	Data Type	Explanation
Property value	Integer	Minimum number of lines
	Possible Values:	0 ... 9 Data field index

Example Code

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

Set boxFormatCltn = VcGantt1.BoxFormatCollection
Set boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.Type = vcFFTText
boxFormatField.MinimumTextLineCount = 3
```

MinimumWidth

Property of VcBoxFormatField

This property lets you set or retrieve the minimum width of the box field in mm. The field width may be enlarged, if above or below the field fields exist that have greater minimum widths.

	Data Type	Explanation
Property value	Integer **Possible Values:**	Minimum width of the box format field 0 ... 9 Data field index

Example Code

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

Set boxFormatCltn = VcGantt1.BoxFormatCollection
Set boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.MinimumWidth = 100
```

PatternBackgroundColorAsARGB

Property of VcBoxFormatField

This property lets you set or retrieve the background color of the box format field. Color values have a transparency or alpha value, followed by a value for a red, a blue and a green partition (ARGB). The values range between 0..255. An alpha value of 0 equals complete transparency, whereas 255 represents a completely solid color. When casting an RGB value on an ARGB value, an alpha value of 255 has to be added.

If the box format field shall have the background color of the box format, select the value **-1**.

	Data Type	Explanation
Property value	Long	Background color of the box format **Default value:** -1

Example Code

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

Set boxFormatCltn = VcGantt1.BoxFormatCollection
Set boxFormatField = boxFormatCltn.FirstFormat.formatField(0)
boxFormatField.BackColor = RGB(0, 255, 0)
```

PatternColorAsARGB

Property of VcBoxFormatField

This property lets you set or retrieve the pattern color of the box format field. Color values have a transparency or alpha value, followed by a value for a

red, a blue and a green partition (ARGB). The values range between 0..255. An alpha value of 0 equals complete transparency, whereas 255 represents a completely solid color. When casting an RGB value on an ARGB value, an alpha value of 255 has to be added.

If the box format field shall have the background color of the box format, select the value **-1**.

	Data Type	Explanation
Property value	Long	Pattern color of the box format field

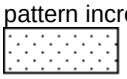
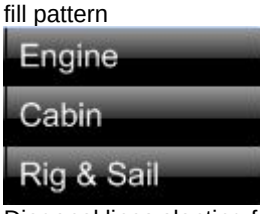
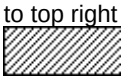
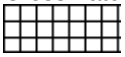
Example Code

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

Set boxFormatCltn = VcGantt1.BoxFormatCollection
Set boxFormatField = boxFormatCltn.FirstFormat.formatField(0)
boxFormatField.PatternColor = RGB(0, 255, 0)
```

PatternEx

Property of VcBoxFormatField

This property lets you set or retrieve the pattern of the field background of the box format field.

	Data Type	Explanation
Property value	FillPatternEnum	Pattern type **Default value:** As defined in the dialog
	Possible Values: vc05PercentPattern... vc90PercentPattern 01 - 11	Dots in foreground color on background color, the density of the foreground pattern increasing with the percentage 
	vcAeroGlassPattern 40	Vertical color gradient in the color of the fill pattern 
	vcBDiagonalPattern 5	Diagonal lines slanting from bottom left to top right 
	vcCrossPattern 6	Cross-hatch pattern 

vcDarkDownwardDiagonalPattern 2014	Diagonal lines slanting from top left to bottom right; spaced 50% closer than vcFDiagonalPattern and of twice the line width
vcDarkHorizontalPattern 2023	Horizontal lines spaced 50% closer than vcHorizontalPattern and of twice the line width
vcDarkUpwardDiagonalPattern 2015	Diagonal lines slanting from bottom left to top right, spaced 50% closer than vcBDiagonalPattern and of twice the line width
vcDarkVerticalPattern 2022	Vertical lines spaced 50% closer than vcVerticalPattern and of of twice the line width
vcDashedDownwardDiagonalPattern 2024	Dashed diagonal lines from top left to bottom right
vcDashedHorizontalPattern 2026	Dashed horizontal lines
vcDashedUpwardDiagonalPattern 2025	Dashed diagonal lines from bottom left to top right
vcDashedVerticalPattern 2027	Dashed vertical lines
vcDiagCrossPattern 7	Diagonal cross-hatch pattern, small
vcDiagonalBrickPattern 2032	Diagonal brick pattern
vcDivotPattern 2036	Divot pattern
vcDottedDiamondPattern 2038	Diagonal cross-hatch pattern of dotted lines
vcDottedGridPattern 2037	Cross-hatch pattern of dotted lines
vcFDiagonalPattern 4	Diagonal lines slanting from top left to bottom right
vcHorizontalBrickPattern 2033	Horizontal brick pattern
vcHorizontalGradientPattern 52	Horizontal color gradient
vcHorizontalPattern 3	Horizontal lines

vcLargeCheckerboardPattern 2044	Checkerboard pattern showing squares of twice the size of vcSmallChecker-BoardPattern
vcLargeConfettiPattern 2029	Confetti pattern, large
vcLightDownwardDiagonalPattern 2012	Diagonal lines slanting to from top left to bottom right; spaced 50% closer than vcBDiagonalPattern
vcLightHorizontalPattern 2019	Horizontal lines spaced 50% closer than vcHorizontalPattern
vcLightUpwardDiagonalPattern 2013	Diagonal lines slanting from bottom left to top right, spaced 50% closer than vcBDiagonalPattern
vcLightVerticalPattern 2018	Vertical lines spaced 50% closer than vcVerticalPattern
vcNarrowHorizontalPattern 2021	Horizontal lines spaced 75 % closer than vcHorizontalPattern
vcNarrowVerticalPattern 2020	Vertical lines spaced 75% closer than vcVerticalPattern
vcNoPattern 1276	No fill pattern
vcOutlinedDiamondPattern 2045	Diagonal cross-hatch pattern, large
vcPlaidPattern 2035	Plaid pattern
vcShinglePattern 2039	Diagonal shingle pattern
vcSmallCheckerBoardPattern 2043	Checkerboard pattern
vcSmallConfettiPattern 2028	Confetti pattern
vcSmallGridPattern 2042	Cross-hatch pattern spaced 50% closer than vcCrossPattern
vcSolidDiamondPattern 2046	Checkerboard pattern showing diagonal squares
vcSpherePattern 2041	Checkerboard of spheres
vcTrellisPattern 2040	Trellis pattern
vcVerticalBottomLightedConvexPattern 43	Vertical color gradient from dark to bright

vcVerticalConcavePattern 40	Vertical color gradient from dark to bright to dark
vcVerticalConvexPattern 41	Vertical color gradient from bright to dark to bright
vcVerticalGradientPattern 62	Vertical color gradient
vcVerticalPattern 2	Vertical lines
vcVerticalTopLightedConvexPattern 42	Vertical color gradient from bright to dark
vcWavePattern 2031	Horizontal wave pattern
vcWeavePattern 2034	Interwoven stripe pattern
vcWideDownwardDiagonalPattern 2016	Diagonal lines slanting from top left to bottom right, showing the same spacing but three times the line width of vcF-DiagonalPattern
vcWideUpwardDiagonalPattern 2017	Diagonal lines slanting from bottom left to top right right, showing the same spacing but three times the line width of vcBDiagonalPattern
vcZigZagPattern 2030	Horizontal zig-zag lines

TextFont

Property of VcBoxFormatField

This property lets you set or retrieve the font of the box format field, if it is of the type **vcFFTText**.

	Data Type	Explanation
Property value	StdFont	Font type of the box format

Example Code

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

Set boxFormatCltn = VcGantt1.BoxFormatCollection
Set boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.TextFont.Bold = True
```

TextFontColor

Property of VcBoxFormatField

This property lets you set or retrieve the font color of the box format field, if it is of the type **vcFFTText**.

	Data Type	Explanation
Property value	OLE_COLOR	Font color of the box format **Default value:** -1

Example Code

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

Set boxFormatCltn = VcGantt1.BoxFormatCollection
Set boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.TextFontColor = RGB(0, 255, 0)
```

Type

Property of VcBoxFormatField

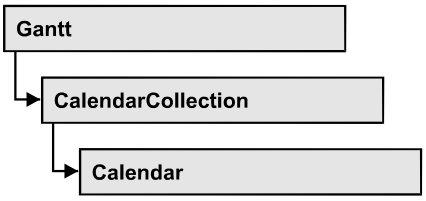
This property lets you enquire the type of the box format field.

	Data Type	Explanation
Property value	FormatFieldTypeEnum **Possible Values:** vcFFTGraphics 64 vcFFTText 36	Type of the box format field graphics text

Example Code

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

Set boxFormatCltn = VcGantt1.BoxFormatCollection
Set boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.Type = vcFFTGraphics
boxFormatField.GraphicsHeight = 200
```

7.11 VcCalendar



A calendar serves to define work and non work periods. It is composed of a continuous sequence of work and nonwork periods, that commonly are made of Workday and Workweek objects, but may also consist of intervals. A calendar just created by default contains an interval that covers the whole project. The objects of the Gantt graph, such as calendar grids, bars and layers can adopt the time pattern provided by the calendar. For example, workfree intervals can interrupt the display of the bar.

A calendar also is useful for scheduling, e.g. to calculate the number of work days between two set dates.

Properties

- CalendarProfileCollection
- IntervalCollection
- Name
- SecondsPerWorkday
- Specification
- Type

Methods

- AddDuration
- CalcDuration
- Clear
- GetEndOfPreviousWorktime
- GetNextIntervalBorder
- GetPreviousIntervalBorder
- GetStartOfInterval
- GetStartOfNextWorktime
- IsWorktime
- Update

Properties

CalendarProfileCollection

Read Only Property of VcCalendar

This property gives access to the CalenderProfileCollection object that contains all calendar profiles available in this VcCalendar object.

	Data Type	Explanation
Property value	VcCalendarProfileCollection	CalendarProfileCollection object

IntervalCollection

Read Only Property of VcCalendar

This property gives access to the IntervalCollection object that contains all intervals available.

	Data Type	Explanation
Property value	VcIntervalCollection	IntervalCollection object

Name

Read Only Property of VcCalendar

This property lets you retrieve the name of a calendar.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the calendar Name of the color map

Example Code

```
Dim calendar As VcCalendar
Dim calendarName As String

Set calendar = VcGantt1.CalendarCollection.FirstCalendar
calendarName = calendar.Name
```

SecondsPerWorkday

Property of VcCalendar

This property lets you set/retrieve the number of seconds of a workday. This feature can be also set in the **Specify Calendars** dialog.

	Data Type	Explanation
Property value	Long	Seconds of a workday

Specification

Read Only Property of VcCalendar

This property lets you retrieve the specification of a calendar. A specification is a string that contains legible ASCII characters from 32 to 127 only, so it can be stored smoothly to text files or data bases. This allows for persistency. A specification can be used to create a calendar by the method **VcCalendar-Collection.AddBySpecification**.

	Data Type	Explanation
Property value	String **Possible Values:**	Specification of the calendar Name of the color map

Type

Property of VcCalendar

This property lets you set or retrieve the calendar type. If you change the type, all properties of this calendar will be deleted.

	Data Type	Explanation
Property value	CalendarTypeEnum **Possible Values:** vcNormalCalendar 139 vcShiftCalendar 12	calendar type

Example Code

```
Dim calendarCltn As VcCalendarCollection
Dim calendar As VcCalendar

Set calendarCltn = VcGantt1.CalendarCollection
Set calendar = calendarCltn.CalendarByIndex(2)
```

```
calendar.Type = vcNormalCalendar
```

Methods

AddDuration

Method of VcCalendar

This method lets you assign a duration (work time) to a date of the calendar, considering the settings of the calendar. If e.g. you have defined workfree weekends to your calendar, a duration of three days added to a Friday will result in the Wednesday following.

	Data Type	Explanation
Parameter:		
⇨ Date	Date/Time	Date the duration is to be inserted at
⇨ Duration	Long	Number of time units (e.g.days)
Return value	Date/Time	Date the duration was inserted at

Example Code

```
Dim calendar As VcCalendar
Dim newDate As Date

Set calendar = VcGantt1.CalendarCollection.CalendarByName("WeekCalendar")
newDate = calendar.AddDuration("16.06.2014", 3)
```

CalcDuration

Method of VcCalendar

This method lets you retrieve the number of work time elements (e.g. work days) available between two defined dates. The unit (e.g. days) of the value returned is the one defined in the **Time Unit** field on the **General** property page.

	Data Type	Explanation
Parameter:		
⇨ fromDate	Date/Time	Start date of the duration that the number of work time elements is to be retrieved of
⇨ toDate	Date/Time	End date of the duration that the number of work time elements is to be retrieved of
Return value	Long	Number of time units (e.g. days) of the duration

Example Code

```
Dim calendar As VcCalendar
Dim duration As Long

Set calendar = VcGantt1.CalendarCollection.CalendarByName("WeekCalendar")
duration = calendar.CalcDuration("01.01.2013", "31.12.2014")
```

Clear

Method of VcCalendar

Removes the profiles and intervals formerly defined in this VcCalendar object, thus completely clearing it (=> 100% working time). The changes will only be displayed after an update.

	Data Type	Explanation

GetEndOfPreviousWorktime

Method of VcCalendar

This method lets you retrieve the end of the work time that precedes the reference date. The reference date has to belong to a non-working period.

	Data Type	Explanation
Parameter:		
⇨ Date	Date/Time	Date that the previous work time refers to
Return value	Date/Time	Final date of the previous work time

Example Code

```
Dim calendar As VcCalendar
Dim endOfWork As Date

Set calendar = VcGantt1.CalendarCollection.CalendarByName("WeekCalendar")
endOfWork = calendar.GetEndOfPreviousWorktime("18.06.2014")
```

GetNextIntervalBorder

Method of VcCalendar

This method lets you retrieve the beginning of the interval succeeding. If the reference date is in a non work time, the date returned will be the beginning of the succeeding work time, and vice versa.

	Data Type	Explanation
Parameter:		
⇨ Date	Date/Time	Date that the following interval border refers to
Return value	Date/Time	Start date of the interval border following

Example Code

```
Dim calendar As VcCalendar
Dim nextIntervalBorder As Date

Set calendar = VcGantt1.CalendarCollection.CalendarByName("WeekCalendar")
nextIntervalBorder = calendar.GetNextIntervalBorder("18.06.2014")
```

GetPreviousIntervalBorder

Method of VcCalendar

This method lets you retrieve the end of the preceding interval. If the reference date is in a non work time, the date returned will be the end of the preceding work time, and vice versa.

	Data Type	Explanation
Parameter:		
⇨ Date	Date/Time	Date that of the preceding interval border refers to
Return value	Date/Time	End date of the interval border preceding

Example Code

```
Dim calendar As VcCalendar
Dim previousIntervalBorder As Date

Set calendar = VcGantt1.CalendarCollection.CalendarByName("WeekCalendar")
previousIntervalBorder = calendar.GetPreviousIntervalBorder("18.06.2014")
```

GetStartOfInterval

Method of VcCalendar

This method lets you retrieve the beginning of the interval that the reference date is located in.

	Data Type	Explanation
Parameter:		
⇨ Date	Date/Time	Reference date of the interval, that the start date is to be retrieved of
Return value	Date/Time	Start date of the interval

Example Code

```
Dim calendar As VcCalendar
Dim startOfInterval As Date

Set calendar = VcGantt1.CalendarCollection.CalendarByName("WeekCalendar")
startOfInterval = calendar.GetStartOfInterval("18.06.2014")
```

GetStartOfNextWorktime

Method of VcCalendar

This method lets you retrieve the beginning of the work time that succeeds the reference date.

	Data Type	Explanation
Parameter: ⇨ Date	Date/Time	Reference date, that the start date of the work time following is to be retrieved of
Return value	Date/Time	Start date of the work time following

Example Code

```
Dim calendar As VcCalendar
Dim startOfNextWorktime As Date

Set calendar = VcGantt1.CalendarCollection.CalendarByName("WeekCalendar")
startOfNextWorktime = calendar.GetStartOfNextWorktime("18.06.2014")
```

IsWorktime

Method of VcCalendar

This method lets you enquire whether or not the date passed is in a work time.

	Data Type	Explanation
Parameter: ⇨ Date	Date/Time	Date to be checked for being a work time
Return value	Boolean	Date passed does /does not belong to a work time

Example Code

```
Dim calendar As VcCalendar
Dim isWorktime As Boolean

Set calendar = VcGantt1.CalendarCollection.CalendarByName("WeekCalendar")
isWorktime = calendar.isWorktime("18.06.2014")
```

Update

Method of VcCalendar

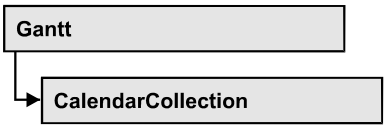
This method lets you update a calendar after having modified it. It ensures other objects that use calendar (e.g. a calendarGrid) to be updated as well.

	Data Type	Explanation
Return value	Void	

Example Code

```
Dim calendar As VcCalendar

Set calendar = VcGantt1.CalendarCollection.CalendarByName("WeekCalendar")
calendar.Update
```

7.12 VcCalendarCollection



An object of the type VcCalendarCollection automatically contains all available calendars. You can access all objects in an iterative loop by **For Each calendar In CalendarCollection** or by the methods **First...** and **Next...**. You can access a single calendar by the methods **CalendarByName** and **CalendarByIndex**. The number of calendars in the collection object can be retrieved by the property **Count**. By the property **Active** you can set or retrieve the calendar which controls the calendar grid.

Properties

- _NewEnum
- Active
- Count

Methods

- Add
- AddBySpecification
- CalendarByIndex
- CalendarByName
- Copy
- FirstCalendar
- NextCalendar
- Remove
- Update

Properties

_NewEnum

Read Only Property of VcCalendarCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all calender objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method

GetEnumerator is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim calendar As VcCalendar

For Each calendar In VcGantt1.CalendarCollection
   Debug.Print calendar.Name
Next
```

Active

Property of VcCalendarCollection

This property lets you set or retrieve the default calendar that is used by nodes, if no other calendar was assigned.

	Data Type	Explanation
Property value	VcCalendar	Calendar currently used

Example Code

```
Dim workday As VcWorkday
Dim freeday As VcWorkday
Dim workweek As VcWorkweek
Dim calendarCltn As VcCalendarCollection
Dim calendar As VcCalendar

Set workday = VcGantt1.WorkdayCollection.CreateWorkday("Work day")
workday.AddNonWorkInterval "00:00:00", "00:00:00"
workday.AddWorkInterval "08:00:00", "16:30:00"

Set freeday = VcGantt1.WorkdayCollection.CreateWorkday("Workfree day")
freeday.AddNonWorkInterval "00:00:00", "00:00:00"

Set calendarCltn = VcGantt1.calendarcollection
Set calendar = calendarCltn.AddCalendar("New Calendar")

Set workweek = VcGantt1.WorkweekCollection.CreateWorkweek("Work week")
workweek.AddWorkday workday, vcMonday, vcFriday
workweek.AddWorkday freeday, vcSaturday, vcSunday

calendar.AddWorkweek workweek, "01.01.13", "31.12.14"

calendar.Update

Set calendarCltn.Active = calendar
```

Count

Read Only Property of VcCalendarCollection

This property lets you retrieve the number of calendars in the calendar collection.

	Data Type	Explanation
Property value	Long	Number of calendars

Methods

Add

Method of VcCalendarCollection

By this method you can create a calendar as a member of the CalendarCollection. If the name has not been used before, the new calendar object will be returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
Parameter: ⇨ calendarName	String **Possible Values:**	Calendar name Name of the color map
Return value	VcCalendar	New calendar object

AddBySpecification

Method of VcCalendarCollection

This method lets you create a calendar by using a calendar specification. This way of creating allows calendar objects to become persistent. The specification of a calendar can be saved and re-loaded (see VcCalendar property **Specification**). In a subsequent the calendar can be created again from the specification and is identified by its name.

	Data Type	Explanation
Parameter: ⇨ Specification	String **Possible Values:**	Calendar specification Name of the color map
Return value	VcCalendar	New calendar object

CalendarByIndex

Method of VcCalendarCollection

This method lets you access a calendar by its index. If no calendar of the specified index does exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter: ⇨ index	Integer **Possible Values:**	Index of the calendar Data field index
Return value	VcCalendar	Calendar object returned

CalendarByName

Method of VcCalendarCollection

By this method you can retrieve a calendar by its name. If a calendar of the specified name does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter: ⇨ CalendarName	String **Possible Values:**	Name of the calendar Name of the color map
Return value	VcCalendar	Calendar

Example Code

```
Dim calendarCltn As VcCalendarCollection
```

```
Set calendarCltn = VcGantt1.CalendarCollection
calendarCltn.Active = calendarCollection.CalendarByName("Calendar_1")
```

Copy

Method of VcCalendarCollection

By this method you can copy a calendar. If the calendar that is to be copied exists, and if the name for the new calendar does not yet exist, the new calendar object is returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
Parameter:		
⇨ calendarName	String **Possible Values:**	Name of the calendar to be copied Name of the color map
⇨ newCalendarName	String **Possible Values:**	Name of the calendar Name of the color map
Return value	VcCalendar	Calendar object

FirstCalendar

Method of VcCalendarCollection

This method can be used to access the initial value, i.e. the first calendar of a calendar collection, and then to continue in a forward iteration loop by the method **NextCalendar** for the calendars following. If there is no calendar in the FilterCollection object, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcCalendar	First calendar

NextCalendar

Method of VcCalendarCollection

This method can be used in a forward iteration loop to retrieve subsequent calendars from a calendar collection after initializing the loop by the method

FirstCalendar. If there is no calendar left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcCalendar	Subsequent calendar

Example Code

```
Dim calendarCltn As VcCalendarCollection
Dim calendar As VcCalendar

Set calendarCltn = VcGantt1.CalendarCollection
Set calendar = calendarCltn.FirstCalendar

While Not calendar Is Nothing
    List1.AddItem (calendar.Name)
    Set calendar = calendarCltn.NextCalendar
Wend
```

Remove

Method of VcCalendarCollection

This method lets you delete a calendar. If the calendar is used in another object, it cannot be deleted. Then False will be returned, otherwise True.

	Data Type	Explanation
Return value	Boolean	Calendar deleted (True)/not deleted (False)

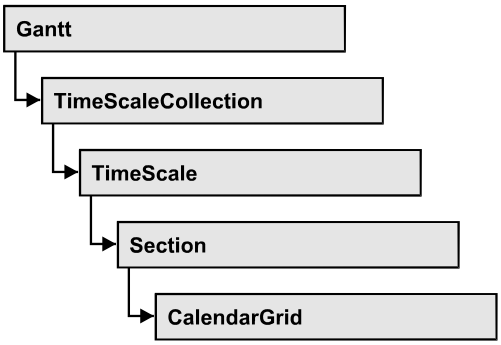
Update

Method of VcCalendarCollection

This method lets you update a calendar collection after having modified it.

	Data Type	Explanation
Return value	Boolean	update successful (True)/ not successful (False)

7.13 VcCalendarGrid



An object of the type **VcCalendarGrid** is a grid of vertical lines to highlight workfree periods by colored vertical areas.

Properties

- BackColorAsARGB
- BackColorDataFieldIndex
- BackColorMapName
- CalendarName
- CalendarNameDataFieldIndex
- CalendarNameMapName
- EndSnapTarget
- Identifiable
- LineColor
- LineColorDataFieldIndex
- LineColorMapName
- LineThickness
- LineType
- Name
- Pattern
- PatternColorAsARGB
- PatternColorDataFieldIndex
- PatternColorMapName
- PatternDataFieldIndex
- PatternMapName
- Priority
- SnapTarget
- Specification
- StartSnapTarget
- UseGraphicalAttributesOfIntervals

- Visible
- VisibleDataFieldIndex
- VisibleMapName

Methods

- IdentifyInterval
- IdentifyIntervalAsVariant

Properties

BackColorAsARGB

Property of VcCalendarGrid

This property lets you set or retrieve the color of the background of the calendar grid. Color values have a transparency or alpha value, followed by a value for a red, a blue and a green partition (ARGB). The values range between 0..255. An alpha value of 0 equals complete transparency, whereas 255 represents a completely solid color. When casting an RGB value on an ARGB value, an alpha value of 255 has to be added.

Also see **set/getPatternColorAsARGB**.

	Data Type	Explanation
Property value	Integer	ARGB color values ({0...255},{0...255},{0...255},{0...255}) **Default value:** &hFFD8D8D8 (gray)
	Possible Values:	Data field index

Example Code

```
Dim section As VcSection
Dim calendarGrid As VcCalendarGrid

Set section = VcGantt1.TimeScaleCollection.Active.Section(0)
Set calendarGrid = section.CalendarGrid
calendarGrid.BackgroundColorAsARGB = &h88FF0A06
```

BackColorDataFieldIndex

Property of VcCalendarGrid

This property lets you set or retrieve the data field index to be used with a color map specified by the property **BackColorMapName**. If you set this property to **-1**, no map will be used.

	Data Type	Explanation
Property value	Integer	Data field index
	Possible Values:	Data field index

BackColorMapName

Property of VcCalendarGrid

This property lets you set or retrieve the name of a color map (type vcColorMap). If set to "", no map will be used. If a map name and additionally a data field index is specified in the property **BackColorDataFieldIndex**, then the background color is controlled by the map. If no data field entry applies, the background color that is specified in the property **BackColor** will be used.

	Data Type	Explanation
Property value	String	Name of the color map
	Possible Values:	Name of the color map

CalendarName

Property of VcCalendarGrid

This property lets you assign a calendar to the calendar grid to highlight the calendar's workfree periods.

	Data Type	Explanation
Property value	String	Character string that passes the calendar name
	Possible Values:	Name of the color map

CalendarNameDataFieldIndex

Property of VcCalendarGrid

This property lets you set or retrieve the index of the data field to store the name of the calendar if you wish to use an individual calendar for a grouping level. This is only possible as long as no data has been loaded. This property also can be set on the **Calendar** property page.

	Data Type	Explanation
Property value	Long	Index of the data field which contains the name of the calendar

CalendarNameMapName

Property of VcCalendarGrid

This property lets you set or retrieve the name of a calendar map (type vcTextMap). If set to "", no map will be used. If a map name and additionally a data field index is specified by the property **CalendarNameDataFieldIndex**, the calendar is selected by the map. If no data field entry applies, the calendar that was assigned to the calendar grid of the grouping level will be used.

	Data Type	Explanation
Property value	String	Name of the calendar map
	Possible Values:	Name of the color map

EndSnapTarget

Property of VcCalendarGrid

This property lets you set or retrieve whether the end date of this calendar is to define as snap target.

	Data Type	Explanation
Property value	Boolean	End date of this calendar grid is/is not defined as snap target
	Possible Values:	Group invisible/visible group nodes are/are not visible

Identifiable

Property of VcCalendarGrid

This property lets you set or retrieve whether or not a calendar grid can be identified. If this property was set to **True**, the calendar grid can be identified by the VcGantt method **IdentifyObjectAt**. Also, a tooltip text requested by **OnTooltipText** will only appear if this property was set to **True**. In the same way, the **OnCalendarGridRClick** event will only be triggered if the calendar grid is identifiable.

To produce specific tooltip texts, in addition the corresponding intervals of a calendar need to be identified: see VcGantt method **IdentifyInterval**.

This property can also be set in the **calendar grid** section of the **Edit time scale section** dialog.

	Data Type	Explanation
Property value	Boolean	Calendar grid can / cannot be identified **Default value:** False
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
Dim section As VcSection
Dim calendarGrid As VcCalendarGrid

Set section = VcGantt1.TimeScaleCollection.Active.Section(0)
Set calendarGrid = section.CalendarGrid
calendarGrid.Identifiable = True
```

LineColor

Property of VcCalendarGrid

This property lets you set or retrieve the line color of a calendar grid and can also be set in the **Line attributes of calendar grid** dialog.

	Data Type	Explanation
Property value	Color	RGB color values ({0...255},{0...255},{0...255}) **Default value:** As defined in the dialog

LineColorDataFieldIndex

Property of VcCalendarGrid

This property lets you set or retrieve the data field index to be used with a map specified by the property **LineColorMapName**. If you set this property to **-1**, no map will be used.

	Data Type	Explanation
Property value	Integer	Data field index
	Possible Values:	Data field index

LineColorMapName

Property of VcCalendarGrid

This property lets you set or retrieve the name of a map for the line color. If set to "" or if the property **LineColorDataFieldIndex** is set to **-1**, then no map will be used.

	Data Type	Explanation
Property value	String	Name of the color map
	Possible Values:	Name of the color map

LineThickness

Property of VcCalendarGrid

This property lets you set or retrieve the line thickness of the calendar grid lines.

If you set this property to values between 1 and 4, an absolute line thickness is defined in pixels. Irrespective of the zoom factor a line will always show the same line thickness in pixels. When printing though, the line thickness is adapted for the sake of legibility and becomes dependent of the zoom factor:

Value	Points	mm
1	1/2 point	0.09 mm
2	1 point	0.18 mm

Value	Points	mm
3	3/2 points	0.26 mm
4	2 points	0.35 mm

A point equals 1/72 inch and represents the unit of the font size.

If you set this property to values between 5 and 1,000, the line thickness is defined in 1/100 mm, so the lines will be displayed in a true thickness in pixels that depends on the zoom factor.

This property also can be set in the **Attributes of calendar grid** dialog.

	Data Type	Explanation
Property value	Integer	Line thickness LineType {1...4}: line thickness in pixels LineType {5...1000}: line thickness in 1/100 mm **Default value:** As defined in the dialog
	Possible Values:	Data field index

LineType

Property of VcCalendarGrid

This property lets you set or retrieve the line type of a calendar grid.

This property also can be set in the **Attributes of calendar grid** dialog.

	Data Type	Explanation
Property value	LineTypeEnum	Line type
	Possible Values:	
	vcDashed 4	Line dashed
	vcDashedDotted 5	Line dashed-dotted
	vcDotted 3	Line dotted
	vcLineType0 100	Line Type 0
	vcLineType1 101	Line Type 1
	vcLineType10 110	Line Type 10
	vcLineType11 111	Line Type 11

vcLineType12 112	Line Type 12
vcLineType13 113	Line Type 13
vcLineType14 114	Line Type 14
vcLineType15 115	Line Type 15
vcLineType16 116	Line Type 16
vcLineType17 117	Line Type 17
vcLineType18 118	Line Type 18
vcLineType2 102	Line Type 2
vcLineType3 103	Line Type 3
vcLineType4 104	Line Type 4
vcLineType5 105	Line Type 5
vcLineType6 106	Line Type 6
vcLineType7 107	Line Type 7
vcLineType8 108	Line Type 8
vcLineType9 109	Line Type 9
vcNone 1	No line type
vcNotSet -1	No line type assigned
vcSolid 2	Line solid

Name

Property of VcCalendarGrid

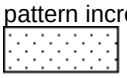
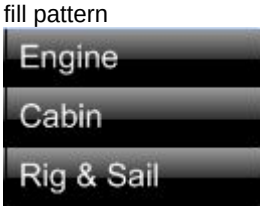
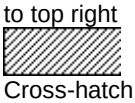
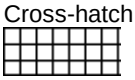
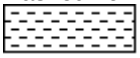
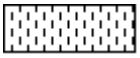
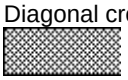
This property lets you set or retrieve the name of a calendar grid.

	Data Type	Explanation
Property value	String	Name of the calendar grid
	Possible Values:	Name of the color map

Pattern

Property of VcCalendarGrid

This property lets you set or retrieve the pattern of the calendar grid. Also see **set/getPatternColor**.

	Data Type	Explanation
Property value	FillPatternEnum	Pattern type
	Possible Values:	
	vc05PercentPattern... vc90PercentPattern 01 - 11	Dots in foreground color on background color, the density of the foreground pattern increasing with the percentage 
	vcAeroGlassPattern 40	Vertical color gradient in the color of the fill pattern 
	vcBDiagonalPattern 5	Diagonal lines slanting from bottom left to top right 
	vcCrossPattern 6	Cross-hatch pattern 
	vcDarkDownwardDiagonalPattern 2014	Diagonal lines slanting from top left to bottom right; spaced 50% closer than vcFDiagonalPattern and of twice the line width 
	vcDarkHorizontalPattern 2023	Horizontal lines spaced 50% closer than vcHorizontalPattern and of twice the line width 
	vcDarkUpwardDiagonalPattern 2015	Diagonal lines slanting from bottom left to top right, spaced 50% closer than vcBDiagonalPattern and of twice the line width 
	vcDarkVerticalPattern 2022	Vertical lines spaced 50% closer than vcVerticalPattern and of of twice the line width 
	vcDashedDownwardDiagonalPattern 2024	Dashed diagonal lines from top left to bottom right 
	vcDashedHorizontalPattern 2026	Dashed horizontal lines 
	vcDashedUpwardDiagonalPattern 2025	Dashed diagonal lines from bottom left to top right 
	vcDashedVerticalPattern 2027	Dashed vertical lines 
	vcDiagCrossPattern 7	Diagonal cross-hatch pattern, small 
	vcDiagonalBrickPattern 2032	Diagonal brick pattern 

vcDivotPattern 2036	Divot pattern
vcDottedDiamondPattern 2038	Diagonal cross-hatch pattern of dotted lines
vcDottedGridPattern 2037	Cross-hatch pattern of dotted lines
vcFDiagonalPattern 4	Diagonal lines slanting from top left to bottom right
vcHorizontalBrickPattern 2033	Horizontal brick pattern
vcHorizontalGradientPattern 52	Horizontal color gradient
vcHorizontalPattern 3	Horizontal lines
vcLargeCheckerboardPattern 2044	Checkerboard pattern showing squares of twice the size of vcSmallChecker-BoardPattern
vcLargeConfettiPattern 2029	Confetti pattern, large
vcLightDownwardDiagonalPattern 2012	Diagonal lines slanting to from top left to bottom right; spaced 50% closer than vcBDiagonalPattern
vcLightHorizontalPattern 2019	Horizontal lines spaced 50% closer than vcHorizontalPattern
vcLightUpwardDiagonalPattern 2013	Diagonal lines slanting from bottom left to top right, spaced 50% closer than vcBDiagonalPattern
vcLightVerticalPattern 2018	Vertical lines spaced 50% closer than vcVerticalPattern
vcNarrowHorizontalPattern 2021	Horizontal lines spaced 75 % closer than vcHorizontalPattern
vcNarrowVerticalPattern 2020	Vertical lines spaced 75% closer than vcVerticalPattern
vcNoPattern 1276	No fill pattern
vcOutlinedDiamondPattern 2045	Diagonal cross-hatch pattern, large
vcPlaidPattern 2035	Plaid pattern
vcShinglePattern 2039	Diagonal shingle pattern
vcSmallCheckerBoardPattern 2043	Checkerboard pattern

vcSmallConfettiPattern 2028	Confetti pattern
vcSmallGridPattern 2042	Cross-hatch pattern spaced 50% closer than vcCrossPattern
vcSolidDiamondPattern 2046	Checkerboard pattern showing diagonal squares
vcSpherePattern 2041	Checkerboard of spheres
vcTrellisPattern 2040	Trellis pattern
vcVerticalBottomLightedConvexPattern 43	Vertical color gradient from dark to bright
vcVerticalConcavePattern 40	Vertical color gradient from dark to bright to dark
vcVerticalConvexPattern 41	Vertical color gradient from bright to dark to bright
vcVerticalGradientPattern 62	Vertical color gradient
vcVerticalPattern 2	Vertical lines
vcVerticalTopLightedConvexPattern 42	Vertical color gradient from bright to dark
vcWavePattern 2031	Horizontal wave pattern
vcWeavePattern 2034	Interwoven stripe pattern
vcWideDownwardDiagonalPattern 2016	Diagonal lines slanting from top left to bottom right, showing the same spacing but three times the line width of vcF-DiagonalPattern
vcWideUpwardDiagonalPattern 2017	Diagonal lines slanting from bottom left to top right right, showing the same spacing but three times the line width of vcBDiagonalPattern
vcZigZagPattern 2030	Horizontal zig-zag lines

PatternColorAsARGB

Property of VcCalendarGrid

This property lets you set or retrieve the pattern color of the calendar grid. Color values have a transparency or alpha value, followed by a value for a red, a blue and a green partition (ARGB). The values range between 0..255. An alpha value of 0 equals complete transparency, whereas 255 represents a completely solid color. When casting an RGB value on an ARGB value, an alpha value of 255 has to be added.

Also see **set/getBackgroundColor** and **set/getPattern**.

	Data Type	Explanation
Property value	Integer	ARGB color values ({0...255},{0...255},{0...255},{0...255})
	Possible Values:	Data field index

Example Code

```
Dim section As VcSection
Dim calendarGrid As VcCalendarGrid

Set section = VcGantt1.TimeScaleCollection.Active.Section(0)
Set calendarGrid = section.CalendarGrid
calendarGrid.PatternColorAsARGB = &h88FF0A06
```

PatternColorDataFieldIndex

Read Only Property of VcCalendarGrid

This property lets you set or retrieve the data field index that has to be specified if the property **PatternColorMapName** is used. If you set this property to **-1**, no map will be used.

	Data Type	Explanation
Property value	Integer	Data field index
	Possible Values:	Data field index

PatternColorMapName

Property of VcCalendarGrid

This property lets you set or retrieve the name of a color map (type vcColorMap). If set to "", no map will be used. Only if a map name and a data field index are specified in the property **PatternColorDataFieldIndex**, the pattern color is controlled by the map. If no data field entry applies, the pattern color of the calendar grid that is specified in the property **PatternColor** will be used.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the color map Name of the color map

PatternDataFieldIndex

Property of VcCalendarGrid

This property lets you set or retrieve the data field index to be used together with the property **PatternMapName**. If you set this property to **-1**, no map will be used.

	Data Type	Explanation
Property value	Integer **Possible Values:**	Data field index Data field index

PatternMapName

Property of VcCalendarGrid

This property lets you set or retrieve the name of a pattern map (type vcPatternMap). If set to "", no map will be used. Only if a map name and additionally a data field index are specified in the property **PatternData-FieldIndex**, the pattern is controlled by the map. If no data field entry applies, the pattern of the layer that is specified in the property **Pattern** will be used.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the pattern map Name of the color map

Priority

Property of VcCalendarGrid

This property lets you set or retrieve the priority of the calendar grid. If two objects are located in the same position of the diagram, the object of higher priority is displayed in front of objects of lower priority. By default, calendar grid lines are of lowest priority. Nodes are assigned the value 0 and thus have the highest priority of all objects. If you want a calendar grid to be displayed in front of the nodes, its priority needs to be set to a positive value.

	Data Type	Explanation
Property value	Integer **Possible Values:**	Rank of Priority {-100 ... 100} **Default value:** -20 Data field index

Example Code

```
Dim section As VcSection
Dim calendarGrid As VcCalendarGrid

Set section = VcGantt1.TimeScaleCollection.Active.Section(0)
Set calendarGrid = section.CalendarGrid
calendarGrid.Priority = 3
```

SnapTarget

Property of VcCalendarGrid

This property lets you set or retrieve whether this calendar grid has a snap target at the date.

	Data Type	Explanation

Specification

Read Only Property of VcCalendarGrid

This property lets you retrieve the specification of a calendar grid. A specification is a string that contains legible ASCII characters from 32 to 127 only, so it can be stored smoothly to text files or data bases. This allows for persistency. A specification can be used to create a calendar grid by the method **VcCalendarGridCollection.AddBySpecification**.

	Data Type	Explanation

StartSnapTarget

Property of VcCalendarGrid

This property lets you set or retrieve whether the start date of this calendar is to define as snap target.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	Start date of this calendar grid is/is not defined as snap target Group invisible/visible group nodes are/are not visible

UseGraphicalAttributesOfIntervals

Property of VcCalendarGrid

This property lets you set or retrieve whether the graphical attributes that have been set for the intervals are to be displayed. This feature can be also set in the dialog **Administrate Intervals** (which you reach by clicking ··· in the **Specify Calendar** dialog). If this property is set to **False**, the settings of the property **VcInterval.UseGraphicalAttributes** have no effect.

	Data Type	Explanation

Visible

Property of VcCalendarGrid

This property lets you set or retrieve whether a calendar grid is visible.

	Data Type	Explanation
Property value	Boolean	Calendar grid visible/invisible **Default value:** True
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
Dim section As VcSection
Dim calendarGrid As VcCalendarGrid

Set section = VcGantt1.TimeScaleCollection.Active.Section(0)
Set calendarGrid = section.CalendarGrid
calendarGrid.Visible = True
```

VisibleDataFieldIndex

Property of VcCalendarGrid

This property lets you set or retrieve the index of the data field to assigne a visibility mode to the calendar grid: 1 (for "visible") or 0 (for invisible). This property also can be set in the **Calendar grid** dialog.

	Data Type	Explanation
Property value	Long	Index of the data field which contains the visibility mode

VisibleMapName

Property of VcCalendarGrid

This property lets you set or retrieve the name of a map (type vcTextMap) to set the visibility mode. If set to "", no map will be used. If a map name and additionally a data field index is specified by the property **VisibilityDataFieldIndex**, the visibility mode is selected by the map. If no data field entry applies, the calendar grid will be set to "visible". This property also can be set in the **CalendarGrid** dialog.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the visibility map Name of the color map

Methods

IdentifyInterval

Method of VcCalendarGrid

This method lets you identify an interval object of the calendar that was assigned to the calendar grid at the coordinates passed. Since usually copies of intervals exist in a calendar, intervals tend not to be unique (for instance, the same weekend interval may repeat 52 times per year). Therefore the method also returns the start and end dates of the interval retrieved.

This method is useful when being invoked within a tooltip event to return the interval at the position of the mouse cursor.

If there is an interval at the position specified, **True** will be returned, if there isn't, the method will deliver **False**.

Please Note: If you are coding in VBScript, you will have to use the analogous method **IdentifyIntervalAsVariant** because of the by-reference parameters .

	Data Type	Explanation
Parameter:		
⇨ x	Long	X coordinate of the cursor
⇨ y	Long	Y coordinate of the cursor
⇦ identifiedIntervalParam	VcInterval	Interval identified
⇦ startDateParam	Date/Time	Start date of the interval identified
⇦ endDateParam	Date/Time	End date of the interval identified
Return value	Boolean	Interval could / could not be identified

Example Code

```
Private Sub VcGantt1_DragDrop(Source As Control, X As Single, Y As Single)
```

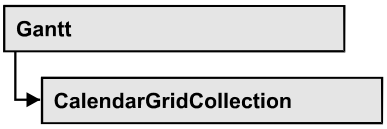
```
    Dim identifiedObj As Object
    Dim identifiedObjType As VcObjectTypeEnum
    Dim identifiedIntervalParam As VcInterval
    Dim startDateParam As System.Date
    Dim endDateParam As System.Date
    Dim xPix, yPix As Long

    xPix = X / Screen.TwipsPerPixelX
    yPix = Y / Screen.TwipsPerPixelY

     Call VcGantt1.IdentifyInterval(xPix, yPix, identifiedIntervalParam, _
                                    startDateParam, endDateParam)
            If Not identifiedIntervalParam Is Nothing Then
                MsgBox ("This Interval """ + identifiedIntervalParam() + _
                        """, ranges from """ + startDateParam + _
                        """, to '""'+endDateParam")
            Else
                MsgBox ("At this position no interval was identified.")
            End If

End Sub
```

IdentifyIntervalAsVariant

Method of VcCalendarGrid

This method is identical to the method **IdentifyInterval** except for the parameters. It was necessary to implement a separate method because some languages (e.g. VBScript) can use by-reference parameters (marked by ⇦) only if the type of these parameters is VARIANT.

7.14 VcCalendarGridCollection



An object of the type VcCalendarGridCollection contains all available calendar grids. You can access all objects in an iterative loop by **For Each calendarGrid In CalendarGridCollection** or by the methods **First...** and **Next...**. You can access a single calendar grid using the methods **CalendarGridByName** and **CalendarGridByIndex**. The number of calendar grids in the collection object can be retrieved by the property **Count**. The methods **Add**, **Copy** and **Remove** allow to handle the calendar grids in the corresponding way.

Properties

- _NewEnum
- Count

Methods

- Add
- AddBySpecification
- CalendarGridByIndex
- CalendarGridByName
- Copy
- FirstCalendarGrid
- NextCalendarGrid
- Remove
- Update

Properties

_NewEnum

Read Only Property of VcCalendarGridCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all calendar grid objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method

GetEnumerator is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim calendarGrid As VcCalendarGrid

For Each calendarGrid In VcGantt1.CalendarGrid
   Debug.Print calendarGrid.Count
Next
```

Count

Read Only Property of VcCalendarGridCollection

This property lets you retrieve the number of calendar grids in the CalendarGridCollection object.

	Data Type	Explanation
Property value	Long	Number of calendar grids

Example Code

```
Dim  calendarGridCltn As Vc CalendarGridCollection
Dim numberOfCalendarGrids As Long

Set calendarGridCltn = VcGantt1.CalendarGridCollection
numberOfCalendarGrids = calendarGridCltn.Count
```

Methods

Add

Method of VcCalendarGridCollection

This method lets you create a calendar grid as a member of the CalendarGridCollection. If the name was not used before, the new calendar grid object will be returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
Parameter:		
⇨ calendarGridName	String **Possible Values:**	name of calendar grid

		Name of the color map
Return value	VcCalendarGrid	New calendar grid object

Example Code

```
Set newCalendarGrid = VcGantt1.CalendarGridCollection.Add("Grid1")
```

AddBySpecification

Method of VcCalendarGridCollection

This method lets you create a calendar grid by using a calendar grid specification. This way of creating allows calendar grid objects to become persistent. The specification of a calendar grid can be saved and re-loaded (see VcCalendarGrid property **Specification**). In a subsequent session the calendar grid can be created again from the specification and is identified by its name.

	Data Type	**Explanation**
Parameter: ⇨ specification	String **Possible Values:**	calendar grid specification Name of the color map
Return value	VcCalendarGrid	New calendar grid object

CalendarGridByIndex

Method of VcCalendarGridCollection

This method lets you access a calendar grid by its index. If a calendar grid of the specified index does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	**Explanation**
Parameter: ⇨ index	Integer **Possible Values:**	Index of the calendar grid Data field index
Return value	VcCalendarGrid	calendar grid object returned

Example Code

```
Dim calendarGridCltn As VcCalendarGrid
```

```
Dim calendar As VcCalendar

Set calendarGridCltn = VcGantt1.CalendarGrid
Set calendarGrid = calendarGridCltn.CalendarGridByIndex(2)
MsgBox calendarGrid.Name
```

CalendarGridByName

Method of VcCalendarGridCollection

This method is used to access a calendar grid by its name. If a calendar grid of the specified name does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter:		
⇨ calendarGridName	String **Possible Values:**	Name of the calendar grid Name of the color map
Return value	VcCalendarGrid	calendar grid

Example Code

```
Dim calendarGridCltn As VcCalendarGridCollection
Dim calendarGrid As VcCalendarGrid

Set calendarGridCltn = VcGantt1.CalendarGridCollection
Set calendarGrid = calendarGridrCltn.CalendarGridByName("Grid 4")
```

Copy

Method of VcCalendarGridCollection

By this method you can copy a calendar grid. If the calendar grid that is to be copied exists, and if the name for the new calendar grid does not yet exist, the new calendar grid object is returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
Parameter:		
⇨ calendarGridName	String **Possible Values:**	Name of the calendar grid to be copied Name of the color map
⇨ newCalendarGridName	String **Possible Values:**	Name of the new calendar grid Name of the color map
Return value	VcCalendarGrid	calendar grid object

Example Code

```
Dim calendarGridCltn As VcCalendarGridCollection
Dim calendarGrid As VcCalendarGrid

Set calendarGridCltn = VcGantt1.CalendarGridCollection
Set calendarGrid = calendarGridCltn.Copy("CurrentCalendarGrid",
"NewCalendarGrid")
```

FirstCalendarGrid

Method of VcCalendarGridCollection

This method can be used to access the initial value, i.e. the first calendar grid of a calendar grid collection and then to continue in a forward iteration loop by the method **NextCalendarGrid** for the calendar grids following. If there is no calendar grid in the CalendarGridCollection, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcCalendarGrid	First calendar grid

Example Code

```
Dim calendarGridCltn As VcCalendarGridCollection
Dim calendarGrid As VcCalendarGrid

Set calendarGridCltn = VcGantt1.CalendarGridCollection
calendarGridCltn.CalendarGrids (vcAnyCalendarGrid)
Set calendarGrid = calendarGridCltn.FirstCalendarGrid
```

NextCalendarGrid

Method of VcCalendarGridCollection

This method can be used in a forward iteration loop to retrieve subsequent calendar grids from a CalendarGridCollection after initializing the loop by the method **FirstCalendarGrid**. If there is no calendar grid left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcCalendarGrid	Subsequent calendar grid

Example Code

```
Dim calendarGridCltn As VcCalendarGridCollection
Dim calendarGrid As VcCalendarGrid

Set calendarGridCltn = VcGantt1.CalendarGridCollection
Set calendarGrid = calendarGridrCltn.FirstCalendarGrid

While Not calendarGrid Is Nothing
```

```
   Listbox.AddItem calendarGrid.Name
   Set calendarGrid = calendarGridCltn.NextCalendarGrid
Wend
```

Remove

Method of VcCalendarGridCollection

This method lets you delete a calendar grid. If the calendar grid is used in another object, it cannot be deleted. Then False will be returned, otherwise True.

	Data Type	Explanation
Parameter:		
⇨ calendarGridName	String	Calendar grid name
	Possible Values:	Name of the color map
Return value	Boolean	Calendar grid deleted (True)/not deleted (False)

Example Code

```
Dim calendarGridCltn As VcCalendarGridCollection
Dim calendarGrid As VcCalendarGrid

Set calendarGridCltn = VcGantt1.CalendarGridCollection
Set calendarGrid = calendarGridCltn.FormatByIndex(1)
calendarGridCltn.Remove (calendarGrid.Name)
```

Update

Method of VcCalendarGridCollection

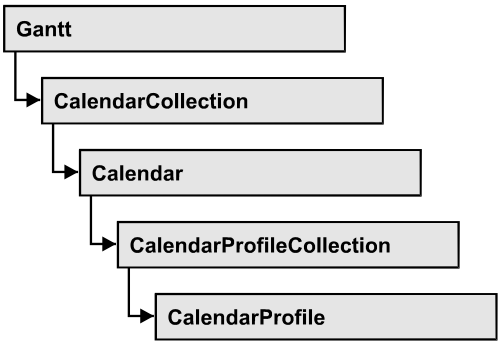
This method has to be used when calendar grid modifications have been carried out. The method **Update** updates all objects that are concerned by the calendar grid you have edited. You should call this method at the end of the code that defines the calendar grids and the calendar grid collection. Otherwise the update will be processed before all calendar grid definitions are processed.

	Data Type	Explanation
Return value	Boolean	update successful (True)/ not successful (False)

Example Code

```
Dim calendarGrid As VcCalendarGrid

Set calendarGrid = VcGantt1.CalendarGrid.Collection.CalendarGridByName("Grid 3")
calendarGrid.Update
```

7.15 VcCalendarProfile



An object of the type **VcCalendarProfile** designates a calendar profile.

Properties

- IntervalCollection
- Name
- Specification
- Type

Methods

- PutInOrderAfter

Properties

IntervalCollection

Read Only Property of VcCalendarProfile

This property gives access to the IntervalCollection object that contains all intervals available.

	Data Type	Explanation
Property value	VcIntervalCollection	IntervalCollection object

Name

Read Only Property of VcCalendarProfile

This property lets you set or retrieve the name of a calendar profile.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the calendar profile Name of the color map

Specification

Read Only Property of VcCalendarProfile

This property lets you retrieve the specification of a calendar profile. A specification is a string that contains legible ASCII characters from 32 to 127 only, so it can be stored smoothly to text files or data bases. This allows for persistency. A specification can be used to create a calendar profile by the method **VcCalendarProfileCollection.AddBySpecification**.

	Data Type	Explanation
Property value	String **Possible Values:**	Specification of the calendar profile Name of the color map

Type

Property of VcCalendarProfile

This property lets you set or retrieve the calendar profile type. If you change the type, all properties of this calendar profile will be deleted.

	Data Type	Explanation
Property value	CalendarProfileTypeEnum **Possible Values:** vcDayProfile 4 vcShiftProfile 5 vcWeekProfile 3 vcYearProfile 2	Type of the calendar profile

Methods

PutInOrderAfter

Method of VcCalendarProfile

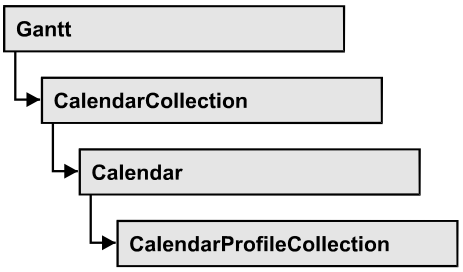
This method lets you set the calendar profile behind the calendar profile specified by name, within the CalendarProfileCollection. If you set the name to "", the calendar profile will be put in the first position. The order of the calendar profiles within the collection determines the order by which they apply to the calendars.

	Data Type	Explanation
Parameter:		
refNameParam	String	Name of the calendar profile behind which the current calendar profileis to be put.
	Possible Values:	Name of the color map

Example Code

```
Dim calProfCltn As VcCalendarProfileCollection
Dim calProf1 As VcCalendarProfile
Dim calProf2 As VcCalendarProfile

calProfCltn = VcGantt1.CalendarProfileCollection()
calProf1 = calProfCltn.Add("calProf1")
calProf2 = calProfCltn.Add("calProf2")
calProf1.PutInOrderAfter("calProf2")
calProfCltn.Update()
```

7.16 VcCalendarProfileCollection



An object of the type VcCalendarProfileCollection automatically contains all available calendar profiles. You can access all objects in an iterative loop by **For Each calendarProfile In CalendarProfileCollection** or by the methods **First...** and **Next...**. You can access a single calendar profile using the methods **CalendarProfileByName** and **CalendarProfileByIndex**. The number of calendar profiles in the collection object can be retrieved by the property **Count**. The methods **Add**, **Copy** and **Remove** allow to handle the calendar profiles in the corresponding way.

Properties

- _NewEnum
- Count

Methods

- Add
- AddBySpecification
- CalendarProfileByIndex
- CalendarProfileByName
- Copy
- FirstCalendarProfile
- NextCalendarProfile
- Remove
- SelectCalendarProfiles
- Update

Properties

_NewEnum

Property of VcCalendarProfileCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all calendar profile objects contained. In Visual Basic this property never is displayed, but it can be addressed by the command **For Each** *element* **In** *collection*. In .NET languages the method GetEnumerator is offered instead. Some development environments replace this property by own language constructs.

	Data Type	Explanation
Property value	Object	Reference object

Count

Read Only Property of VcCalendarProfileCollection

This property lets you retrieve the number of calendar profiles in the calendar profile collection.

	Data Type	Explanation
Property value	Long	Number of CalendarProfile objects

Methods

Add

Method of VcCalendarProfileCollection

By this method you can create a calendar profile as a member of the CalendarProfileCollection. If the name has not been used before, the new filter object will be returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
Parameter: ⇨ profileName	String **Possible Values:**	Calendar profile name Name of the color map
Return value	VcCalendarProfile	New calendar profile object

AddBySpecification

Method of VcCalendarProfileCollection

This method lets you create a calendar profile by using a calendar profile specification. This way of creating allows calendar profile objects to become persistent. The specification of a calendar profile can be saved and re-loaded (see VcCalendarProfile property **Specification**). In a subsequent the calendar profile can be created again from the specification and is identified by its name.

	Data Type	Explanation
Parameter: ⇨ Specification	String **Possible Values:**	Calendar profile specification Name of the color map
Return value	VcCalendarProfile	New calendarprofile object

CalendarProfileByIndex

Method of VcCalendarProfileCollection

This method lets you access a calendar profile by its index. If no calendar profile of the specified index does exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter: ⇨ index	Integer **Possible Values:**	Index of the calendar profile Data field index
Return value	VcCalendarProfile	Calendar profile object returned

CalendarProfileByName

Method of VcCalendarProfileCollection

By this method you can retrieve a calendar profile by its name. If no calendar profile of the specified name does exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter:		
⇨ profileName	String **Possible Values:**	Name of the calendar profile object Name of the color map
Return value	VcCalendarProfile	Calendar profile object returned

Copy

Method of VcCalendarProfileCollection

By this method you can copy a calendar profile. If the calendar profile that is to be copied exists, and if the name for the new calendar profile does not yet exist, the new calendar profile object is returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
Parameter:		
⇨ profileName	String **Possible Values:**	Name of the calendar profile to be copied Name of the color map
⇨ newprofileName	String **Possible Values:**	Name of the new calendar profile Name of the color map
Return value	VcCalendarProfile	Calendar profile object

FirstCalendarProfile

Method of VcCalendarProfileCollection

This method can be used to access the initial value, i.e. the first calendar profile of a calendar profile collection, and then to continue in a forward iteration loop by the method **NextCalendarProfile** for the calendar profiles following. If there is no calendar profile in the FilterCollection object, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcCalendarProfile	First calendar profile object

NextCalendarProfile

Method of VcCalendarProfileCollection

This method can be used in a forward iteration loop to retrieve subsequent calendar profiles from a calendar profile collection after initializing the loop by the method **FirstCalendarProfile**. If there is no calendar profile left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcCalendarProfile	Subsequent calendar profile object

Remove

Method of VcCalendarProfileCollection

This method lets you delete a calendar profile. If the calendar profile is used in another object, it cannot be deleted. Then False will be returned, otherwise True.

	Data Type	Explanation
Parameter:		
⇨ profileName	String	Calendar profile name
	Possible Values:	Name of the color map
Return value	Boolean	Calendar profile deleted (True)/not deleted (False)

SelectCalendarProfiles

Method of VcCalendarProfileCollection

This method lets you specify the calendar profiles that the calendar profile collection is to contain.

	Data Type	Explanation
Parameter:		
⇨ selectionType	CalendarProfileTypeEnum **Possible Values:** vcDayProfile 4 vcShiftProfile 5 vcWeekProfile 3 vcYearProfile 2	Type of calendar profile to be selected
Return value	Long	Number of calendar profiles selected

Example Code

```
Dim calendarProfileCltn As VcCalendarProfileCollection

Set calendarProfileCltn = VcGantt1.CalendarProfileCollection
calendarProfileCltn.SelectCalendarProfile (vcSelected)
```

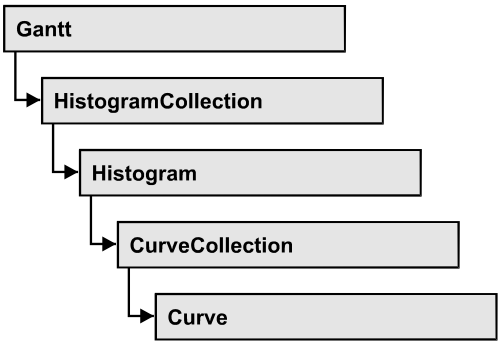
Update

Method of VcCalendarProfileCollection

This method lets you update a calendar profile collection after having modified it.

	Data Type	Explanation
Return value	Boolean	update successful (True)/ not successful (False)

7.17 VcCurve



A VcCurve object represents a stacked curve in the histogram which allows you, for example, to display the capacity and availability of resources. The values for the histogram curves can be entered directly or derived from layers. To enter the values directly, select the option **Data specified manually** in the **Select Curve Data Source** dialog box and generate the curve in your application using the **SetValues** method. To derive the curve from activity values, select the option **Data generated by layer** in the **Select Curve Data Source** dialog box and select a layer.

Properties

- Addend
- CurveSource
- CurveType
- Fill2Color
- Fill2Pattern
- Fill2ReferenceName
- FillColor
- FillPattern
- FillReferenceName
- FilterName
- Histogram
- LayerName
- LineColor
- LineThickness
- LineType
- MarkCurve
- Name
- OverloadResultsCalendarName
- Pattern2Color
- PatternColor

- PointsEquidistant
- Specification
- StackReferenceName
- TimeUnit
- UnitsPerStep
- UpdateBehaviorName
- ValencyDataFieldIndex
- Visible

Methods

- Clear
- DeletePoint
- DeletePointAsVariant
- GetFirstOverload
- GetFirstOverloadAsVariant
- GetFirstOverloadEx
- GetNextOverload
- GetNextOverloadAsVariant
- GetNextOverloadEx
- GetValues
- GetValuesAsVariant
- GetValuesEx
- SetValues

Properties

Addend

Property of VcCurve

This property lets you add a value to all y values of a histogram curve which was generated by API commands.

	Data Type	Explanation
Property value	Long	Value that is added to the y values of the histogram curve

Example Code

```
Dim histogram As VcHistogram
Dim fixCurve As VcCurve
```

```
Set histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
Set fixCurve = histogram.CurveCollection.CurveByName("Availability")

fixCurve.Addend ("1")
```

CurveSource

Read Only Property of VcCurve

This property lets you enquire the source that the data of a histogram curve are taken from. You can set this property in the **Select Curve Data Source** dialog box. If **vcSetCurve** is returned (**Data specified manually** in the **Select Curve Data Source** dialog box), you can set the data in your application by the **SetValues** method. If **vcCalculateFromLayer** is returned (**Data generated by layer**), the data will be calculated from the layers.

	Data Type	Explanation
Property value	CurveSourceEnum	Calculation from field data, from dc data, from layer data, curve set
	Possible Values: vcCalculateFromLayer 1 vcSetCurve 3	 Curve values calculated from layer Curve values are set manually

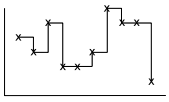
Example Code

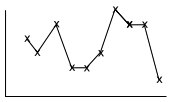
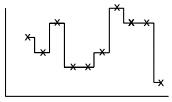
```
Dim histogram As VcHistogram
Dim curve As VcCurve
Dim curveSource As Long

Set histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
Set curve = histogram.CurveCollection.CurveByName("Curve1")

curveSource = curve.CurveSource
```

CurveType

Read Only Property of VcCurve

This property lets you enquire the type of histogram curve.

	Data Type	Explanation
Property value	CurveTypeEnum	Capacity curve **Default value:** vcCapacityCurve
	Possible Values: vcCapacityCurve 215	 Capacity curve 

vcLineCurve 214	Line curve 
vcStepCurve 216	Step curve 

Example Code

```
Dim histogram As VcHistogram
Dim curve As VcCurve
Dim curveType As Long

Set histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
Set curve = histogram.CurveCollection.CurveByName("Curve1")

curveType = curve.CurveType
```

Fill2Color

Property of VcCurve

This property lets you set or retrieve the background color of pattern in the area above the second reference curve. The filling of the second reference curve will be displayed only if the values of the current curve are greater than those of the second reference curve.

You can also set this property in the **Edit Histogram** dialog.

	Data Type	Explanation
Property value	Color	RGB color values ({0...255},{0...255},{0...255}) **Default value:** As defined in the dialog

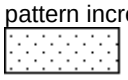
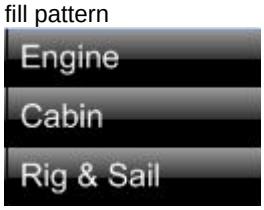
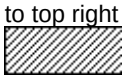
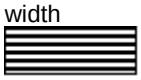
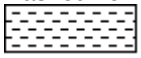
Example Code

```
Dim histogram As VcHistogram
Dim curve As VcCurve

Set histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
Set curve = histogram.CurveCollection.CurveByName("Curve1")

curve.Fill2Color = RGB(150, 100, 170)
```

Fill2Pattern

Property of VcCurve

This property lets you set or retrieve the fill pattern of the area between a histogram curve and the second reference curve. You can also set this property in the **Edit Histogram** dialog.

	Data Type	Explanation
Property value	FillPatternEnum	Type of fill pattern **Default value:** As defined in the dialog
	Possible Values: vc05PercentPattern... vc90PercentPattern 01 - 11	Dots in foreground color on background color, the density of the foreground pattern increasing with the percentage 
	vcAeroGlassPattern 40	Vertical color gradient in the color of the fill pattern 
	vcBDiagonalPattern 5	Diagonal lines slanting from bottom left to top right 
	vcCrossPattern 6	Cross-hatch pattern 
	vcDarkDownwardDiagonalPattern 2014	Diagonal lines slanting from top left to bottom right; spaced 50% closer than vcFDiagonalPattern and of twice the line width 
	vcDarkHorizontalPattern 2023	Horizontal lines spaced 50% closer than vcHorizontalPattern and of twice the line width 
	vcDarkUpwardDiagonalPattern 2015	Diagonal lines slanting from bottom left to top right, spaced 50% closer than vcBDiagonalPattern and of twice the line width 
	vcDarkVerticalPattern 2022	Vertical lines spaced 50% closer than vcVerticalPattern and of of twice the line width 
	vcDashedDownwardDiagonalPattern 2024	Dashed diagonal lines from top left to bottom right 
	vcDashedHorizontalPattern 2026	Dashed horizontal lines 

vcDashedUpwardDiagonalPattern 2025	Dashed diagonal lines from bottom left to top right
vcDashedVerticalPattern 2027	Dashed vertical lines
vcDiagCrossPattern 7	Diagonal cross-hatch pattern, small
vcDiagonalBrickPattern 2032	Diagonal brick pattern
vcDivotPattern 2036	Divot pattern
vcDottedDiamondPattern 2038	Diagonal cross-hatch pattern of dotted lines
vcDottedGridPattern 2037	Cross-hatch pattern of dotted lines
vcFDiagonalPattern 4	Diagonal lines slanting from top left to bottom right
vcHorizontalBrickPattern 2033	Horizontal brick pattern
vcHorizontalGradientPattern 52	Horizontal color gradient
vcHorizontalPattern 3	Horizontal lines
vcLargeCheckerboardPattern 2044	Checkerboard pattern showing squares of twice the size of vcSmallChecker-BoardPattern
vcLargeConfettiPattern 2029	Confetti pattern, large
vcLightDownwardDiagonalPattern 2012	Diagonal lines slanting to from top left to bottom right; spaced 50% closer than vcBDiagonalPattern
vcLightHorizontalPattern 2019	Horizontal lines spaced 50% closer than vcHorizontalPattern
vcLightUpwardDiagonalPattern 2013	Diagonal lines slanting from bottom left to top right, spaced 50% closer than vcBDiagonalPattern
vcLightVerticalPattern 2018	Vertical lines spaced 50% closer than vcVerticalPattern
vcNarrowHorizontalPattern 2021	Horizontal lines spaced 75 % closer than vcHorizontalPattern
vcNarrowVerticalPattern 2020	Vertical lines spaced 75% closer than vcVerticalPattern
vcNoPattern 1276	No fill pattern

vcOutlinedDiamondPattern 2045	Diagonal cross-hatch pattern, large
vcPlaidPattern 2035	Plaid pattern
vcShinglePattern 2039	Diagonal shingle pattern
vcSmallCheckerBoardPattern 2043	Checkerboard pattern
vcSmallConfettiPattern 2028	Confetti pattern
vcSmallGridPattern 2042	Cross-hatch pattern spaced 50% closer than vcCrossPattern
vcSolidDiamondPattern 2046	Checkerboard pattern showing diagonal squares
vcSpherePattern 2041	Checkerboard of spheres
vcTrellisPattern 2040	Trellis pattern
vcVerticalBottomLightedConvexPattern 43	Vertical color gradient from dark to bright
vcVerticalConcavePattern 40	Vertical color gradient from dark to bright to dark
vcVerticalConvexPattern 41	Vertical color gradient from bright to dark to bright
vcVerticalGradientPattern 62	Vertical color gradient
vcVerticalPattern 2	Vertical lines
vcVerticalTopLightedConvexPattern 42	Vertical color gradient from bright to dark
vcWavePattern 2031	Horizontal wave pattern
vcWeavePattern 2034	Interwoven stripe pattern
vcWideDownwardDiagonalPattern 2016	Diagonal lines slanting from top left to bottom right, showing the same spacing but three times the line width of vcF-DiagonalPattern
vcWideUpwardDiagonalPattern 2017	Diagonal lines slanting from bottom left to top right right, showing the same spacing but three times the line width of vcBDiagonalPattern

vcZigZagPattern 2030	Horizontal zig-zag lines 

Example Code

```
Dim histogram As VcHistogram
Dim curve As VcCurve

Set histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
Set curve = histogram.CurveCollection.CurveByName("Curve1")

curve.Fill2Pattern = vcDiagCrossPattern
```

Fill2ReferenceName

Property of VcCurve

This property lets you set or retrieve the name of the second reference curve of a curve. The area between the curve and its second reference curve specifies can be filled by a pattern. This property is set in the **Edit Histogram** dialog.

	Data Type	Explanation
Property value	String	Name of the 2nd reference curve
	Possible Values:	Name of the color map

Example Code

```
Dim histogram As VcHistogram
Dim curve As VcCurve
Dim fillRef As Object

Set histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
Set curve = histogram.CurveCollection.CurveByName("Curve1")
Set referenceCurve = histogram.CurveCollection. _
            CurveByName(curve.Fill2ReferenceName)
```

FillColor

Property of VcCurve

This property lets you set or retrieve the color of the area between a histogram curve and the fill reference object set. You can also set this property in the **Edit Histogram** dialog.

	Data Type	Explanation
Property value	Color	RGB color values ({0...255},{0...255},{0...255}) **Default value:** As defined in the dialog

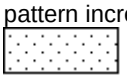
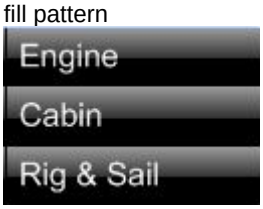
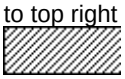
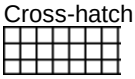
Example Code

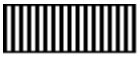
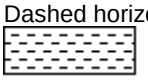
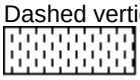
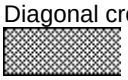
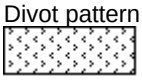
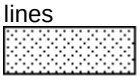
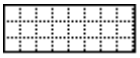
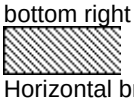
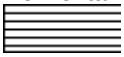
```
Dim histogram As VcHistogram
Dim curve As VcCurve

Set histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
Set curve = histogram.CurveCollection.CurveByName("Curve1")

curve.FillColor = RGB(150, 100, 170)
```

FillPattern

Property of VcCurve

This property lets you set or retrieve the fill pattern of the area between a histogram curve and the fill reference object set. You can also set this property in the **Edit Histogram** dialog.

	Data Type	Explanation
Property value	FillPatternEnum	Type of fill pattern **Default value:** As defined in the dialog
	Possible Values: vc05PercentPattern... vc90PercentPattern 01 - 11	Dots in foreground color on background color, the density of the foreground pattern increasing with the percentage 
	vcAeroGlassPattern 40	Vertical color gradient in the color of the fill pattern 
	vcBDiagonalPattern 5	Diagonal lines slanting from bottom left to top right 
	vcCrossPattern 6	Cross-hatch pattern 
	vcDarkDownwardDiagonalPattern 2014	Diagonal lines slanting from top left to bottom right; spaced 50% closer than vcFDiagonalPattern and of twice the line width 

vcDarkHorizontalPattern 2023	Horizontal lines spaced 50% closer than vcHorizontalPattern and of twice the line width 
vcDarkUpwardDiagonalPattern 2015	Diagonal lines slanting from bottom left to top right, spaced 50% closer than vcBDiagonalPattern and of twice the line width 
vcDarkVerticalPattern 2022	Vertical lines spaced 50% closer than vcVerticalPattern and of of twice the line width 
vcDashedDownwardDiagonalPattern 2024	Dashed diagonal lines from top left to bottom right 
vcDashedHorizontalPattern 2026	Dashed horizontal lines 
vcDashedUpwardDiagonalPattern 2025	Dashed diagonal lines from bottom left to top right 
vcDashedVerticalPattern 2027	Dashed vertical lines 
vcDiagCrossPattern 7	Diagonal cross-hatch pattern, small 
vcDiagonalBrickPattern 2032	Diagonal brick pattern 
vcDivotPattern 2036	Divot pattern 
vcDottedDiamondPattern 2038	Diagonal cross-hatch pattern of dotted lines 
vcDottedGridPattern 2037	Cross-hatch pattern of dotted lines 
vcFDiagonalPattern 4	Diagonal lines slanting from top left to bottom right 
vcHorizontalBrickPattern 2033	Horizontal brick pattern 
vcHorizontalGradientPattern 52	Horizontal color gradient 
vcHorizontalPattern 3	Horizontal lines 
vcLargeCheckerboardPattern 2044	Checkerboard pattern showing squares of twice the size of vcSmallChecker-BoardPattern 
vcLargeConfettiPattern 2029	Confetti pattern, large 

vcLightDownwardDiagonalPattern 2012	Diagonal lines slanting to from top left to bottom right; spaced 50% closer than vcBDiagonalPattern
vcLightHorizontalPattern 2019	Horizontal lines spaced 50% closer than vcHorizontalPattern
vcLightUpwardDiagonalPattern 2013	Diagonal lines slanting from bottom left to top right, spaced 50% closer than vcBDiagonalPattern
vcLightVerticalPattern 2018	Vertical lines spaced 50% closer than vcVerticalPattern
vcNarrowHorizontalPattern 2021	Horizontal lines spaced 75 % closer than vcHorizontalPattern
vcNarrowVerticalPattern 2020	Vertical lines spaced 75% closer than vcVerticalPattern
vcNoPattern 1276	No fill pattern
vcOutlinedDiamondPattern 2045	Diagonal cross-hatch pattern, large
vcPlaidPattern 2035	Plaid pattern
vcShinglePattern 2039	Diagonal shingle pattern
vcSmallCheckerBoardPattern 2043	Checkerboard pattern
vcSmallConfettiPattern 2028	Confetti pattern
vcSmallGridPattern 2042	Cross-hatch pattern spaced 50% closer than vcCrossPattern
vcSolidDiamondPattern 2046	Checkerboard pattern showing diagonal squares
vcSpherePattern 2041	Checkerboard of spheres
vcTrellisPattern 2040	Trellis pattern
vcVerticalBottomLightedConvexPattern 43	Vertical color gradient from dark to bright
vcVerticalConcavePattern 40	Vertical color gradient from dark to bright to dark
vcVerticalConvexPattern 41	Vertical color gradient from bright to dark to bright
vcVerticalGradientPattern 62	Vertical color gradient

vcVerticalPattern 2	Vertical lines
vcVerticalTopLightedConvexPattern 42	Vertical color gradient from bright to dark
vcWavePattern 2031	Horizontal wave pattern
vcWeavePattern 2034	Interwoven stripe pattern
vcWideDownwardDiagonalPattern 2016	Diagonal lines slanting from top left to bottom right, showing the same spacing but three times the line width of vcF-DiagonalPattern
vcWideUpwardDiagonalPattern 2017	Diagonal lines slanting from bottom left to top right right, showing the same spacing but three times the line width of vcBDiagonalPattern
vcZigZagPattern 2030	Horizontal zig-zag lines

Example Code

```
Dim histogram As VcHistogram
Dim curve As VcCurve

Set histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
Set curve = histogram.CurveCollection.CurveByName("Curve1")

curve.FillPattern = vcDiagCrossPattern
```

FillReferenceName

Property of VcCurve

This property lets you enquire the name of the fill reference (for example a different curve or the x axis) of a histogram curve. The fill reference specifies an object that limits an area to be filled by colors and/or patterns. This property is set in the **Edit Histogram** dialog.

Note: The name of the x axis as fill reference has to be "VC_AXIS".

	Data Type	Explanation
Property value	String	Name of the reference curve
	Possible Values:	Name of the color map

Example Code

```
Private Sub VcGantt2_OnDiagramRClick(ByVal x As Long, ByVal y As Long,
returnStatus As Variant)
  Dim histogram As VcHistogram
  Dim curve As VcCurve

  Set histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
  Set curve = histogram.CurveCollection.CurveByName("Curve_1")
  curve.FillReferenceName = "VC_AXIS"
End Sub
```

FilterName

Property of VcCurve

This property lets you assign a filter to the curve or enquire the existing one.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the filter Name of the color map

Example Code

```
Dim histogram As VcHistogram
Dim curve As VcCurve

Set histogram = VcGantt1.histogramCollection.HistogramByName("HISTOGRAM_1")
Set curve = histogram.CurveCollection.CurveByName("Curve1")
curve.FilterName = "Critical"
```

Histogram

Read Only Property of VcCurve

This property lets you retrieve the histogram which the curve belongs to.

	Data Type	Explanation
Property value	VcHistogram	Histogram object

Example Code

```
Dim histogram As VcHistogram
Dim curve As VcCurve

Set curve = VcGantt1.HistogramCollection.FirstHistogram.curveCollection. _
                           CurveByName("Curve1")
Set histogram = curve.Histogram
```

LayerName

Property of VcCurve

This property lets you assign a layer to the curve or enquire the existing one.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the layer Name of the color map

Example Code

```
Dim histogram As VcHistogram
Dim curve As VcCurve

Set histogram = VcGantt1.histogramCollection.HistogramByName("HISTOGRAM_1")
Set curve = histogram.CurveCollection.CurveByName("Curve1")
curve.LayerName = "Start-End"
```

LineColor

Property of VcCurve

This property lets you set or retrieve the line color of a histogram curve. This property you can also set in the **Edit Histogram** dialog.

	Data Type	Explanation
Property value	Color	RGB color values ({0...255},{0...255},{0...255}) **Default value:** As defined in the dialog

Example Code

```
Dim histogram As VcHistogram
Dim curve As VcCurve

Set histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
Set curve = histogram.CurveCollection.CurveByName("Curve1")

curve.LineColor = RGB(200, 0, 180)
```

LineThickness

Property of VcCurve

This property lets you set or retrieve the line thickness of a histogram curve.

If you set this property to values between 1 and 4, an absolute line thickness is defined in pixels. Irrespective of the zoom factor a line will always show

the same line thickness in pixels. When printing though, the line thickness is adapted for the sake of legibility and becomes dependent of the zoom factor:

Value	Points	mm
1	1/2 point	0.09 mm
2	1 point	0.18 mm
3	3/2 points	0.26 mm
4	2 points	0.35 mm

A point equals 1/72 inch and represents the unit of the font size.

If you set this property to values between 5 and 1,000, the line thickness is defined in 1/100 mm, so the lines will be displayed in a true thickness in pixels that depends on the zoom factor.

This property also can be set in the **Edit Histogram** dialog.

	Data Type	Explanation
Property value	Integer	Line thickness LineType {1...4}: line thickness in pixels LineType {5...1000}: line thickness in 1/100 mm **Default value:** As defined in the dialog
	Possible Values:	Data field index

Example Code

```
Dim histogram As VcHistogram
Dim curve As VcCurve

Set histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
Set curve = histogram.CurveCollection.CurveByName("Curve1")

curve.LineType = vcSolid
curve.LineThickness = 3

' or:
curve.LineType = vcLineType5
curve.LineThickness = 20
```

LineType

Property of VcCurve

This property lets you set or retrieve the line type of a histogram curve. If for stacked curves you do not wish the lines to be displayed, you can select **vcNone**. This property also can be set in the **Edit Histogram** dialog.

	Data Type	Explanation
Property value	LineTypeEnum	Line type **Default value:** vcSolid
	Possible Values:	
	vcDashed 4	Line dashed
	vcDashedDotted 5	Line dashed-dotted
	vcDotted 3	Line dotted
	vcLineType0 100	Line Type 0
	vcLineType1 101	Line Type 1
	vcLineType10 110	Line Type 10
	vcLineType11 111	Line Type 11
	vcLineType12 112	Line Type 12
	vcLineType13 113	Line Type 13
	vcLineType14 114	Line Type 14
	vcLineType15 115	Line Type 15
	vcLineType16 116	Line Type 16
	vcLineType17 117	Line Type 17
	vcLineType18 118	Line Type 18
	vcLineType2 102	Line Type 2
	vcLineType3 103	Line Type 3
	vcLineType4 104	Line Type 4
	vcLineType5 105	Line Type 5
	vcLineType6 106	Line Type 6
	vcLineType7 107	Line Type 7
	vcLineType8 108	Line Type 8
	vcLineType9 109	Line Type 9
	vcNone 1	No line type
	vcNotSet -1	No line type assigned
	vcSolid 2	Line solid

Example Code

```
Dim histogram As VcHistogram
Dim curve As VcCurve

Set histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
Set curve = histogram.CurveCollection.CurveByName("Curve1")

curve.LineType = vcSolid
```

MarkCurve

Property of VcCurve

This property lets you set or retrieve the marking state of a histogram curve set by the API.

	Data Type	Explanation
Property value	Boolean	Curve marked/not marked
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
Dim histogram As VcHistogram
Dim fixCurve As VcCurve

Set histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
Set fixCurve = histogram.CurveCollection.CurveByName("Availability")

fixCurve.MarkCurve = True
```

Name

Read Only Property of VcCurve

This property lets you retrieve the name of a histogram curve.

	Data Type	Explanation
Property value	String	Curve name
	Possible Values:	Name of the color map

Example Code

```
Dim histogram As VcHistogram
Dim curve As VcCurve
Dim curveName As String

Set histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
Set curve = histogram.CurveCollection.CurveByName("Curve1")

curveName = curve.Name
```

OverloadResultsCalendarName

Property of VcCurve

This property lets you set or retrieve a calendar to store the intervalls that have been calculated by the overload dates. You could use this calendar, for instance, to display a calendar grid in a group

	Data Type	Explanation
Property value	String **Possible Values:**	Name of overload results calendar object Name of the color map

Pattern2Color

Property of VcCurve

This property lets you set or retrieve the foreground color of the pattern of the area above the second reference curve. The filling of the second reference curve will be displayed only if the values of the current curve are greater than those of the second reference curve.

You can also set this property in the **Edit Histogram** dialog.

	Data Type	Explanation
Property value	Color	RGB color values ({0...255},{0...255},{0...255}) **Default value:** As defined in the dialog

Example Code

```
Dim histogram As VcHistogram
Dim curve As VcCurve

Set histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
Set curve = histogram.CurveCollection.CurveByName("Curve1")

curve.Pattern2Color = RGB(150, 150, 110)
```

PatternColor

Property of VcCurve

This property lets you set or retrieve the color of the pattern of the area between a histogram curve and the fill reference object set. You can also set this property in the **Edit Histogram** dialog.

	Data Type	Explanation
Property value	Color	RGB color values ({0...255},{0...255},{0...255}) **Default value:** As defined in the dialog

Example Code

```
Dim histogram As VcHistogram
Dim curve As VcCurve

Set histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
Set curve = histogram.CurveCollection.CurveByName("Curve1")

curve.PatternColor = RGB(150, 150, 110)
```

PointsEquidistant

Property of VcCurve

This property lets you set or retrieve whether the curve points are to be equidistant. In case of **False**, the curve points will be created only in those points where the y values are changing. This property also can be set in the **Select Curve Data Source** dialog.

	Data Type	Explanation
Property value	Boolean	Curve points equidistant (True) / not equidistant (False)
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
Dim histogram As VcHistogram
Dim curve As VcCurve

Set histogram = VcGantt1.HistogramCollection.FirstHistogram
Set curve = histogram.CurveCollection.CurveByName("Curve1")
curve.PointsEquidistant = False
```

Specification

Read Only Property of VcCurve

This property lets you retrieve the specification of a curve. A specification is a string that contains legible ASCII characters from 32 to 127 only, so it can be stored without problems to text files or data bases. This allows for persistency. A specification can be used to create a curve by the method **VcCurveCollection.AddBySpecification**.

	Data Type	Explanation
Property value	String **Possible Values:**	Specification of the curve Name of the color map

StackReferenceName

Property of VcCurve

This property lets you set or retrieve the name of the stack reference curve of a histogram curve. It specifies on which other curve each curve is to be stacked, and it has to be specified in order to be able to stack the curves. You can also set this property in the **Edit Histogram** dialog.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the stack curve Name of the color map

Example Code

```
Dim histogram As VcHistogram
Dim curve As VcCurve
Dim referenceCurve As Object

Set histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
Set curve = histogram.CurveCollection.CurveByName("Curve1")

Set referenceCurve =
histogram.CurveCollection.CurveByName(curve.StackReferenceName)
```

TimeUnit

Read Only Property of VcCurve

This property lets you retrieve the time unit of a histogram curve. The property can be applied to curves that were generated by the API only. If applied to a curve generated from layer values, the property will return the result of -1. You can set the time unit on the property page **General**.

	Data Type	Explanation
Property value	TimeUnitEnum **Possible Values:** vcDay 5 vcHour 6	time unit **Default value:** As defined in the dialog Time unit **day** Time unit **hour**

vcMinute 7	Time unit **minute**
vcSecond 8	Time unit **second**

Example Code

```
Dim histogram As VcHistogram
Dim curve As VcCurve
Dim timeUnit As Long

Set histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
Set curve = histogram.CurveCollection.CurveByName("Curve1")

timeUnit = curve.timeUnit
```

UnitsPerStep

Read Only Property of VcCurve

This property lets you retrieve the number of units per step of a histogram curve. The number can be set on the property page **General**.

	Data Type	Explanation
Property value	Integer	Number of units **Default value:** As defined in the dialog
	Possible Values:	Data field index

Example Code

```
Dim histogram As VcHistogram
Dim curve As VcCurve
Dim unitsPerStep As Integer

Set histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
Set curve = histogram.CurveCollection.CurveByName("Curve1")

unitsPerStep = curve.UnitsPerStep
```

UpdateBehaviorName

Property of VcCurve

This property lets you set or retrieve the name of the UpdateBehavior.

	Data Type	Explanation
Property value	String	Name of the UpdateBehavior
	Possible Values:	Name of the color map

ValencyDataFieldIndex

Property of VcCurve

This property lets you set or retrieve the valency field of a curve generated by layer. The valency field is the data field from which for each activity the valency for the capacity sum is to be taken.

	Data Type	Explanation
Property value	Integer	Index of the valency field
	Possible Values:	Data field index

Visible

Property of VcCurve

This property lets you set or retrieve whether a curve is visible. You can also set this property in the **Administer Histograms** dialog.

	Data Type	Explanation
Property value	Boolean	curve visible/invisible **Default value:** True
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
Dim histogram As VcHistogram
Dim curve As VcCurve

Set histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
Set curve = histogram.CurveCollection.CurveByName("Curve1")

curve.Visible = True
```

Methods

Clear

Method of VcCurve

This method lets you set all y values of a curve to zero. The method can be applied only to those curves the values of which were generated by the API.

	Data Type	Explanation
Return value	Void	

Example Code

```
Dim histogram As VcHistogram
Dim fixCurve As VcCurve

Set histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
Set fixCurve = histogram.CurveCollection.CurveByName("Availability")

fixCurve.Clear
```

DeletePoint

Method of VcCurve

This method lets you remove the curve point nearest to the x-coordinate.

Note: If you use VBScript, you can only use the analogue method **DeletePointAsVariant** because of the parameters by Reference.

	Data Type	Explanation
Parameter:		
⇨ x	Long	X value of the curve point to be deleted
⇨ y	Long	Y value of the curve point to be deleted
⇦ pointDate	Date	Date of the curve point which was deleted
Return value	Boolean	Curve point was (True) / was not (False) deleted successfully

Example Code

```
Private Sub VcGantt1_OnCurveRClick(ByVal curve As VcGanttLib.VcCurve, _
                                   ByVal x As Long, ByVal y As Long, _
                                   returnStatus As Variant)
    Dim pointDate As Date
    Dim deleted As Boolean

    returnStatus = vcRetStatNoPopup
    deleted = curve.DeletePoint(x, y, pointDate)
    If deleted Then Call MsgBox(-pointDate)
    End If
End Sub
```

DeletePointAsVariant

Method of VcCurve

This method is identical with the method **DeletePoint** except for the parameters. It was necessary to implement this event because some languages

(e.g. VBScript) can use parameters by Reference (indicated by ⇦) only if the type of these parameters is VARIANT.

GetFirstOverload

Method of VcCurve

An **overload** is the area between the current curve and a reference curve with the former showing higher values than the latter. The reference curve is the curve defined as the second fill reference (**2nd Ref**) in the **Edit Histogram** dialog.

This method can be used to access the initial value, i.e. the first overload, and then to continue in a forward iteration loop by the method **GetNextOverload** for the overloads following.

Please note: If you use VBScript, due to the by-reference parameters, you can only use the analogous method **GetFirstOverloadAsVariant**.

Please note: For floating point numbers in the parameters **fromValue** and **toValue** please use the method **GetFirstOverloadEx**.

	Data Type	Explanation
Parameter:		
⇦ fromDate	Date/Time	Start date of the overload area
⇦ fromValue	Long	Y-value of the start date of the overload area
⇦ toDate	Date/Time	Final date of the overload area
⇦ toValue	Long	Y-value of the final date of the overload area
Return value	Boolean	Overload was (True) / was not (False) retrieved successfully

Example Code

```
Dim histogram As VcHistogram

Dim curve1 As VcCurve, fixCurve As VcCurve

Dim yValues As String
Dim bOK As Boolean

Dim fromDate As Date, toDate As Date
Dim fromValue As Long, toValue As Long

Set histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
Set curve1 = histogram.CurveCollection.CurveByName("LayerCurve")
Set fixCurve = histogram.CurveCollection.CurveByName("FixCurve")
```

```
yValues = "6;6;6;6;6;0;0;6;6;6;6;6;0;0;6;6;6;6;6;0;0;6;6;6;6;6;0;0;6;6;6;6;6"
fixCurve.SetValues "31.08.14", yValues

bOK = curve1.GetFirstOverload(fromDate, fromValue, toDate, toValue)
```

GetFirstOverloadAsVariant

Method of VcCurve

This method is identical with the method **GetFirstOverload** except for the parameters. It was necessary to implement this event because some languages (e.g. VBScript) can use parameters by Reference (indicated by ⇦) only if the type of these parameters is VARIANT.

GetFirstOverloadEx

Method of VcCurve

An **overload** is the area between the current curve and a reference curve with the former showing higher values than the latter. The reference curve is the curve defined as the second fill reference (**2nd Ref**) in the **Edit Histogram** dialog.

This method can be used to access the initial value, i.e. the first overload, and then to continue in a forward iteration loop by the method **GetNext-OverloadEx** for the overloads following.

Please note: If you use VBScript, due to the by-reference parameters, you can only use the analogous method **GetFirstOverloadAsVariant**.

Please note: Compared to the method **GetFirstOverload** this method allows for floating point numbers in the parameters **fromValue** and **toValue**.

	Data Type	Explanation
Parameter:		
⇦ fromDate	Date/Time	Start date of the overload area
⇦ fromValue	Double	Y-value of the start date of the overload area
⇦ toDate	Date/Time	Final date of the overload area
⇦ toValue	Double	Y-value of the final date of the overload area
Return value	Boolean	Overload was (True) / was not (False) retrieved successfully

Example Code

```
Dim histogram As VcHistogram

Dim curve1 As VcCurve, fixCurve As VcCurve

Dim yValues As String
Dim bOK As Boolean

Dim fromDate As Date, toDate As Date
Dim fromValue As Long, toValue As Long

Set histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
Set curve1 = histogram.CurveCollection.CurveByName("LayerCurve")
Set fixCurve = histogram.CurveCollection.CurveByName("FixCurve")

yValues = "6;6;6;6;6;0;0;6;6;6;6;6;0;0;6;6;6;6;6;0;0;6;6;6;6;6;0;0;6;6;6;6;6"
fixCurve.SetValues "31.08.14", yValues

bOK = curve1.GetFirstOverload(fromDate, fromValue, toDate, toValue)
```

GetNextOverload

Method of VcCurve

An **overload** is the area between the current curve and a reference curve with the former showing higher values than the latter. The reference curve is the curve defined as the second fill reference (**2nd Ref**) in the **Edit Histogram**.

This method can be used in a forward iteration loop to retrieve subsequent calendars from a calendar collection after initializing the loop by the method **GetFirstOverload**.

Please note: If you use VBScript, you can only use the analogue method **GetNextOverloadAsVariant** because of the parameters by Reference.

Please note: For floating point numbers in the parameters **fromValue** and **toValue** please use the method **GetNextOverloadEx**.

	Data Type	Explanation
Parameter:		
⇦ fromDate	Date/Time	Start date of the overload area
⇦ fromValue	Long	Y-value of the start date of the overload area
⇦ toDate	Date/Time	Final date of the overload area
⇦ toValue	Long	Y-value of the final date of the overload area
Return value	Boolean	Overload was (True) / was not (False) retrieved successfully.

Example Code

. . .

```
bOK = curve1.GetFirstOverload(fromDate, fromValue, toDate, toValue)

MsgBox fromDate & " (" & fromValue & ") - " & toDate & " (" & toValue & ")"

While bOK
    bOK = curve1.GetNextOverload(fromDate, fromValue, toDate, toValue)
    If bOK Then MsgBox fromDate & " (" & fromValue & ") - " & toDate & " (" _
                                & toValue & ")"
Wend
```

GetNextOverloadAsVariant

Method of VcCurve

This method is identical with the method **GetNextOverload** except for the parameters. It was necessary to implement this event because some languages (e.g. VBScript) can use parameters by Reference (indicated by ⇦) only if the type of these parameters is VARIANT.

GetNextOverloadEx

Method of VcCurve

An **overload** is the area between the current curve and a reference curve with the former showing higher values than the latter. The reference curve is the curve defined as the second fill reference (**2nd Ref**) in the **Edit Histogram**.

This method can be used in a forward iteration loop to retrieve subsequent overloads from an overload collection after initializing the loop by the method **GetFirstOverloadEx**.

Please note: If you use VBScript, you can only use the analogue method **GetNextOverloadAsVariant** because of the parameters by Reference.

Please note: Compared to the method **GetNextOverload** this method allows for floating point numbers in the parameters **fromValue** and **toValue**.

	Data Type	Explanation
Parameter:		
⇦ fromDate	Date/Time	Start date of the overload area
⇦ fromValue	Double	Y-value of the start date of the overload area
⇦ toDate	Date/Time	Final date of the overload area
⇦ toValue	Double	Y-value of the final date of the overload area
Return value	Boolean	Overload was (True) / was not (False) retrieved successfully.

Example Code

```
...
bOK = curve1.GetFirstOverload(fromDate, fromValue, toDate, toValue)

MsgBox fromDate & " (" & fromValue & ") - " & toDate & " (" & toValue & ")"

While bOK
    bOK = curve1.GetNextOverload(fromDate, fromValue, toDate, toValue)
    If bOK Then MsgBox fromDate & " (" & fromValue & ") - " & toDate & " (" _
                                & toValue & ")"
Wend
```

GetValues

Method of VcCurve

This method lets you retrieve the value of a histogram curve that belongs to a specified date. Since the date specified may not be located in a defined point (pair of coordinates) of the curve, the date and value of the closest defined point before resp. after the specified date will be returned. If a point was hit exactly, its corresponding value will be returned two times i.e. as previous and next value.

Note: If you use VBScript, you can only use the analogous method **GetValuesAsVariant** because of the parameters by Reference.

Note: For floating point values please use the property **GetValuesEx**.

	Data Type	Explanation
Parameter:		
⇨ inputDate	Date/Time	Date that the value of the histogram curve is to be retrieved
⇦ leftDate	Date/Time	Date of the last defined point of the curve before the specified date
⇦ leftValue	Long	Value of the last defined point of the curve before the specified date
⇦ rightDate	Date/Time	Date of the next defined point of the curve after the specified date
⇦ rightValue	Long	Value of the next defined point of the curve after the specified date
Return value	void	

Example Code

```
Dim histogram As VcHistogram
Dim curve1 As VcCurve
Dim inputDate As String
```

```
Dim leftDate As Date, rightDate As Date
Dim leftValue As Long, rightValue As Long

Set histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")

Set curve1 = histogram.CurveCollection.CurveByName("LayerCurve")

inputDate = InputBox("Date: ")

curve1.GetValues CDate(inputDate), leftDate, leftValue, rightDate, rightValue

MsgBox leftDate & " (" & leftValue & ") " & rightDate & " (" & rightValue & ") "
```

GetValuesAsVariant

Method of VcCurve

This method is identical with the method **GetValues** except for the parameters. It was necessary to implement this event because some languages (e.g. VBScript) can use parameters by Reference (indicated by ⇦) only if the type of these parameters is VARIANT.

GetValuesEx

Method of VcCurve

This method lets you retrieve the value of a histogram curve that belongs to a specified date. Compared to the method **GetValues** this method is appropriate for floating point values. Since the date specified may not be located in a defined point (pair of coordinates) of the curve, the date and value of the closest defined point before and after the specified date will be returned. If a point was hit exactly, its corresponding value will be returned twice, i.e. as the previous and the following value.

Note: If you use VBScript, because of the by-reference parameters you can only use the analogous method **GetValuesAsVariant**.

	Data Type	Explanation
Parameter:		
⇨ inputDate	Date/Time	Date that the value of the histogram curve is to be retrieved
⇦ leftDate	Date/Time	Date of the last defined point of the curve before the specified date
⇦ leftValue	Double	Value of the last defined point of the curve before the specified date
⇦ rightDate	Date/Time	Date of the next defined point of the curve after the specified date

⇦ rightValue	Double	Value of the next defined point of the curve after the specified date
Return value	void	

Example Code

```
Dim histogram As VcHistogram
Dim curve1 As VcCurve
Dim inputDate As String

Dim leftDate As Date, rightDate As Date
Dim leftValue As Long, rightValue As Long

Set histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")

Set curve1 = histogram.CurveCollection.CurveByName("LayerCurve")

inputDate = InputBox("Date: ")

curve1.GetValues CDate(inputDate), leftDate, leftValue, rightDate, rightValue

MsgBox leftDate & " (" & leftValue & ") " & rightDate & " (" & rightValue & ") "
```

SetValues

Method of VcCurve

This method lets you set the values of a histogram curve that was generated by the API. A curve built by **SetValues** can be used as a capacitiy curve to display engine resources or can be used as a reference curve.

The usage of the VcCurve.SetValues method depends on the **Curve points equidistant** check box in the **Select Curve Data Source** dialog box:

Curve points equidistant: You can transfer a start value (**startValue**) and a string separated by semicolons that contains the y values. The coordinates of points that form the curve are calculated from the start value and the y values, combined with the **Time Unit** and **Smallest time interval** (property page **General**). Curves generated in this way cannot be edited interactively.

Curve points not equidistant: You have to call the method for each pair of (x,y) values. The **Time Unit** and **Smallest time interval** are not relevant. The curve can be edited interactively.

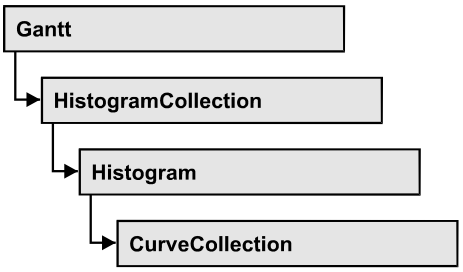
	Data Type	**Explanation**
Parameter:		
⇨ startDate	Date/Time	Start date
⇨ values	String	Y values as a string
	Possible Values:	

		Name of the color map
Return value	Boolean	Values were/were not set successfully

Example Code

```
Dim histogram As VcHistogram
Dim curve As VcCurve
Dim yValues As String

Set histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
Set curve = histogram.CurveCollection.CurveByName("Curve1")

' If the option Curve points equidistant is checked for the curve:
  yValues = "5;1;1;2;2;2;4;5;5;3;2;1;"
  curve.SetValues("2012/02/14 12:05:00", yValues)

' If the option Curve points equidistant is not checked for the curve:
  curve.SetValues("2012/02/14 12:05:00", "5")
  curve.SetValues("2012/02/14 12:07:00", "1")
  curve.SetValues("2012/02/14 12:23:00", "1")
  curve.SetValues("2012/02/14 13:05:00", "2")
```

7.18 VcCurveCollection



An object of the type VcCurveCollection automatically contains all curves of the histogram. You can access all objects in an iterative loop by **For Each curve In CurveCollection** or by the methods **First...** and **Next...**. You can access a single curve using the methods **CurveByName** and **CurveByIndex**. The number of curves in the collection object can be retrieved by the property **Count**. The methods **Add**, **Copy** and **Remove** allow to handle the curves in the corresponding way.

Properties

- _NewEnum
- Count

Methods

- Add
- AddBySpecification
- Copy
- CurveByIndex
- CurveByName
- FirstCurve
- NextCurve
- Remove

Properties

_NewEnum

Read Only Property of VcCurveCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all curve objects.

In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method **GetEnumerator** is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim curve As VcCurve

For Each curve In VcGantt1.CurveCollection
   Debug.Print curve.Name
Next
```

Count

Read Only Property of VcCurveCollection

This property lets you retrieve the number of curves in the CurveCollection.

	Data Type	Explanation
Property value	Long	Number of curves

Example Code

```
Dim histogram As VcHistogram
Dim curveCltn As VcCurveCollection
Dim numberOfCurves As Long

Set histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
Set curveCltn = histogram.CurveCollection

numberOfCurves = curveCltn.Count
```

Methods

Add

Method of VcCurveCollection

By this method you can create a curve as a member of the CurveCollection. If the name has not been used before, the new curve object will be returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
Parameter: ⇨ curveName	String **Possible Values:**	Curve name Name of the color map
Return value	VcCurve	New curve object

Example Code

```
Set newCurve = VcGantt1.CurveCollection.Add("test1")
```

AddBySpecification

Method of VcCurveCollection

This method lets you create a curve by using a curve specification. This way of creating allows curve objects to become persistent. The specification of a curve can be saved and re-loaded (see VcCurve property **Specification**) In a subsequent session the curve can be created again from the specification and is identified by its name.

	Data Type	Explanation
Parameter: ⇨ Specification	String **Possible Values:**	Curve specification Name of the color map
Return value	VcCurve	New curve object

Copy

Method of VcCurveCollection

By this method you can copy a curve. If the curve that is to be copied exists, and if the name for the new curve does not yet exist, the new curve object is returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
Parameter: ⇨ curveName	String **Possible Values:**	Name of the curve to be copied Name of the color map

⇨ newCurveName	String **Possible Values:**	Name of the new curve Name of the color map
Return value	VcCurve	Curve object

Example Code

```
Dim histogram As VcHistogram
Dim curveCltn As VcCurveCollection
Dim curve As VcCurve

Set histogram = VcGantt1.HistogramCollection.FirstHistogram
Set curveCltn = histogram.CurveCollection
Set curve = curveCltn.Copy("CurrentCurve", "NewCurve")
```

CurveByIndex

Method of VcCurveCollection

This method lets you access a curve by its index. If a curve of the specified index does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	**Explanation**
Parameter: ⇨ index	Integer **Possible Values:**	Index of the curve Data field index
Return value	VcCurve	Curve object returned

Example Code

```
Dim histogram As VcHistogram
Dim curveCltn As VcCurveCollection
Dim curve As VcCurve

Set histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
Set curveCltn = histogram.CurveCollection
Set curve = curveCltn.CurveByIndex(2)
```

CurveByName

Method of VcCurveCollection

By this method you can retrieve a curve by its name. If a curve of the specified name does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter:		
⇨ curveName	String	Name of the curve
	Possible Values:	Name of the color map
Return value	VcCurve	Curve

Example Code

```
Dim histogram As VcHistogram
Dim curveCltn As VcCurveCollection
Dim curve As VcCurve

Set histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
Set curveCltn = histogram.CurveCollection

Set curve = curveCltn.CurveByName("Curve1")
```

FirstCurve

Method of VcCurveCollection

This method can be used to access the initial value, i.e. the first curve of a curve collection, and to continue in a forward iteration loop by the method **NextCurve** for the curves following. If there is no curve in the curve collection, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcCurve	First curve

Example Code

```
Dim histogram As VcHistogram
Dim curveCltn As VcCurveCollection
Dim curve As VcCurve

Set histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
Set curveCltn = histogram.CurveCollection

Set curve = curveCltn.FirstCurve
```

NextCurve

Method of VcCurveCollection

This method can be used in a forward iteration loop to retrieve subsequent curves from a curve collection after initializing the loop by the method **FirstCurve**. If there is no curve left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcCurve	Subsequent Curve

Example Code

```
Dim histogram As VcHistogram
Dim curveCollection As VcCurveCollection
Dim curve As VcCurve

Set histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
Set curveCollection = histogram.CurveCollection

Set curve = curveCollection.FirstCurve

While Not curve Is Nothing
    Set curve = curveCollection.NextCurve
Wend
```

Remove

Method of VcCurveCollection

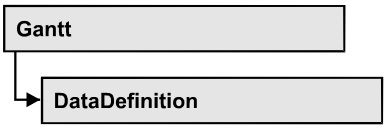
This method lets you delete a curve. If the curve is used in another object, it cannot be deleted. Then False will be returned, otherwise True.

	Data Type	Explanation
Parameter:		
⇨ curveName	String	curve name
	Possible Values:	Name of the color map
Return value	Boolean	Curve deleted (True)/not deleted (False)

Example Code

```
Dim histogram As VcHistogram
Dim curveCltn As VcCurveCollection

Set histogram = VcGantt1.HistogramCollection.FirstHistogram
Set curveCltn = histogram.CurveCollection
curveCltn.Remove ("CurrentCurve")
```

7.19 VcDataDefinition



The data of nodes and links can be defined in the dialog **Administrate Data Tables** which can be reached by selecting **Data tables...** on the **Objects** property page. It grants access to the names and types of the available fields. The data definition of a VcGantt object contains two data definition tables: vcMaindata and vcRelations.

Properties

- DefinitionTable

Properties

DefinitionTable

Read Only Property of VcDataDefinition

This property allows the access to the two tables of the data definition object.

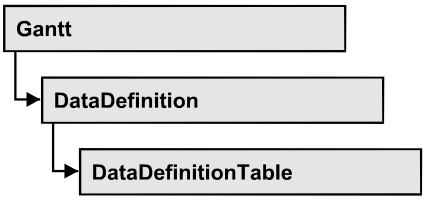
- **vcMaindata:** definitions for nodes
- **vcRelations**: definitions for links

	Data Type	Explanation
Parameter:		
⇨ tableType	DataTableEnum	Type of data definition table
	Possible Values: vcMaindata 0 vcRelations 1	 Table type **vcMaindata** (for nodes) Table type **vcRelations** (for links)
Property value	VcDataDefinitionTable	Data definition table

Example Code

```
Dim dataDefinition As VcDataDefinition
Dim dataDefinitionTable As VcDataDefinitionTable

Set dataDefinition = VcGantt1.DataDefinition
Set dataDefinitionTable = dataDefinition.DefinitionTable(vcMaindata)
```

7.20 VcDataDefinitionTable



A **VcDataDefinitionTable** object is an element of a data definition. It represents a table of data definition fields. You can access these fields individually by the methods **FieldByIndex** or **FieldByName** or retrieve them in an iterative loop by the methods **FirstField** and **NextField**. By the **Count** property you can enquire the number of the fields of the table. You can set data field definitions on the property page **Administrate Data Tables**.

Properties

- _NewEnum
- Count

Methods

- CreateDataField
- FieldByIndex
- FieldByName
- FirstField
- NextField

Properties

_NewEnum

Read Only Property of VcDataDefinitionTable

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all data definition field objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method **GetEnumerator** is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim datdeftable As VcDataDefinitionTable

For Each datdeftable In VcGantt1.VcDataDefinition
   Debug.Print datdeftable.Count
Next
```

Count

Read Only Property of VcDataDefinitionTable

This property lets you retrieve the number of fields in the data table. You can add fields by the **Administrate Data Tables** dialog or at run time by the method **CreateDataField**.

	Data Type	Explanation
Property value	Long	Number of fields

Example Code

```
Dim dataDefinition As VcDataDefinition
Dim dataDefinitionTable As VcDataDefinitionTable
Dim numberOfFields As Long

Set dataDefinition = VcGantt1.DataDefinition
Set dataDefinitionTable = dataDefinition.DefinitionTable(vcMaindata)

numberOfFields = dataDefinitionTable.Count
```

Methods

CreateDataField

Method of VcDataDefinitionTable

This method lets you add a new data field at run time to the end of the data table. The data field of the new data field is Integer. You can change the data type by the property **Type** of **VcDefinitionField**.

	Data Type	Explanation
Parameter:		
⇨ newfieldName	String **Possible Values:**	Name of the new field

		Name of the color map
Return value	VcDefinitionField	Data definition field

Example Code

```
Dim dataDefinitionTable As VcDataDefinitionTable
Dim dataDefinitionField As VcDefinitionField

Set dataDefinitionTable =  _ VcGantt1.DataDefinition.DefinitionTable(vcMaindata)
Set dataDefinitionField = dataDefinitionTable.CreateDataField("Description")
dataDefinitionField.Type = vcDefFieldAlphanumericType
VcGantt1.DataTableCollection.Update
```

FieldByIndex

Method of VcDataDefinitionTable

By this method you can access a field of the data definition table by index. A field can be referred to by its name or by its index. The index of the first field is 1. You can set data field definitions in the **Administrate Data Tables** dialog.

	Data Type	Explanation
Parameter:		
⇨ fieldIndex	Integer	Field index
	Possible Values:	Data field index
Return value	VcDefinitionField	Data definition field

Example Code

```
Dim dataDefinitionTable As VcDataDefinitionTable
Dim dataDefinitionField As VcDefinitionField
Set dataDefinitionTable = VcGantt1.DataDefinition.DefinitionTable(vcMaindata)

Set dataDefinitionField = dataDefinitionTable.FirstField
For I = 1 To dataDefinitionTable.Count
      List1.AddItem dataDefinitionField.Name
      Set dataDefinitionField = dataDefinitionTable.FieldByIndex(I)
Next
```

FieldByName

Method of VcDataDefinitionTable

By this method you can get a field of the data table by its name. If a field of the specified name does not exist, a **none** object will be returned (**Nothing** in Visual Basic). A field can be referred to by its name or by its index. You can set data definitions in the **Administrate Data Tables** dialog.

	Data Type	Explanation
Parameter:		
⇨ fieldName	String	Field name
	Possible Values:	Name of the color map
Return value	VcDefinitionField	Data definition field

Example Code

```
Dim dataDefinitionTable As VcDataDefinitionTable
Dim definitionField As VcDefinitionField

Set dataDefinitionTable = VcGantt1.DataDefinition.DefinitionTable(vcMaindata)
Set definitionField = dataDefinitionTable.FieldByName("Start")
```

FirstField

Method of VcDataDefinitionTable

This method can be used to access the first field of a data table and to continue in a forward iteration loop by the method **NextField** for the fields following. If there is no field in the data table, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcDefinitionField	First Data definition field

Example Code

```
Dim dataDefinitionTable As VcDataDefinitionTable
Dim dataDefinitionField As VcDefinitionField

Set dataDefinitionTable = VcGantt1.DataDefinition.DefinitionTable(vcMaindata)
Set dataDefinitionField = dataDefinitionTable.FirstField
```

NextField

Method of VcDataDefinitionTable

This method can be used in a forward iteration loop to retrieve subsequent fields from a data table after initializing the loop by the method **FirstField**. If there is no field left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcDefinitionField	Subsequent data definition field

Example Code

```
Dim dataDefinitionTable As VcDataDefinitionTable
Dim dataDefinitionField As VcDefinitionField

Set dataDefinitionTable = VcGantt1.DataDefinition.DefinitionTable(vcMaindata)

Set dataDefinitionField = dataDefinitionTable.FirstField
While Not dataDefinitionField Is Nothing
      List1.AddItem dataDefinitionField.Name
      Set dataDefinitionField = dataDefinitionTable.NextField
Wend

or

Dim dataDefinitionTable As VcDataDefinitionTable
Dim dataDefinitionField As VcDefinitionField

Set dataDefinitionTable = VcGantt1.DataDefinition.DefinitionTable(vcMaindata)

Set dataDefinitionField = dataDefinitionTable.FirstField
For I = 1 To dataDefinitionTable.Count
      List1.AddItem dataDefinitionField.Name
      Set dataDefinitionField = dataDefinitionTable.NextField
 Next
```

7.21 VcDataDefinitionTable

A VcDataDefinitionTable object is an element of a data definition. It represents a table of data definition fields. You can access these fields individually by the methods **FieldByIndex** or **FieldByName** or retrieve them in an iterative loop by the methods **FirstField** and **NextField**. By the **Count** property you can enquire the number of the fields of the table. You can set data field definitions on the property page **Administrate Data Tables**.

Properties

- _NewEnum
- Count

Methods

- CreateDataField
- FieldByIndex
- FieldByName
- FirstField
- NextField

Properties

_NewEnum

Read Only Property of VcDataDefinitionTable

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all data definition field objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method **GetEnumerator** is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim datdeftable As VcDataDefinitionTable

For Each datdeftable In VcGantt1.VcDataDefinition
   Debug.Print datdeftable.Count
```

```
Next
```

Count

Read Only Property of VcDataDefinitionTable

This property lets you retrieve the number of fields in the data table. You can add fields by the **Administrate Data Tables** dialog or at run time by the method **CreateDataField**.

	Data Type	Explanation
Property value	Long	Number of fields

Example Code

```
Dim dataDefinition As VcDataDefinition
Dim dataDefinitionTable As VcDataDefinitionTable
Dim numberOfFields As Long

Set dataDefinition = VcGantt1.DataDefinition
Set dataDefinitionTable = dataDefinition.DefinitionTable(vcMaindata)

numberOfFields = dataDefinitionTable.Count
```

Methods

CreateDataField

Method of VcDataDefinitionTable

This method lets you add a new data field at run time to the end of the data table. The data field of the new data field is Integer. You can change the data type by the property **Type** of **VcDefinitionField**.

	Data Type	Explanation
Parameter:		
⇨ newfieldName	String	Name of the new field
	Possible Values:	Name of the color map
Return value	VcDefinitionField	Data definition field

Example Code

```
Dim dataDefinitionTable As VcDataDefinitionTable
Dim dataDefinitionField As VcDefinitionField

Set dataDefinitionTable =  _ VcGantt1.DataDefinition.DefinitionTable(vcMaindata)
Set dataDefinitionField = dataDefinitionTable.CreateDataField("Description")
```

```
dataDefinitionField.Type = vcDefFieldAlphanumericType
VcGantt1.DataTableCollection.Update
```

FieldByIndex

Method of VcDataDefinitionTable

By this method you can access a field of the data definition table by index. A field can be referred to by its name or by its index. The index of the first field is 1. You can set data field definitions in the **Administrate Data Tables** dialog.

	Data Type	Explanation
Parameter:		
⇨ fieldIndex	Integer	Field index
	Possible Values:	Data field index
Return value	VcDefinitionField	Data definition field

Example Code

```
Dim dataDefinitionTable As VcDataDefinitionTable
Dim dataDefinitionField As VcDefinitionField
Set dataDefinitionTable = VcGantt1.DataDefinition.DefinitionTable(vcMaindata)

Set dataDefinitionField = dataDefinitionTable.FirstField
For I = 1 To dataDefinitionTable.Count
      List1.AddItem dataDefinitionField.Name
      Set dataDefinitionField = dataDefinitionTable.FieldByIndex(I)
Next
```

FieldByName

Method of VcDataDefinitionTable

By this method you can get a field of the data table by its name. If a field of the specified name does not exist, a **none** object will be returned (**Nothing** in Visual Basic). A field can be referred to by its name or by its index. You can set data definitions in the **Administrate Data Tables** dialog.

	Data Type	Explanation
Parameter:		
⇨ fieldName	String	Field name
	Possible Values:	Name of the color map
Return value	VcDefinitionField	Data definition field

Example Code

```
Dim dataDefinitionTable As VcDataDefinitionTable
Dim definitionField As VcDefinitionField

Set dataDefinitionTable = VcGantt1.DataDefinition.DefinitionTable(vcMaindata)
Set definitionField = dataDefinitionTable.FieldByName("Start")
```

FirstField

Method of VcDataDefinitionTable

This method can be used to access the first field of a data table and to continue in a forward iteration loop by the method **NextField** for the fields following. If there is no field in the data table, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcDefinitionField	First Data definition field

Example Code

```
Dim dataDefinitionTable As VcDataDefinitionTable
Dim dataDefinitionField As VcDefinitionField

Set dataDefinitionTable = VcGantt1.DataDefinition.DefinitionTable(vcMaindata)
Set dataDefinitionField = dataDefinitionTable.FirstField
```

NextField

Method of VcDataDefinitionTable

This method can be used in a forward iteration loop to retrieve subsequent fields from a data table after initializing the loop by the method **FirstField**. If there is no field left, a **none** object will be returned (**Nothing** in Visual Basic).

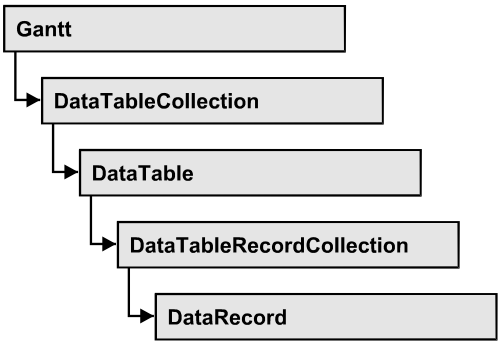
	Data Type	Explanation
Return value	VcDefinitionField	Subsequent data definition field

Example Code

```
Dim dataDefinitionTable As VcDataDefinitionTable
Dim dataDefinitionField As VcDefinitionField

Set dataDefinitionTable = VcGantt1.DataDefinition.DefinitionTable(vcMaindata)

Set dataDefinitionField = dataDefinitionTable.FirstField
While Not dataDefinitionField Is Nothing
      List1.AddItem dataDefinitionField.Name
      Set dataDefinitionField = dataDefinitionTable.NextField
Wend

or
```

```
Dim dataDefinitionTable As VcDataDefinitionTable
Dim dataDefinitionField As VcDefinitionField

Set dataDefinitionTable = VcGantt1.DataDefinition.DefinitionTable(vcMaindata)

Set dataDefinitionField = dataDefinitionTable.FirstField
For I = 1 To dataDefinitionTable.Count
      List1.AddItem dataDefinitionField.Name
      Set dataDefinitionField = dataDefinitionTable.NextField
 Next
```

7.22 VcDataRecord



A data record is the logical base of an object in a Gantt diagram, for example of a node, of a group node, of a link, of an operation or of a task. Objects have specific features, that are described in the fields of the record. For the fields of a data record, descriptions exist that are stored to data table fields. Data records and data table fields are collected in corresponding collection objects, which form a data table.

Properties

- AllData
- DataField
- DataTableName
- ID

Methods

- DeleteDataRecord
- IdentifyObject
- RelatedDataRecord
- UpdateDataRecord

Properties

AllData

Property of VcDataRecord

This property lets you set or retrieve the complete data of a data record. When setting the property, a CSV string (using semicolons as separators) or the data type "variant" are allowed, that contains all data fields of the record in an array. On retrieving the property, a string will be returned.

	Data Type	Explanation
Property value	Variant	All data of the data record

Example Code

```
Dim dataTable As VcDataTable
Dim dataRecCltn As VcDataRecordCollection
Dim dataRecValue() As Variant
Dim dataRecord As VcDataRecord

Set dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata1")
Set dataRecCltn = dataTable.DataRecordCollection
ReDim dataRecValue(dataTable.DataTableFieldCollection.Count)
dataRecValue(0) = 1
dataRecValue(1) = "Node One"

'Variant
Set dataRecord = dataRecCltn.Add(dataRecValue)
'CSV
dataRecord.AllData = "1;Node One;"

dataRecord.UpdateDataRecord
```

DataField

Property of VcDataRecord

This property lets you assign or retrieve data to/from a field of a data record. After the data field was modified by the **DataField** property, the graphical display in the diagram needs to be updated by the **UpdateDataRecord** method.

	Data Type	Explanation
Parameter:		
⇨ index	Integer	Index of data field
	Possible Values:	Data field index
Property value	Variant	Content of the data field

Example Code

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

Set dataTable = VcGantt1.DataTableCollection.FirstDataTable
Set dataRecordCltn = dataTable.DataRecordCollection
Set dataRecord = dataRecordCltn.DataRecordByID(1)

dataRecord.DataField(1) = "Node Two"
dataRecord.UpdateDataRecord
```

DataTableName

Read Only Property of VcDataRecord

This property lets you retrieve the name of the data table that this data record belongs to.

	Data Type	Explanation
Property value	String	Name of the associated table
	Possible Values:	Name of the color map

Example Code

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

Set dataTable = VcGantt1.DataTableCollection.FirstDataTable
Set dataRecordCltn = dataTable.DataRecordCollection
Set dataRecord = dataRecordCltn.DataRecordByID(1)

MsgBox dataRecord.DataTableName
```

ID

Read Only Property of VcDataRecord

By this property you can retrieve the ID of a data record.

	Data Type	Explanation
Property value	String	Data record ID
	Possible Values:	Name of the color map

Example Code

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord
Set dataTable = VcGantt1.DataTableCollection.FirstDataTable
Set dataRecordCltn = dataTable.DataRecordCollection
Set dataRecord = dataRecordCltn.DataRecordByID(1)
MsgBox dataRecord.ID
```

Methods

DeleteDataRecord

Method of VcDataRecord

This method lets you delete a data record.

	Data Type	Explanation
Return value	Boolean	Data record was (true) / was not (false) deleted successfully

Example Code

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

Set dataTable = VcGantt1.DataTableCollection.FirstDataTable
Set dataRecordCltn = dataTable.DataRecordCollection
Set dataRecord = dataRecCltn.DataRecordByID(1)

dataRecord.DeleteDataRecord
```

IdentifyObject

Method of VcDataRecord

This method lets you identify the object having been established via this VcDataRecord object.

The return value will be **true** if a data-based object could be identified, i.e. if a data-based object could be created for the graphic from the record.

	Data Type	Explanation
Parameter:		
⇨ establishedObject Param	Object	Identified object
establishedObjectTypeParam	VcObjectTypeEnum	Object type
	Possible Values:	
	vcObjTypeBox 15	object type **box**
	vcObjTypeCalendarGrid 18	object type **calendar grid**
	vcObjTypeCurve 12	object type **curve**
	vcObjTypeDateLine 9	object type **date line**
	vcObjTypeGroup 7	object type **group**
	vcObjTypeGroupInDiagram 11	object type **group in diagram area**
	vcObjTypeGroupInTable 7	object type **group in table area**
	vcObjTypeHistogram 13	object type **histogram**
	vcObjTypeLayer 8	object type **layer**
	vcObjTypeLinkCollection 3	object type **link collection**
	vcObjTypeNodeInDiagram 2	object type **node in diagram area**
	vcObjTypeNodeInLegend 17	object type **node in legend area**

	vcObjTypeNodeInTable 1 vcObjTypeNone 0 vcObjTypeNumericScale 10 vcObjTypeSummaryNode 14 vcObjTypeTable 4 vcObjTypeTableCaption 5 vcObjTypeTimeScale 6	object type **node in table area** no object object type **numeric scale** object type **summary bar** object type **table** object type **table caption** object type **time scale**
Return value	Boolean	data-based object has been/has not been established

RelatedDataRecord

Method of VcDataRecord

This property lets you relate a data record to another one or retrieve a related data record. When using extended data tables, the data records of a table can be related to the data records of another table by primary keys.

	Data Type	**Explanation**
Parameter:		
⇨ index	Integer **Possible Values:**	Index of data field Data field index
Return value	VcDataRecord	Related data record

Example Code

```
Private Sub VcGantt1_OnNodeLClick(ByVal node As VcGanttLib.VcNode, ByVal
location As VcGanttLib.LocationEnum, ByVal x As Long, ByVal y As Long,
returnStatus As Variant)

    Dim dataTable As VcDataTable
    Dim dataRecordCltn As VcDataRecordCollection
    Dim firstDataRecord As VcDataRecord
    Dim secondDataRecord As VcDataRecord

    Set dataTable = VcGantt1.DataTableCollection.DataTableByIndex(0)
    Set dataRecordCltn = dataTable.DataRecordCollection

    Set firstDataRecord = dataRecordCltn.DataRecordByID(node.DataField(0))
    Set secondDataRecord = firstDataRecord.RelatedDataRecord(2)

    MsgBox secondDataRecord.AllData

End Sub
```

UpdateDataRecord

Method of VcDataRecord

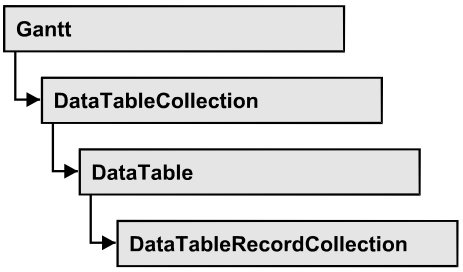
If data fields of a data record were modified by the **DataField** property, the diagram needs to be updated by the **UpdateDataRecord** method.

	Data Type	Explanation
Return value	Boolean	Data record was (true) / was not (false) updated successfully

Example Code

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

Set dataTable = VcGantt1.DataTableCollection.FirstDataTable
Set dataRecordCltn = dataTable.DataRecordCollection
Set dataRecord = dataRecordCltn.DataRecordByID(1)

dataRecord.DataField(1) = "Node Two"
dataRecord.UpdateDataRecord
```

7.23 VcDataRecordCollection



An object of the type VcDataRecordCollection automatically contains all data records of a table. The property **Count** retrieves the number of records present in the collection; the Enumerator object and the methods **FirstDataRecord** and **NextDataRecord** allow to access data records by iteration while by **DataRecordByID** single data records can be accessed. **Add** and **Remove** are basic administering methods, and **Update** lets you refresh the graphical display of objects by data of the records recently modified.

Properties

- _NewEnum
- Count

Methods

- Add
- DataRecordByID
- FirstDataRecord
- GetNewUniqueID
- NextDataRecord
- Remove
- Update

Properties

_NewEnum

Property of VcDataRecordCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all data records. In Visual Basic this property is not indicated, but it can be used by the

command **For Each** *element* **In** *collection*. In .NET languages the method GetEnumerator is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

Set dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
Set dataRecordCltn = dataTable.DataRecordCollection

For Each dataRecord In dataRecordCltn
   Debug.Print dataRecord.AllData
Next dataRecord
```

Count

Read Only Property of VcDataRecordCollection

This property lets you retrieve the number of data records in the DataRecord-Collection object.

	Data Type	Explanation
Property value	Long	Number of data records in the collection object

Example Code

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection

Set dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
Set dataRecordCltn = dataTable.DataRecordCollection
MsgBox "Number of DataRecords: " & dataRecordCltn.Count
```

Methods

Add

Method of VcDataRecordCollection

By this method you can create a data record as a member of the DataRecord-Collection. If the recordDescription did not fail to have a new data record created, the data record will be returned; otherwise a **VcPrimaryKeyNot-UniqueException** will be thrown.

After adding the data record, the method **VcGantt.EndLoading** needs to be invoked to make the modification take effect.

	Data Type	Explanation
Parameter:		
⇨ dataRecordContent	Object	Content of the data record (as an array or a string)
Return value	VcDataRecord	Data record created

Example Code

```
Const Main_ID = 0
Const Main_Name = 1
Const Main_Start = 2
Const Main_Duration = 4

'...

Dim dataTable As VcDataTable
Dim dataRecCltn As VcDataRecordCollection
Dim dataRec1 As VcDataRecord
Dim dataRecVal() As Variant

Set dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
Set dataRecCltn = dataTable.DataRecordCollection

ReDim dataRecVal(dataTable.DataTableFieldCollection.Count)

dataRecVal(Main_ID) = 1
dataRecVal(Main_Name) = "Node 1"
dataRecVal(Main_Start) = DateSerial(2014, 1, 8)
dataRecVal(Main_Duration) = 8
Set dataRec1 = dataRecCltn.Add(dataRecVal)
VcGantt1.EndLoading()

' equivalent
' dataRec1 = dataRecCltn.Add("1;Node 1;01.08.14;;8")
```

DataRecordByID

Method of VcDataRecordCollection

This method lets you access a data record by its identification. If a data record of the specified ID does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

If the identification consists of several fields (composite primary key), this multipart ID has to be specified as follows:

ID=ID1|ID2|ID3

	Data Type	Explanation
Parameter:		
⇨ dataRecordID	String	ID of data record

	Possible Values:	Name of the color map
Return value	VcDataRecord	Data record object

Example Code

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

Set dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
Set dataRecordCltn = dataTable.DataRecordCollection
Set dataRecord = dataRecordCltn.DataRecordByID(0)
```

FirstDataRecord

Method of VcDataRecordCollection

This method can be used to access the initial value, i.e. the first data record of a data record collection, and to continue in a forward iteration loop by the method **NextDataRecord** for the data records following. If there is no data record in the data record collection, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcDataRecord	First data record

Example Code

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

Set dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
Set dataRecordCltn = dataTable.DataRecordCollection
Set dataRecord = dataRecordCltn.FirstDataRecord
```

GetNewUniqueID

Method of VcDataRecordCollection

By this method you can have a unique ID generated for a data record. This method is useful if you wish to add a data record for example by the method **Add** but do not wish to create the ID manually.

	Data Type	Explanation
Return value	Long	New data record ID

NextDataRecord

Method of VcDataRecordCollection

This method can be used in a forward iteration loop to retrieve subsequent data records from a data record collection after initializing the loop by the method **FirstDataRecord**. If there is no data record left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcDataRecord	Subsequent data record

Example Code

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

Set dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
Set dataRecordCltn = dataTable.DataRecordCollection

VcGantt1.SuspendUpdate True

Set dataRecord = dataRecordCltn.FirstDataRecord
While Not dataRecord Is Nothing
   dataRecord.DataField(4) = "10"
   dataRecord.UpdateDataRecord
   Set dataRecord = dataRecordCltn.NextDataRecord
Wend

VcGantt1.SuspendUpdate False
```

Remove

Method of VcDataRecordCollection

This method lets you delete a data record. The method returns **true** after having deleted a data record and **false** when no data record was deleted. The content of the data record is used to identify the object by its identification.

	Data Type	Explanation
Parameter:		
⇨ dataRecordContent	Object	Content of the data record (as an array or a string)
Return value	Boolean	True

Example Code

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

Set dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
Set dataRecordCltn = dataTable.DataRecordCollection
```

```
VcGantt1.SuspendUpdate True

Set dataRecord = dataRecordCltn.FirstDataRecord
While Not dataRecord Is Nothing
   dataRecord.DataField(4) = "10"
   dataRecord.UpdateDataRecord
   Set dataRecord = dataRecordCltn.NextDataRecord
Wend

VcGantt1.SuspendUpdate False
VcGantt1.EndLoading()
```

Update

Method of VcDataRecordCollection

This method updates a data record in the the data record collection if it previously was created by the **Add()** method. If the data record to be updated does not exist, it will then be created by the **Update** method. Also see **VcDataRecordCollection.Add()**.

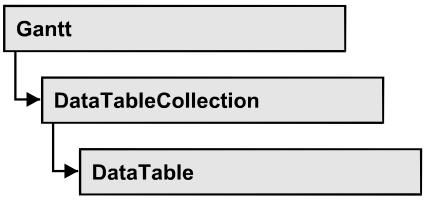
After updating the data record, the method **VcGantt.EndLoading** needs to be invoked to make the modification take effect.

	Data Type	Explanation
Parameter:		
⇨ dataRecordContent	Object	Content of the data record (as an array or a string)
Return value	Boolean	Update successful (True) / not successful (False)

Example Code

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

Set dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
Set dataRecordCltn = dataTable.DataRecordCollection
dataRecordCltn.Update("1;1.8.2017;;8")
VcGantt1.EndLoading()
```

7.24 VcDataTable



A data table comprises **data records**, including their data fields and their contents, and it comprises the descriptions of the record fields, which are called **data table fields**. Data records and data table fields can be processed and iterated over by collection objects.

Data tables on their hand can be processed by a collection object of their own.

Properties

- DataRecordCollection
- DataTableFieldCollection
- Description
- MultiplePrimaryKeysAllowed
- Name

Properties

DataRecordCollection

Read Only Property of VcDataTable

This property returns the DataRecordCollection object of the data table. The collection contains all existing data records of a table. It is empty on the start of the program.

	Data Type	Explanation
Property value	VcDataRecordCollection	DataRecordCollection object

Example Code

```
Dim dataTable As VcDataTable

Set dataTable = VcGantt1.DataTableCollection.FirstDataTable()
MsgBox dataTable.DataRecordCollection.Count
```

DataTableFieldCollection

Read Only Property of VcDataTable

This property returns the DataTableFieldCollection object of the data table. The collection contains the definitions of the fields of a data record of the table. On the start of the program, it holds the data fields that were defined at design time. More data fields can be added at run time by the method **Add** of the object **DataTableFieldCollection**. The definition of data table fields needs to be terminated before data records are filled in the table.

	Data Type	Explanation
Property value	VcTableFieldCollection	DataTableFieldCollection object

Example Code

```
Dim dataTable As VcDataTable

Set dataTable = VcGantt1.DataTableCollection.DataTableByIndex(0)
MsgBox dataTable.DataTableFieldCollection.Count
```

Description

Property of VcDataTable

This property lets you set or retrieve the description of the data table. Names of objects, for example of the table, that contain some information on the object, often are long and cannot be displayed fully in previews; so their benefit is limited. To use the opportunity of short names without having to abandon the information of a long name, you can store additional information to this field. Its contents will be displayed in the data table dialog.

	Data Type	Explanation
Property value	String **Possible Values:**	Description of the data table **Default value:** Empty string Name of the color map

Example Code

```
Dim dataTable As VcDataTable

Set dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
dataTable.Description = "This table contains data for nodes"
```

MultiplePrimaryKeysAllowed

Property of VcDataTable

This property lets you set or retrieve whether using a composed primary keys is permited.

	Data Type	Explanation
Property value	Boolean	Use of composite primary keys allowed (true)/not allowed (false) **Default value:** False
	Possible Values:	Group invisible/visible group nodes are/are not visible

Name

Property of VcDataTable

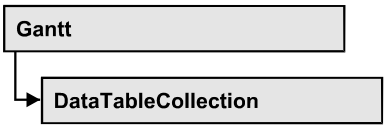
This property lets you set or retrieve the name of the data table. The name of a data table has to set by obligation; beside, it has to be unique. An empty character string is not allowed. Upper and lower case characters are accepted as different. By the method **DataTableByName** of the object **DataTable-Collection** you can retrieve a reference to the data table object.

	Data Type	Explanation
Property value	String	Name of the data table **Default value:** Empty string
	Possible Values:	Name of the color map

Example Code

```
Dim dataTable As VcDataTable

Set dataTable = VcGantt1.DataTableCollection.DataTableByIndex(0)
MsgBox dataTable.Name
```

7.25 VcDataTableCollection



An object of the type VcDataTableCollection holds a collection of tables. The property **Count** retrieves the number of tables present in the collection; the Enumerator object and the methods **FirstDataTable** and **NextDataTable** allow to access tables by iteration while by **DataTableByName** and **DataTableByindex** single tables can be accessed. **Add** and **Copy** are basic administrating methods, and **Update** makes the recent modifications of the data structures known to the XGantt object, which is equivalent to an update.

Properties

- _NewEnum
- Count

Methods

- Add
- Copy
- DataTableByIndex
- DataTableByName
- FirstDataTable
- NextDataTable
- Update

Properties

_NewEnum

Property of VcDataTableCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all data tables. In Visual Basic this property never is displayed, but it can be addressed by the command **For Each** *element* **In** *collection*. In .NET languages the method GetEnumerator is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable

Set dataTableCltn = VcGantt1.DataTableCollection
For Each dataTable In dataTableCltn
   List1.AddItem (dataTable.Name)
Next
```

Count

Property of VcDataTableCollection

This property lets you retrieve the number of data tables in the DataTable-Collection object.

	Data Type	Explanation
Property value	Long	Number of data tables in the collection object

Example Code

```
Dim dataTableCltn As VcDataTableCollection

Set dataTableCltn = VcGantt1.DataTableCollection
MsgBox (dataTableCltn.Count)
```

Methods

Add

Method of VcDataTableCollection

By this method you can create a data table as a member of the DataTable-Collection. If the name was not used before,an object of the type **VcData-Table** will be returned; otherwise **Nothing** in Visual Basic or **0** in other languages. Only if the DummyObjec3 property **ExtendedDataTables** is set to **True**, tables can be added. In total, 90 data tables can be added at maximum.

	Data Type	Explanation
Parameter:		
⇨ dataTableName	String **Possible Values:**	Name of the new data table

		Name of the color map
Return value	VcDataTable	Data table generated

Example Code

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable

Set dataTableCltn = VcGantt1.DataTableCollection
Set dataTable = dataTableCltn.Add("Resources")
dataTableCltn.Update
```

Copy

Method of VcDataTableCollection

This method lets you copy a data table. Probably existing data records are not copied, just the definition fields. Only if the VcNet property **ExtendedDataTables** was set to **True**, data tables can be copied. If the data table could be copied, a new object of the type **VcDataTable** will be returned; otherwise **Nothing** in Visual Basic or **0** in other languages. The table names are case sensitive.

	Data Type	**Explanation**
Parameter:		
⇨ dataTableName	String **Possible Values:**	Name of the data table to be copied (source table) Name of the color map
⇨ newDataTableName	String **Possible Values:**	Name of the data table to be generated (target table) Name of the color map
Return value	VcDataTable	Data table object generated

Example Code

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable

Set dataTableCltn = VcGantt1.DataTableCollection
Set dataTable = dataTableCltn.Copy("Resources", "NewResources")
dataTableCltn.Update
```

DataTableByIndex

Method of VcDataTableCollection

This method lets you access a data table by its index. The index of the first table is 0. If a data table of the specified index does not exist, a **none** object will be returned (**Nothing** in Visual Basic or **0** in other languages).

	Data Type	Explanation
Parameter:		
⇨ index	Integer	Index of the data table
	Possible Values:	Data field index
Return value	VcDataTable	Data table object returned

Example Code

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable

Set dataTableCltn = VcGantt1.DataTableCollection
Set dataTable = dataTableCltn.DataTableByIndex(2)
MsgBox (dataTable.Name)
```

DataTableByName

Method of VcDataTableCollection

This method lets you access a data table by its name. If a data table of the specified name does not exist, a **none** object will be returned (**Nothing** in Visual Basic or **0** in other languages).

	Data Type	Explanation
Parameter:		
⇨ dataTableName	String	Name of the data table
	Possible Values:	Name of the color map
Return value	VcDataTable	Data table object returned

Example Code

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable

Set dataTableCltn = VcGantt1.DataTableCollection
Set dataTable = dataTableCltn.DataTableByName("Resources")
MsgBox (dataTable.Description)
```

FirstDataTable

Method of VcDataTableCollection

This method can be used to access the initial value, i.e. the first data table of a data table collection, and to continue in a forward iteration loop by the method **NextDataTable** for the data tables following. If there is no data table in the data table collection, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcDataTable	First data table

Example Code

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable

Set dataTableCltn = VcGantt1.DataTableCollection
Set dataTable = dataTableCltn.FirstDataTable
```

NextDataTable

Method of VcDataTableCollection

This method can be used in a forward iteration loop to retrieve subsequent data tables from a data table collection after initializing the loop by the method **FirstDataTable**. If there is no data table left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcDataTable	Subsequent data table

Example Code

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable
Dim i As Integer

Set dataTableCltn = VcGantt1.DataTableCollection
Set dataTable = dataTableCltn.FirstDataTable
For i = 0 To dataTableCltn.Count
   List1.AddItem (dataTable.Name)
   Set dataTable = dataTableCltn.NextDataTable
Next i
```

Update

Method of VcDataTableCollection

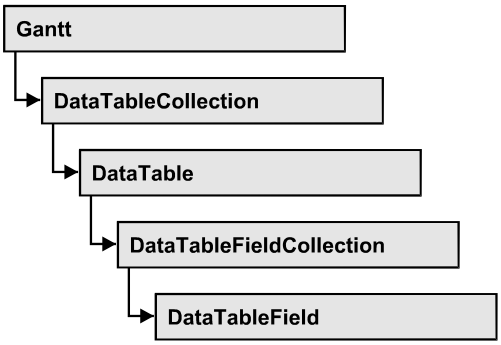
This method lets you update recent modifications of the data structures. It makes the modifications on data table definitions and on data table fields become operative in the VARCHART component and avoids individual updates after several modifications.

	Data Type	Explanation
Return value	Boolean	Update successful (True) / not successful (False)

Example Code

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable

dataTableCltn = VcGantt1.DataTableCollection
dataTable = dataTableCltn.Add("Resources")
dataTable.DataTableFieldCollection.Add ("Id")
dataTableCltn.Update
```

7.26 VcDataTableField



An object of the type **VcDataTableField** defines the properties of a data field in a data record. Part of the definition of a data table field are its name, its data type and whether it represents the primary key, by which a data record can be uniquely identified. For example, by referring to the primary key, other data tables can relate to a data table. To create a relation, a table needs to specify the primary key of a different table by the property **RelationshipFieldIndex**.

The DataTableField objects of a data table are administered by the object **DataTableFieldCollection**.

Properties

- DataTableName
- DateFormat
- Editable
- Hidden
- Index
- Name
- PrimaryKey
- RelationshipFieldIndex
- Type

Properties

DataTableName

Read Only Property of VcDataTableField

This property lets you retrieve the name of the associated data table.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the data table Name of the color map

Example Code

```
Dim dataTable As VcDataTable

Set dataTable = VcGantt1.DataTableCollection.FirstDataTable
MsgBox dataTable.DataTableFieldCollection.FirstDataTableField.DataTableName
```

DateFormat

Property of VcDataTableField

This property lets you set or retrieve the date format of the record field that is specified by the property **RelationshipFieldIndex**. The date format is used when reading or storing CSV files and when the format type **String** is used when adding a data record by the method **Add**. This property only works if the data type of the field was set to **vcDataTableFieldDateTime**.

Note: Remember to set the property **Type** before setting the property **DateFormat**.

	Data Type	Explanation
Property value	String **Possible Values:**	Date format {DMYhms:;./} **Default value:** DD.MM.YYYY hh:mm:ss Name of the color map

Example Code

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

Set dataTable = VcGantt1.DataTableCollection.DataTableByName("Operation")
Set dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Start")
dataTableField.Type = vcDataTableFieldDateTimeType
'DateFormat = "01.12.2014"
dataTableField.DateFormat = "DD.MM.YYYY"
```

Editable

Property of VcDataTableField

This property lets you set or retrieve whether the record field should be editable at run time in the chart table and in the dialog **EditNode**.

	Data Type	Explanation
Property value	Boolean	Field editable (True) / not editable (False) **Default value:** True
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

Set dataTable = VcGantt1.DataTableCollection.DataTableByName("Operation")
Set dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Start")
dataTableField.Editable = False
VcGantt1.DataTableCollection.Update
```

Hidden

Property of VcDataTableField

This property lets you set or retrieve whether the data field should be hidden at run time in the dialogs **EditNode** and **EditLink**.

	Data Type	Explanation
Property value	Boolean	Field hidden (True) / not hidden (False) **Default value:** False
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

Set dataTable = VcGantt1.DataTableCollection.DataTableByName("Operation")
Set dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Start")
dataTableField.Hidden = True
VcGantt1.DataTableCollection.Update
```

Index

Read Only Property of VcDataTableField

This property lets you retrieve the index of the data table field in the associated data table.

	Data Type	Explanation
Property value	Integer	Index of the data table field
	Possible Values:	Data field index

Name

Property of VcDataTableField

This property lets you set or retrieve the name of the record field. The name is indicated in runtime dialogs such as the **EditNode** dialog. Accessing a field by the API although requires its index that the field has within the **DataTableFieldCollection** object.

	Data Type	Explanation
Property value	String	Name of the field **Default value:** Empty string
	Possible Values:	Name of the color map

Example Code

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

Set dataTable = VcGantt1.DataTableCollection.DataTableByName("Operation")
Set dataTableField = dataTable.DataTableFieldCollection.Add("Start")
VcGantt1.DataTableCollection.Update
```

PrimaryKey

Property of VcDataTableField

This property lets you set or retrieve whether this field contains the primary key, which is used for the unique identification of a data record. In a data table, only one of the fields that were defined can be the primary key. Within the same table, assigning the primary key function to a field automatically cancels the previous assignment. A primary key is required in a table if records of a different table are to depend on the records of the former one.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	The field serves (True) / does not serve (False) as a primary key. **Default value:** False Group invisible/visible group nodes are/are not visible

Example Code

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField
Dim isPrimaryKey As Boolean

Set dataTable = VcGantt1.DataTableCollection.DataTableByName("Operation")
Set dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Id")
dataTableField.PrimaryKey = True
VcGantt1.DataTableCollection.Update
```

RelationshipFieldIndex

Property of VcDataTableField

This property lets you combine a data field and its data description. For this, please set the index of the data record field to which the settings of this data table field shall refer.

	Data Type	Explanation
Property value	Long	Index of the record field to which the data definition of the data table field refers. **Default value:** -1

Example Code

```
Dim dataTableTask As VcDataTable
Dim dataTaskFieldId As VcDataTableField
Dim dataTaskFieldName As VcDataTableField
Dim dataTableOperation As VcDataTable
Dim dataOperationFieldId As VcDataTableField
Dim dataOperationFieldName As VcDataTableField
Dim dataOperationFieldTaskId As VcDataTableField

'Create table Task
dataTableTask = VcGantt1.DataTableCollection.Add("Task")
dataTaskFieldId = dataTableTask.DataTableFieldCollection.Add("Id")
dataTaskFieldId.PrimaryKey = True
dataTaskFieldName = dataTableTask.DataTableFieldCollection.Add("Name")
dataTaskFieldName.Type = vcDataTableFieldAlphanumericType

'Create table Operation
dataTableOperation = VcGantt1.DataTableCollection.Add("Operation")
dataOperationFieldId = dataTableOperation.DataTableFieldCollection.Add("Id")
dataOperationFieldId.PrimaryKey = True
dataOperationFieldName = dataTableOperation.DataTableFieldCollection.Add("Name")
dataOperationFieldName.Type = vcDataTableFieldAlphanumericType
```

```
dataOperationFieldTaskId =
dataTableOperation.DataTableFieldCollection.Add("TaskId")
dataOperationFieldTaskId.Type = vcDataTableFieldIntegerType

'Link tables Task and Operations
dataOperationFieldTaskId.RelationshipFieldIndex =
VcGantt1.DetectFieldIndex("Task", "Id")
```

Type

Property of VcDataTableField

This property lets you set or retrieve the data type of the field.

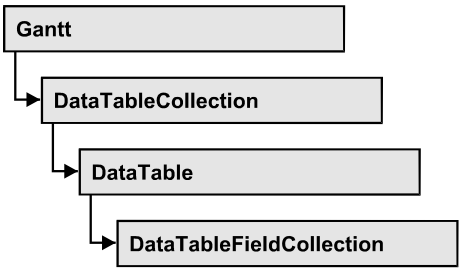
Note: Setting the property **Type** may change the property **DateFormat**. By setting this property to **vcDataTableAlphanumeric** or to **vcDataTable-FieldInteger** the date format probably set will change to "".

	Data Type	Explanation
Property value	DataTableFieldTypeEnum	Data type of the field, can contain 512 characters maximum **Default value:** VcDataTableFieldIntegerType
	Possible Values: vcDataTableFieldAlphanumericType 1 vcDataTableFieldDateTimeType 3 vcDataTableFieldIntegerType 2	 Data type **alphanumeric**: "" Data type **date** : DD.MM.YYYY Data type **integer** (32 bits): ""

Example Code

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

Set dataTable = VcGantt1.DataTableCollection.DataTableByName("Operation")
Set dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Start")
dataTableField.Type = vcDataTableFieldDateTimeType
VcGantt1.DataTableCollection.Update
```

7.27 VcDataTableFieldCollection



An object of the type VcDataTableFieldCollection automatically contains all data fields of a data table. The property **Count** retrieves the number of fields present in the collection; the Enumerator object and the methods **FirstDataField** and **NextDataField** allow to access data fields by iteration while by **DataFieldByName** and **DataFieldByIndex** single data fields can be accessed. **Add** and **Copy** represent basic administering methods.

Properties

- _NewEnum
- Count

Methods

- Add
- Copy
- DataTableFieldByIndex
- DataTableFieldByName
- FirstDataTableField
- NextDataTableField

Properties

_NewEnum

Property of VcDataTableFieldCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all data table fields objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method **GetEnumerator** is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

Set dataTable = VcGantt1.DataTableCollection.FirstDataTable
For Each dataTableField In dataTable.DataTableFieldCollection
   List1.AddItem (dataTableField.Name)
Next
```

Count

Read Only Property of VcDataTableFieldCollection

This property lets you retrieve the number of data table fields in the DataTableFieldCollection object.

	Data Type	Explanation
Property value	Long	Number of data table fields in the collection object

Example Code

```
Dim dataTable As VcDataTable

Set dataTable = VcGantt1.DataTableCollection.FirstDataTable
MsgBox ("Number of data fields: " &  dataTable.DataTableFieldCollection.Count)
```

Methods

Add

Method of VcDataTableFieldCollection

By this method you can create a data table field as a member of the DataTableFieldCollection. If the name was not used before, the new data field will be returned; otherwise "Nothing" (Visual Basic) or "0" (other languages) will be returned. You can add at maximum 9,999 fields to a table.

	Data Type	Explanation
Parameter:		
⇨ dataTableFieldName	String	Name of the data table field to be generated
	Possible Values:	Name of the color map

Return value	VcDataTableField	Data table field generated

Example Code

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

Set dataTable = VcGantt1.DataTableCollection.FirstDataTable
Set dataTableField = dataTable.DataTableFieldCollection.Add("Priority")
VcGantt1.DataTableCollection.Update
```

Copy

Method of VcDataTableFieldCollection

This method lets you copy a data table field. The field is identified by its name.

	Data Type	Explanation
Parameter:		
⇨ dataTableFieldName	String **Possible Values:**	Name of the data table field to be copied (source field) Name of the color map
⇨ newDataTableFieldName	String **Possible Values:**	Name of the data table field to be generated (target field) Name of the color map
Return value	VcDataTableField	Data table field generated

Example Code

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

Set dataTable = VcGantt1.DataTableCollection.FirstDataTable
Set dataTableField = dataTable.DataTableFieldCollection.Copy("Name", "NewName")
VcGantt1.DataTableCollection.Update
```

DataTableFieldByIndex

Method of VcDataTableFieldCollection

This method lets you access a data table field by its index. If a data field does not exist at the index specified, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter:		
⇨ Index	Integer	Index of data table field
	Possible Values:	Data field index
Return value	VcDataTableField	Data table field returned

Example Code

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

Set dataTable = VcGantt1.DataTableCollection.FirstDataTable
Set dataTableField = dataTable.DataTableFieldCollection.DataTableFieldByIndex(1)
MsgBox (dataTableField.Name)
```

DataTableFieldByName

Method of VcDataTableFieldCollection

This method lets you access a data table field by its name. If a field of the specified name does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter:		
⇨ dataTableFieldName	String	Name of data table field
	Possible Values:	Name of the color map
Return value	VcDataTableField	Data table field returned

Example Code

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

Set dataTable = VcGantt1.DataTableCollection.FirstDataTable
Set dataTableField = dataTable.DataTableFieldCollection.DataTableFieldBy("Name")
dataTableField.Editable = False
VcGantt1.DataTableCollection.Update
```

FirstDataTableField

Method of VcDataTableFieldCollection

This method can be used to access the initial value, i.e. the first data table field of a data table field collection, and to continue in a forward iteration loop by the method **NextDataTableField** for the fields following. If there is

no field in the data table field collection, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcDataTableField	First data table field

Example Code

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

Set dataTable = VcGantt1.DataTableCollection.FirstDataTable
Set dataTableField = dataTable.DataTableFieldCollection.FirstDataTableField
```

NextDataTableField

Method of VcDataTableFieldCollection

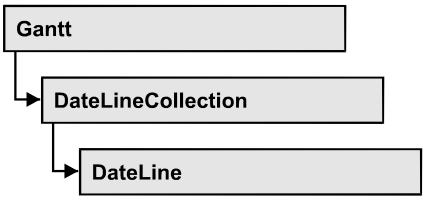
This method can be used in a forward iteration loop to retrieve subsequent data table fields from a data table field collection after initializing the loop by the method **FirstDataTableField**. If there is no field left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcDataTableField	Subsequent data table field

Example Code

```
Dim dataTable As VcDataTable
Dim dataTableFieldCltn As VcDataTableFieldCollection
Dim dataTableField As VcDataTableField
Dim i As Integer

Set dataTable = VcGantt1.DataTableCollection.FirstDataTable
Set dataTableFieldCltn = dataTable.DataTableFieldCollection
Set dataTableField = dataTableFieldCltn.FirstDataTableField
For i = 0 To dataTableFieldCltn.Count
   List1.AddItem (dataTableField.Name)
   Set dataTableField = dataTableFieldCltn.NextDataTableField
Next i
```

7.28 VcDateLine



An object of the type VcDateLine is a time-orientated vertical line in a Gantt diagram that marks a date.

Properties

- AlwaysCurrentDate
- Date
- DateDataFieldIndex
- Font
- FontColor
- Identifiable
- LabelPosition
- LineColor
- LineThickness
- LineType
- Moveable
- Name
- Priority
- SnapTarget
- Specification
- Text
- TurningAnnotationEnabled
- UpdateBehaviorName
- Visible
- VisibleDataFieldIndex
- VisibleMapName

Methods

- PutInOrderAfter

Properties

AlwaysCurrentDate

Read Only Property of VcDateLine

This property lets you set or retrieve whether a date line always displays the current date and time at the time of the start of VARCHART ActiveX. This property can be set in the **Specify Date Lines** dialog.

	Data Type	Explanation
Property value	Boolean	Property active/not active **Default value:** False
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
Dim dateLine As VcDateLine
Dim DateLineTimer As Timer

Set dateLine = VcGantt1.DateLineCollection.DateLineByName("DateLine1")

If dateLine.AlwaysCurrentDate = True Then DateLineTimer.Enabled = True
```

Date

Property of VcDateLine

This property lets you specify or enquire the position of a date line. Please note: date and time must be separated by a blank. This property also can be set in the **Specify Date Lines** dialog.

	Data Type	Explanation
Property value	Date/Time	Date {1.1.1970...31.12.2035} **Default value:** none or current date

Example Code

```
Dim dateLine As VcDateLine

Set dateLine = VcGantt1.DateLineCollection.DateLineByName("DateLine1")
dateLine.Date = "30.09.14 12:00:00"
```

DateDataFieldIndex

Property of VcDateLine

This property lets you set or retrieve the index of the data field containing the date of the indvidual date line.

	Data Type	Explanation
Property value	Long	Index of the data field which contains the date

Font

Property of VcDateLine

This property lets you set or retrieve all font attributes of the date line and can also be set in the **Edit Date Line** dialog.

	Data Type	Explanation
Property value	StdFont	Font attributes of the date line texts

Example Code

```
Dim newFont As New StdFont

newFont.Name = "Times New Roman"
newFont.Italic = True
newFont.Bold = True
newFont.Size = 12

Set VcDateLine.Font = newFont
```

FontColor

Property of VcDateLine

This property lets you set or retrieve the font color of the date line and can also be set in the **Edit Date Line** dialog.

	Data Type	Explanation
Property value	Color	RGB color values

Example Code

```
Dim dateLine As VcDateLine

Set dateLine = VcGantt1.DateLineCollection.DateLineByName("DateLine1")
dateLine.FontColor = RGB(120, 100, 150)
```

Identifiable

Property of VcDateLine

This property lets you set or retrieve whether or not a date line grid can be identified. If this property was set to **True**, the date line can be identified by the VcGantt method **IdentifyObjectAt**.

This property can also be set in the **Specify Date lines** dialog.

	Data Type	Explanation
Property value	Boolean	Date line can / cannot be identified **Default value:** False
	Possible Values:	Group invisible/visible group nodes are/are not visible

LabelPosition

Read Only Property of VcDateLine

This property lets you specify or retrieve the position at which the annotation of the date line shall be displayed.

	Data Type	Explanation
Property value	LabelPositionEnum	label position of date line

LineColor

Property of VcDateLine

This property lets you set or retrieve the line color of a date line and can also be set in the **Edit Date Line** dialog.

	Data Type	Explanation
Property value	Color	RGB color values **Default value:** 255. Visual Basic: RGB (255, 0, 0)

Example Code

```
Dim dateLine As VcDateLine

Set dateLine = VcGantt1.DateLineCollection.DateLineByName("DateLine1")
dateLine.LineColor = RGB(120, 100, 150)
```

LineThickness

Property of VcDateLine

This property lets you set or retrieve the line thickness of a date line. If you set this property to values between 1 and 4, an absolute line thickness is defined in pixels. Irrespective of the zoom factor a line will always show the same line thickness in pixels. When printing though, the line thickness is adapted for the sake of legibility and becomes dependent of the zoom factor:

Value	Points	mm
1	1/2 point	0.09 mm
2	1 point	0.18 mm
3	3/2 points	0.26 mm
4	2 points	0.35 mm

A point equals 1/72 inch and represents the unit of the font size.

If you set this property to values between 5 and 1,000, the line thickness is defined in 1/100 mm, so the lines will be displayed in a true thickness in pixels that depends on the zoom factor.

	Data Type	Explanation
Property value	Integer	Line thickness LineType {1...4}: line thickness in pixels LineType {5...1000}: line thickness in 1/100 mm **Default value:** As defined in the dialog
	Possible Values:	Data field index

Example Code

```
Dim dateLine As VcDateLine

Set dateLine = VcGantt1.DateLineCollection.DateLineByName("DateLine1")
dateLine.LineType = vcSolid
dateLine.LineThickness = 3
```

LineType

Property of VcDateLine

This property lets you set or retrieve the line type of a date line. This property also can be set in the **Edit Date Line** dialog.

	Data Type	Explanation
Property value	LineTypeEnum	Line type **Default value:** vcSolid
	Possible Values:	
	vcDashed 4	Line dashed
	vcDashedDotted 5	Line dashed-dotted
	vcDotted 3	Line dotted
	vcLineType0 100	Line Type 0
	vcLineType1 101	Line Type 1
	vcLineType10 110	Line Type 10
	vcLineType11 111	Line Type 11
	vcLineType12 112	Line Type 12
	vcLineType13 113	Line Type 13
	vcLineType14 114	Line Type 14
	vcLineType15 115	Line Type 15
	vcLineType16 116	Line Type 16
	vcLineType17 117	Line Type 17
	vcLineType18 118	Line Type 18
	vcLineType2 102	Line Type 2
	vcLineType3 103	Line Type 3
	vcLineType4 104	Line Type 4
	vcLineType5 105	Line Type 5
	vcLineType6 106	Line Type 6
	vcLineType7 107	Line Type 7
	vcLineType8 108	Line Type 8
	vcLineType9 109	Line Type 9
	vcNone 1	No line type
	vcNotSet -1	No line type assigned
	vcSolid 2	Line solid

Example Code

```
Dim dateLine As VcDateLine

Set dateLine = VcGantt1.DateLineCollection.DateLineByName("DateLine1")
dateLine.LineType = vcSolid
```

Moveable

Property of VcDateLine

This property lets you set or retrieve whether a date line can be moved interactively. This property also can be set in the **Specify Date Lines** dialog.

	Data Type	Explanation
Property value	Boolean	Moveable (True)/ not moveable (False) **Default value:** True
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
Dim dateLine As VcDateLine

Set dateLine = VcGantt1.DateLineCollection.DateLineByName("DateLine1")
If chkMoveable.Value = vbUnchecked Then
   dateLine.Moveable = False
Else
   dateLine.Moveable = True
End If
```

Name

Read Only Property of VcDateLine

This property lets you retrieve the name of a date line.

	Data Type	Explanation
Property value	String	Name
	Possible Values:	Name of the color map

Example Code

```
Dim dateLineCltn As VcDateLineCollection
Dim dateLine As VcDateLine

Set dateLineCltn = VcGantt1.DateLineCollection

For Each dateLine in dateLineCltn
   ListBox.AddItem (dateLine.Name)
Next dateLine
```

Priority

Property of VcDateLine

This property lets you specify or retrieve the priority of a date line. If two objects are located at the same position in the diagram, the object of higher priority is displayed in front of the objects of lower priority. By default, grids are of the lowest priority. Nodes are assigned the value 0 and thus the highest priority of all objects. By default, date lines are displayed behind nodes, but in front of calendar grids and date line grids. If you want a date line to be displayed in front of the nodes, you must set its priority to a positive value. This property also can be set in the **Specify Date Lines** dialog.

	Data Type	Explanation
Property value	Integer	Priority value **Default value:** 0
	Possible Values:	Data field index

Example Code

```
Dim dateLine As VcDateLine

Set dateLine = VcGantt1.DateLineCollection.DateLineByName("dateLine1")
dateLine.Priority = 10
```

SnapTarget

Property of VcDateLine

This property lets you set or retrieve whether this date line has a snap target at the date.

	Data Type	Explanation
Property value	Boolean	Snap target is/is not defined at the date of this date line
	Possible Values:	Group invisible/visible group nodes are/are not visible

Specification

Read Only Property of VcDateLine

This property lets you retrieve the specification of a date line. A specification is a string that contains legible ASCII characters from 32 to 127 only, so it

can be stored without problems to text files or data bases. This allows for persistency. A specification can be used to create a date line by the method **VcDateLineCollection.AddBySpecification**.

	Data Type	Explanation
Property value	String **Possible Values:**	Specification of the date line Name of the color map

Text

Property of VcDateLine

This property lets you set or retrieve an annotation text for the date line. This property also can be set in the **Specify Date Lines** dialog.

	Data Type	Explanation
Parameter: ⇨ Rückgabewert	String **Possible Values:**	Annotation Name of the color map
Property value	String **Possible Values:**	Annotation text of the date line Name of the color map

Example Code

```
Dim dateLine As VcDateLine

Set dateLine = VcGantt1.DateLineCollection.DateLineByName("DateLine1")
dateLine.Text = "Stichtag"
```

TurningAnnotationEnabled

Property of VcDateLine

This property lets you specify or retrieve whether the annotation of the date line is turned by 90 degrees.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	Annotation of date line is/is not turned by 90 degrees Group invisible/visible

		group nodes are/are not visible

UpdateBehaviorName

Property of VcDateLine

This property lets you set or retrieve the name of the UpdateBehavior.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the UpdateBehavior Name of the color map

Visible

Property of VcDateLine

This property lets you set or retrieve the visibility of a date line. This property also can be set in the **Specify Date Lines** dialog.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	Date line visible/invisible **Default value:** True Group invisible/visible group nodes are/are not visible

Example Code

```
Dim dateLine As VcDateLine

Set dateLine = VcGantt1.DateLineCollection.DateLineByName("dateLine1")
If chkVisible.Value = vbUnchecked Then
    dateLine.Visible = False
Else
    dateLine. Visible = True
End If
```

VisibleDataFieldIndex

Read Only Property of VcDateLine

This property lets you set or retrieve the index of the data field to assign a visibility mode to the indvidual date line. The property can also be set in the **Specify Date Lines** dialog.

	Data Type	Explanation

VisibleMapName

Read Only Property of VcDateLine

This property lets you set or retrieve the name of a map (type vcTextMap) to set the visibility mode. If set to "", no map will be used. If a map name and additionally a data field index is specified by the property **VisibilityDataFieldIndex**, the visibility mode is selected by the map. This property also can be set in the **Specify Date lines** dialog. If no data field entry from the map applies, the visibility will adopt the value set in the dialog.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the map that contains the visibility mode Name of the color map

Methods

PutInOrderAfter

Method of VcDateLine

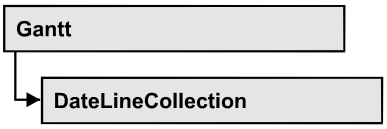
This method lets you set the date line behind a date line specified by name, within the DateLineCollection. If you set the name to "", the date line will be put in the first position. The order of the date lines within the collection determines the order by which they are displayed.

	Data Type	Explanation
Parameter: ⇦ refName	String **Possible Values:**	Name of the date line behind which the current date line is to be put. Name of the color map
Return value	Void	

Example Code

```
Dim datLinCltn As VcDateLineCollection
Dim datLin1 As VcDateLine
Dim datLin2 As VcDateLine

datLinCltn = VcGantt1.DateLineCollection()
datLin1 = datLinCltn.Add("datLin1")
datLin2 = datLinCltn.Add("datLin2")
datLin1.PutInOrderAfter("datLin2")
datLinCltn.Update()
```

7.29 VcDateLineCollection



An object of the type **VcDateLineCollection** automatically contains all available date lines. You can access all objects in an iterative loop by **For Each dateLine In DateLineCollection** or by the methods **First...** and **Next...**. You can access a single date line using the methods **DateLineBy-Name** and **DateLineByIndex**. The number of date lines in the collection object can be retrieved by the property **Count**. The methods **Add**, **Copy** and **Remove** allow to handle the date lines in the corresponding way.

Properties

- _NewEnum
- Count

Methods

- Add
- AddBySpecification
- Copy
- DateLineByIndex
- DateLineByName
- FirstDateLine
- NextDateLine
- Remove
- Update

Properties

_NewEnum

Read Only Property of VcDateLineCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all date line objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method

GetEnumerator is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim dateline As VcDateLine

For Each dateline In VcGantt1DateLineCollection
   Debug.Print dateline.Name
Next
```

Count

Read Only Property of VcDateLineCollection

This property lets you retrieve the number of date lines contained in the date line collection.

	Data Type	Explanation
Property value	Long	Number of data lines

Example Code

```
Dim numberOfDateLines As Long

numberOfDateLines = VcGantt1.DateLineCollection.Count
```

Methods

Add

Method of VcDateLineCollection

By this method you can create a date line as a member of the DateLineCollection. If the name was not used before, the new date line object will be returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned. To make the new date line visible in the diagram, the date line collection needs to be updated by the **Update** call.

	Data Type	Explanation
Parameter:		
⇨ dateLineName	String **Possible Values:**	name of date line

		Name of the color map
Return value	VcDateLine	New date line object

Example Code

```
Set newDateLine = VcGantt1.DateLineCollection.Add("DateLine1")
```

AddBySpecification

Method of VcDateLineCollection

By this method you can create a date line by a date line specification. This way of creating allows date line objects to become persistent. The specification of a data line can be saved and re-loaded (see VcDateLine property **Specification**). In a subsequent session, the date line can be created again from the specification and is identified by its name. To make the new date line visible in the diagram, the date line collection needs to be updated by the **Update** call.

	Data Type	**Explanation**
Parameter:		
⇨ specification	String	date line specification
	Possible Values:	Name of the color map
Return value	VcDateLine	New date line object

Copy

Method of VcDateLineCollection

By this method you can copy a date line. If the date line that is to be copied exists, and if the name for the new date line does not yet exist, the new date line object is returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned. To make the copied date line visible in the diagram, the date line collection needs to be updated by the **Update** call.

	Data Type	**Explanation**
Parameter:		
⇨ dateLineName	String	Name of the date line to be copied
	Possible Values:	Name of the color map
⇨ newDateLineName	String	Name of the new date line

	Possible Values:	Name of the color map
Return value	VcDateLine	Date line object

Example Code

```
Dim DateLineCltn As VcDateLineCollection
Dim dateLine As VcDateLine

Set dateLineCltn = VcGantt1.DateLineCollection
Set dateLine = dateLineCltn.Copy("DateLineOne", "NewDateLine")
dateLineCltn.Update
```

DateLineByIndex

Method of VcDateLineCollection

This method lets you access a date line by its index. If a date line of the specified index does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	**Explanation**
Parameter:		
⇨ index	Integer	Index of the date line
	Possible Values:	Data field index
Return value	VcDateLine	Date line object returned

Example Code

```
Dim dateLineCltn As VcDateLineCollection
Dim dateLine As VcDateLine

Set dateLineCltn = VcGantt1.DateLineCollection
Set dateLine = dateLineCltn.DateLineByIndex(0)
MsgdateLine DateLine.Name
```

DateLineByName

Method of VcDateLineCollection

By this method you can retrieve a date line by its name. If a date line of the specified name does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	**Explanation**
Parameter:		
⇨ dateLineName	String	Name of the date line

	Possible Values:	Name of the color map
Return value	VcDateLine	Date line

Example Code

```
Dim dateLineCltn As VcDateLineCollection
Dim dateLine As VcDateLine

Set dateLineCltn = VcGantt1.DateLineCollection
Set dateLine = dateLineCltn.DateLineByName("DateLineOne")
MsgdateLine DateLine.Name
```

FirstDateLine

Method of VcDateLineCollection

This method can be used to access the initial value, i.e. the first date line of a date line collection, and and to continue in a forward iteration loop by the method **NextDateLine** for the date lines following. If there is no date line in the date line collection, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcDateLine	First date line

Example Code

```
Dim dateLineCltn  As VcDateLineCollection
Dim dateLine As VcDateLine

Set dateLineCltn = VcGantt1.DateLineCollection
Set dateLine = dateLineCltn.FirstDateLine
While Not dateLine Is Nothing
   ListdateLine.AddItem (dateLine.Name)
   Set dateLine = dateLineCltn.NextDateLine
Wend
```

NextDateLine

Method of VcDateLineCollection

This method can be used in a forward iteration loop to retrieve subsequent date lines from a date line collection after initializing the loop by the method **FirstDateLine**. If there is no date line left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcDateLine	Subsequent date line

Example Code

```
Dim dateLineCltn As VcDateLineCollection
Dim dateLine As VcDateLine

Set dateLineCltn = VcGantt1.DateLineCollection

Set dateLine = dateLineCltn.FirstDateLine
While Not dateLine Is Nothing
   ListdateLine.AddItem (dateLine.Name)
   Set dateLine = dateLineCltn.NextDateLine
Wend
```

Remove

Method of VcDateLineCollection

This method lets you delete a date line. To make the deletion visible in the diagram, the date line collection needs to be updated by the **Update** call.

	Data Type	Explanation
Parameter:		
⇨ dateLineName	String	Date line name
	Possible Values:	Name of the color map
Return value	Boolean	Date line deleted (True)/not deleted (False)

Example Code

```
Dim dateLineCltn As VcDateLineCollection
Dim dateLine As VcDateLine

Set dateLineCltn = VcGantt1.DateLineCollection
Set dateLine = dateLineCltn.DateLineByIndex(2)
dateLineCltn.Remove (DateLine.Name)
dateLineCltn.Update
```

Update

Method of VcDateLineCollection

This method lets you update a date line collection after having modified it.

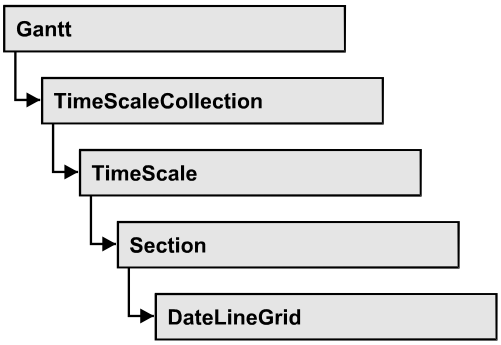
	Data Type	Explanation
Return value	Boolean	update successful (True)/ not successful (False)

Example Code

```
Dim dateLineCltn As VcDateLineCollection
Dim dateLine As VcDateLine

Set dateLineCltn = VcGantt1.DateLineCollection
Set dateLine = dateLineCltn.DateLineByIndex(2)
dateLineCltn.Remove (DateLine.Name)
```

```
dateLineCltn.Update
```

7.30 VcDateLineGrid



An object of the type **VcDateLineGrid** is a predefined grid for highlighting time periods (days, weeks, months, ...) by vertical lines.

Properties

- AdjustToReferenceDate
- AnnotationAtBottom
- AnnotationAtCenter
- AnnotationAtTop
- FormatName
- HorAlignment
- LineColor
- LineColorDataFieldIndex
- LineColorMapName
- LineThickness
- LineType
- Name
- ObserveDST
- Period
- Priority
- ReferenceDate
- SnapTarget
- TurningAnnotationEnabled
- Unit
- UseReferenceDate
- Visible
- VisibleDataFieldIndex
- VisibleMapName

Properties

AdjustToReferenceDate

Property of VcDateLineGrid

The lines of a line grid by default are positioned on the beginning of a time unit, for example on 00:00 h of a day. This property lets you position the line grid on a different value of the time unit, i.e. the one defined by the reference date, for example on 13:17 of a day. The reference date you can set by the property **set/getReferenceDate**.

This property can also be set in the **Grouping** dialog.

	Data Type	Explanation
Property value	Boolean	Date line grid positioned (True)/not positioned on reference date **Default value:** False
	Possible Values:	Group invisible/visible group nodes are/are not visible

AnnotationAtBottom

Property of VcDateLineGrid

This property lets you position the annotations of the lines in the line grid at the bottom of the Gantt graph, or retrieve whether they are there. Also see **set/getAnnotationAtCenter** and **set/getAnnotationAtTop**.

	Data Type	Explanation
Property value	Boolean	Date line grid annotations positioned at bottom (True)/not at bottom (False) **Default value:** False
	Possible Values:	Group invisible/visible group nodes are/are not visible

AnnotationAtCenter

Property of VcDateLineGrid

This property lets you position the annotations of the lines in the line grid at the center of the Gantt graph, or retrieve whether they are there. Also see **set/getAnnotationAtBottom** and **set/getAnnotationAtTop**.

	Data Type	Explanation
Property value	Boolean	Date line grid annotations positioned in the center (True) / not in the center (False) **Default value:** False
	Possible Values:	Group invisible/visible group nodes are/are not visible

AnnotationAtTop

Property of VcDateLineGrid

This property lets you position the annotations of the lines in the line grid at the top of the Gantt graph, or retrieve whether they are there. Also see **set/getAnnotationAtCenter** and **set/getAnnotationAtBottom**.

	Data Type	Explanation
Property value	Boolean	Date line grid annotations positioned at top (True)/not at top (False) **Default value:** False
	Possible Values:	Group invisible/visible group nodes are/are not visible

FormatName

Property of VcDateLineGrid

This property lets you set or retrieve the name of the line format of the date line grid.

	Data Type	Explanation
Property value	String	Name of the line format
	Possible Values:	Name of the color map

HorAlignment

Property of VcDateLineGrid

This property lets you set or retrieve the horizontal alignment of the line annotations.

	Data Type	Explanation
Property value	HorizontalAlignmentEnum **Possible Values:** vcHorCenterAligned -1 vcLeftAligned -3 vcRightAligned -2	Type of alignment horizontally centered left aligned right aligned

LineColor

Property of VcDateLineGrid

This property lets you set or retrieve the color of a date line grid.

	Data Type	Explanation
Property value	Color	RGB color values **Default value:** 255. Visual Basic: RGB (255, 0, 0)

Example Code

```
Dim dateLineGrid As VcDateLineGrid

Set dateLineGrid =
VcGantt1.TimeScaleCollection.Active.Section(0).dateLineGrid(0)

dateLineGrid.LineColor = RGB(130, 80, 200)
```

LineColorDataFieldIndex

Property of VcDateLineGrid

This property lets you set or retrieve the data field index to be used with a map specified by the property **LineColorMapName**. If you set this property to **-1**, no map will be used.

	Data Type	Explanation
Property value	Integer **Possible Values:**	Data field index

		Data field index

LineColorMapName

Property of VcDateLineGrid

This property lets you set or retrieve the name of a map for the line color. If set to "" or if the property **LineColorDataFieldIndex** is set to **-1**, then no map will be used.

	Data Type	Explanation

LineThickness

Property of VcDateLineGrid

This property lets you set or retrieve the line thickness of the grid lines.

If you set this property to values between 1 and 4, an absolute line thickness is defined in pixels. Irrespective of the zoom factor a line will always show the same line thickness in pixels. When printing though, the line thickness is adapted for the sake of legibility and becomes dependent of the zoom factor:

Value	Points	mm
1	1/2 point	0.09 mm
2	1 point	0.18 mm
3	3/2 points	0.26 mm
4	2 points	0.35 mm

A point equals 1/72 inch and represents the unit of the font size.

If you set this property to values between 5 and 1,000, the line thickness is defined in 1/100 mm, so the lines will be displayed in a true thickness in pixels that depends on the zoom factor.

This property also can be set in the **Edit Date Line Grid** dialog.

	Data Type	Explanation
Property value	Integer	Line thickness LineType {1...4}: line thickness in pixels LineType {5...1000}: line thickness in 1/100 mm **Default value:** As defined in the dialog
	Possible Values:	Data field index

Example Code

```
Dim dateLineGrid As VcDateLineGrid

Set dateLineGrid = VcGantt1.TimeScaleCollection. _
                            Active.Section(0).dateLineGrid(0)

dateLineGrid.LineThickness = 2
```

LineType

Property of VcDateLineGrid

This property lets you set or retrieve the line type of a date line grid.

	Data Type	Explanation
Property value	LineTypeEnum	Line type **Default value:** vcDashed
	Possible Values:	
	vcDashed 4	Line dashed
	vcDashedDotted 5	Line dashed-dotted
	vcDotted 3	Line dotted
	vcLineType0 100	Line Type 0
	vcLineType1 101	Line Type 1
	vcLineType10 110	Line Type 10
	vcLineType11 111	Line Type 11
	vcLineType12 112	Line Type 12
	vcLineType13 113	Line Type 13
	vcLineType14 114	Line Type 14
	vcLineType15 115	Line Type 15
	vcLineType16 116	Line Type 16
	vcLineType17 117	Line Type 17
	vcLineType18 118	Line Type 18
	vcLineType2 102	Line Type 2

vcLineType3 103	Line Type 3
vcLineType4 104	Line Type 4
vcLineType5 105	Line Type 5
vcLineType6 106	Line Type 6
vcLineType7 107	Line Type 7
vcLineType8 108	Line Type 8
vcLineType9 109	Line Type 9
vcNone 1	No line type
vcNotSet -1	No line type assigned
vcSolid 2	Line solid

Example Code

```
Dim dateLineGrid As VcDateLineGrid

Set dateLineGrid =
VcGantt1.TimeScaleCollection.Active.Section(0).dateLineGrid(0)

dateLineGrid.LineType = vcSolid
```

Name

Property of VcDateLineGrid

This property lets you set or retrieve the name of a date line grid.

	Data Type	Explanation
Property value	String	Name of the date line grid
	Possible Values:	Name of the color map

ObserveDST

Property of VcDateLineGrid

This property lets you set or retrieve whether for this line grid daylight saving time is considered or not.

	Data Type	Explanation
Property value	DateLineGridObserveDSTEnum	Daylight saving time is/is not considered.
	Possible Values:	
	vcGODDefault 9999	Default setting from .INI file is used
	vcGODNo 0	Daylight saving time is not considered

vcGODYes 1	Daylight saving time is considered

Period

Property of VcDateLineGrid

This property lets you set or retrieve after how many time units a grid line is drawn. The distance between two grid lines is given by the product of the unit (property **Unit**) and the period (property **Period**).

	Data Type	Explanation
Property value	Long	Period value **Default value:** 1

Example Code

```
Dim dateLineGrid As VcDateLineGrid

Set dateLineGrid =
VcGantt1.TimeScaleCollection.Active.Section(0).dateLineGrid(0)

dateLineGrid.Unit = vcGridUnitDay
dateLineGrid.Period = 1
```

Priority

Property of VcDateLineGrid

This property lets you specify or enquire the priority of a date line grid.

If two objects are located at the same position in the diagram, the object of higher priority is displayed in front of the objects of lower priority. By default, grids are of the lowest priority. Nodes are assigned the value 0 and thus the highest priority of all objects. By default, date line grids are displayed in front of calendar grids, but behind nodes and date lines. If you want a date line grid to be displayed in front of the nodes, you must set its priority to a positive value.

	Data Type	Explanation
Property value	Long	Priority value {-1000...+1000} **Default value:** -20

Example Code

```
Dim dateLineGrid As VcDateLineGrid
```

```
Set dateLineGrid = VcGantt1.TimeScaleCollection.Active.Section(0). _
                                    dateLineGrid(0)
dateLineGrid.Priority = 10
```

ReferenceDate

Property of VcDateLineGrid

This property lets you set or retrieve the reference date. For the date line grid to actually use the reference date, the property **UseReferenceDate** needs to be set. To adjust the date line grid to the reference date, please see property **AdjustToReferenceDate**.

The reference date shifts the beginning of the grid away from the default start on Monday 0:00 h by the offset specified. For this, the difference between the default start and the reference date is the essential part; the absolute date is not. For example: if you want the grid to start on Tuesday, you can set the reference date to May 6, 2014. You will obtain the same result by setting the reference date to April, 29, 2014. It is the difference between the date given and Monday, which is 1 day. The offset does not have to be specified in days, you can also set a day time, such as 29.4.2014 8:15 h. If you are dealing with an hour grid, only minutes are of relevance at all, so in the latter example the grid offset would be 15 minutes.

	Data Type	Explanation
Property value	Date	Reference date

SnapTarget

Property of VcDateLineGrid

This property lets you set or retrieve whether this date line grid has a snap target at the date.

	Data Type	Explanation
Property value	Boolean	Snap target is/is not defined at the date of this date line grid
	Possible Values:	Group invisible/visible group nodes are/are not visible

TurningAnnotationEnabled

Property of VcDateLineGrid

This property lets you set or retrieve whether the annotations at the lines of the date line grid can be turned by 90 degrees (vertically).

	Data Type	Explanation
Property value	Boolean	The annotations can be turned (True) / were already turned (False) **Default value:** True
	Possible Values:	Group invisible/visible group nodes are/are not visible

Unit

Property of VcDateLineGrid

This property lets you set or retrieve the unit of a date line grid. The distance between two grid lines is given by the product of unit (property **Unit**) and period (property **Period**).

	Data Type	Explanation
Property value	GridUnitEnum	Time unit **Default value:** vcGridUnitWeek
	Possible Values:	
	vcGridUnitDay 5	Grid unit **day**
	vcGridUnitHour 6	Grid unit **hour**
	vcGridUnitMinute 7	Grid unit **minute**
	vcGridUnitMonth 3	Grid unit **month**
	vcGridUnitQuarter 2	Grid unit **quarter**
	vcGridUnitSecond 8	Grid unit **second**
	vcGridUnitWeek 4	Grid unit **week**
	vcGridUnitYear 1	Grid unit **year**

Example Code

```
Dim dateLineGrid As VcDateLineGrid

Set dateLineGrid =
VcGantt1.TimeScaleCollection.Active.Section(0).DateLineGrid(0)

dateLineGrid.Period = 1
dateLineGrid.Unit = vcGridUnitDay
```

UseReferenceDate

Property of VcDateLineGrid

This property lets you set or retrieve whether the date line grid uses a reference date.

	Data Type	Explanation
Property value	Boolean	Date line grid uses (True)/does not use (False) reference date **Default value:** False
	Possible Values:	Group invisible/visible group nodes are/are not visible

Visible

Property of VcDateLineGrid

This property lets you set or retrieve whether a date line grid is visible.

	Data Type	Explanation
Property value	Boolean	Date line grid visible (True)/invisible (False) **Default value:** True
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
Dim dateLineGrid As VcDateLineGrid

Set dateLineGrid = VcGantt1.TimeScaleCollection.Active.Section(0). _
                                DateLineGrid(0)

dateLineGrid.Visible = True
```

VisibleDataFieldIndex

Property of VcDateLineGrid

This property lets you set or retrieve the index of the data field to assign a visibility mode to the calendar grid: 1 (for "visible") or 0 (for invisible). This property also can be set in the **DateLineGrid** dialog.

	Data Type	Explanation
Property value	Long	Index of the data field which contains the visibility mode

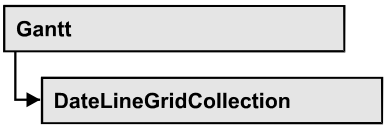
VisibleMapName

Property of VcDateLineGrid

This property lets you set or retrieve the name of a map (type vcTextMap) to set the visibility mode. If set to "", no map will be used. If a map name and additionally a data field index is specified by the property **VisibilityDataFieldIndex**, the visibility mode is selected by the map. If no data field entry applies, the date line grid will be set to "visible". This property also can be set in the **DateLineGrid** dialog.

	Data Type	Explanation
Parameter:		
⇔ Rückgabewert	String	Name of the visibility map
	Possible Values:	Name of the color map
Property value	String	Name of the map that contains the visibility mode
	Possible Values:	Name of the color map

7.31 VcDateLineGridCollection



An object of the type VcDateLineGridCollection contains all available date line grids. You can access all objects in an iterative loop by **For Each dateLineGrid In DateLineGridCollection** or by the methods **First...** and **Next...**. You can access a single date line using the methods **DateLineGridByName** and **DateLineGridByIndex**. The number of date line grids in the collection object can be retrieved by the property **Count**. The methods **Add**, **Copy** and **Remove** allow to handle the date line grids in the corresponding way.

Properties

- _NewEnum
- Count

Methods

- Add
- AddBySpecification
- Copy
- DateLineGridByIndex
- DateLineGridByName
- FirstDateLineGrid
- NextDateLineGrid
- Remove
- Update

Properties

_NewEnum

Read Only Property of VcDateLineGridCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all date line grid objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method

GetEnumerator is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim dateLineGrid As VcDateLineGrid

For Each dateLineGrid In VcGantt1.DateLineGrid
   Debug.Print dateLineGrid.Count
Next
```

Count

Read Only Property of VcDateLineGridCollection

This property lets you retrieve the number of date line grids in the DateLineGridCollection object.

	Data Type	Explanation
Property value	Long	Number of date line grids

Example Code

```
Dim  dateLineGridCltn As Vc DateLineGridCollection
Dim numberOfDateLineGrids As Long

Set dateLineGridCltn = VcGantt1.DateLineGridCollection
numberOfDateLineGrids = dateLineGridCltn.Count
```

Methods

Add

Method of VcDateLineGridCollection

This method lets you create a date line grid as a member of the DateLineGridCollection. If the name was not used before, the new date line grid object will be returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
Parameter:		
⇨ dateLineGridName	String **Possible Values:**	name of date line grid

		Name of the color map
Return value	VcDateLineGrid	New date line grid object

Example Code

```
Set newDateLineGrid = VcGantt1.DateLineGridCollection.Add("Grid1")
```

AddBySpecification

Method of VcDateLineGridCollection

This method lets you create a date line grid by using a date line grid specification. This way of creating allows date line grid objects to become persistent. The specification of a date line grid can be saved and re-loaded (see VcDateLineGrid property **Specification**). In a subsequent session the date line grid can be created again from the specification and is identified by its name.

	Data Type	**Explanation**
Parameter:		
⇨ specification	String **Possible Values:**	date line grid specification Name of the color map
Return value	VcDateLineGrid	New date line grid object

Copy

Method of VcDateLineGridCollection

By this method you can copy a date line grid. If the date line grid that is to be copied exists, and if the name for the new date line grid does not yet exist, the new date line grid object is returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	**Explanation**
Parameter:		
⇨ dateLineGridName	String **Possible Values:**	Name of the date line grid to be copied Name of the color map
⇨ newDateLineGridName	String **Possible Values:**	Name of the new date line grid Name of the color map

Return value	VcDateLineGrid	date line grid object

Example Code

```
Dim dateLineGridCltn As VcDateLineGridCollection
Dim dateLineGrid As VcDateLineGrid

Set dateLineGridCltn = VcGantt1.DateLineGridCollection
Set dateLineGrid = dateLineGridCltn.Copy("CurrentDateLineGrid",
"NewDateLineGrid")
```

DateLineGridByIndex

Method of VcDateLineGridCollection

This method lets you access a date line grid by its index. If a date line grid of the specified index does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter:		
⇨ index	Integer	Index of the date line grid
	Possible Values:	Data field index
Return value	VcDateLineGrid	date line grid object returned

Example Code

```
Dim dateLineGridCltn As VcDateLineGrid
Dim dateLine As VcDateLine

Set dateLineGridCltn = VcGantt1.DateLineGrid
Set dateLineGrid = dateLineGridCltn.DateLineGridByIndex(2)
MsgBox dateLineGrid.Name
```

DateLineGridByName

Method of VcDateLineGridCollection

This method is used to access a date line grid by its name. If a date line grid of the specified name does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter:		
⇨ DateLineGridName	String	Name of the date line grid
	Possible Values:	Name of the color map

Return value	VcDateLineGrid	date line grid

Example Code

```
Dim dateLineGridCltn As VcDateLineGridCollection
Dim dateLineGrid As VcDateLineGrid

Set dateLineGridCltn = VcGantt1.DateLineGridCollection
Set dateLineGrid = dateLineGridrCltn.DateLineGridByName("Grid 4")
```

FirstDateLineGrid

Method of VcDateLineGridCollection

This method can be used to access the initial value, i.e. the first date line grid of a date line grid collection and then to continue in a forward iteration loop by the method **NextDateLineGrid** for the date line grids following. If there is no date line grid in the DateLineGridCollection, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcDateLineGrid	First date line grid

Example Code

```
Dim dateLineGridCltn As VcDateLineGridCollection
Dim dateLineGrid As VcDateLineGrid

Set dateLineGridCltn = VcGantt1.DateLineGridCollection
dateLineGridCltn.DateLineGrids (vcAnyDateLineGrid)
Set dateLineGrid = dateLineGridCltn.FirstDateLineGrid
```

NextDateLineGrid

Method of VcDateLineGridCollection

This method can be used in a forward iteration loop to retrieve subsequent date line grids from a DateLineGridCollection after initializing the loop by the method **FirstDateLineGrid**. If there is no date line grid left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcDateLineGrid	Subsequent date line grid

Example Code

```
Dim dateLineGridCltn As VcDateLineGridCollection
Dim dateLineGrid As VcDateLineGrid

Set dateLineGridCltn = VcGantt1.DateLineGridCollection
Set dateLineGrid = dateLineGridrCltn.FirstDateLineGrid
```

```
While Not dateLineGrid Is Nothing
   Listbox.AddItem dateLineGrid.Name
   Set dateLineGrid = dateLineGridCltn.NextDateLineGrid
Wend
```

Remove

Method of VcDateLineGridCollection

This method lets you delete a date line grid. If the date line grid is used in another object, it cannot be deleted. Then False will be returned, otherwise True.

	Data Type	Explanation
Parameter:		
⇨ dateLineGridName	String	date line grid name
	Possible Values:	Name of the color map
Return value	Boolean	date line grid deleted (True)/not deleted (False)

Example Code

```
Dim dateLineGridCltn As VcDateLineGridCollection
Dim dateLineGrid As VcDateLineGrid

Set dateLineGridCltn = VcGantt1.DateLineGridCollection
Set dateLineGrid = dateLineGridCltn.FormatByIndex(1)
dateLineGridCltn.Remove (dateLineGrid.Name)
```

Update

Method of VcDateLineGridCollection

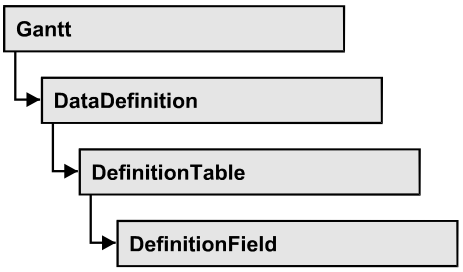
This method has to be used when date line grid modifications have been carried out. The method **Update** updates all objects that are concerned by the date line grid you have edited. You should call this method at the end of the code that defines the date line grids and the date line grid collection. Otherwise the update will be processed before all date line grid definitions are processed.

	Data Type	Explanation
Return value	Boolean	update successful (True)/ not successful (False)

Example Code

```
Dim dateLineGrid As VcDateLineGrid

Set dateLineGrid = VcGantt1.DateLineGrid.Collection.DateLineGridByName("Grid 3")
dateLineGrid.Update
```

7.32 VcDefinitionField



An object of the type VcDefinitionField defines a field of the data definition table. The definition basically consists of a name and a data type.

Properties

- DateFormat
- Editable
- Hidden
- ID
- Name
- Type

Properties

DateFormat

Property of VcDefinitionField

This property lets you set or retrieve the date format of the field of a data definition table. This property only works if the data type of the field was set to **vcDataTableFieldDateTime**. The dateFormat setting is used when reading or storing CSV files and when the format type **String** is used when adding a data record by the methods **InsertNodeRecord** or **InsertLink-Record** of the VcGantt object. The format of the date output in the chart is controlled by the VcGantt property **DateOutputFormat**.

Note: You should set the property Type first before setting the property DateFormat.

	Data Type	Explanation
Property value	String	Date format {DMYhms:;./} **Default value:** bei vcDefFieldDateTime DD.MM.YYYY hh:mm:ss
	Possible Values:	Name of the color map

Example Code

```
Dim dataDefTable As VcDataDefinitionTable
Dim dataDefField As VcDefinitionField

Set dataDefTable = VcGantt1.DataDefinition.DefinitionTable(vcMaindata)
Set dataDefField = dataDefTable.FieldByName("Start")
dataDefField.Type = vcDefFieldDateTimeType
'DateFormat = "DD.MM.YYYY"
dataDefField.DateFormat = "01.12.2014"
```

Editable

Property of VcDefinitionField

This property lets you set or retrieve whether the data field should be editable at run time in the chart table and in the dialog **EditNode**.

	Data Type	Explanation
Property value	Boolean	Definition field editable/not editable **Default value:** True
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
Dim dataDefTable As VcDataDefinitionTable
Dim dataDefField As VcDefinitionField

Set dataDefTable = VcGantt1.DataDefinition.DefinitionTable(vcMaindata)
Set dataDefField = dataDefTable.FieldByName("Start")
dataDefField.Editable = False
```

Hidden

Property of VcDefinitionField

This property lets you require/set whether a data field is hidden at run time.

	Data Type	Explanation
Property value	Boolean	Definition field hidden/not hidden **Default value:** False
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
Dim dataDefTable As VcDataDefinitionTable
Dim dataDefField As VcDefinitionField

Set dataDefTable = VcGantt1.DataDefinition.DefinitionTable(vcMaindata)
Set dataDefField = dataDefTable.FieldByName("Start")
dataDefField.Hidden = True
```

ID

Read Only Property of VcDefinitionField

This property lets you retrieve the index of the field of a data definition table.

	Data Type	Explanation
Property value	Integer	Index of the definition field
	Possible Values:	Data field index

Example Code

```
Dim dataDefTable As VcDataDefinitionTable
Dim dataDefField As VcDefinitionField

Set dataDefTable = VcGantt1.DataDefinition.DefinitionTable(vcMaindata)
Set dataDefField = dataDefTable.FieldByName("Start")
MsgBox dataDefField.ID
```

Name

Property of VcDefinitionField

This property lets you set or retrieve the name of the field of a data definition table.

	Data Type	Explanation
Property value	String	Name of the definition field
	Possible Values:	Name of the color map

Example Code

```
Dim dataDefTable As VcDataDefinitionTable
Dim dataDefField As VcDefinitionField

Set dataDefTable = VcGantt1.DataDefinition.DefinitionTable(vcMaindata)
Set dataDefField = dataDefTable.CreateDataField("Start")
```

Type

Property of VcDefinitionField

This property lets you set or retrieve the type of the field of a data definition table.

Note: By setting the property **Type** the property **DateFormat** will change!

vcDefFieldAlphanumericType: DateFormat = ""

vcDefFieldDateTimeType: DateFormat = "DD.MM.YYYY hh:mm:ss"

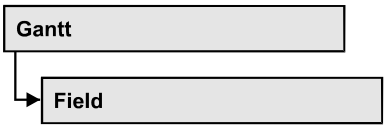
vcDefFieldIntegerType: DateFormat = ""

	Data Type	Explanation
Property value	DefinitionFieldTypeEnum	type of the definition field **Default value:** vcDefFieldIntegerType
	Possible Values: vcDefFieldAlphanumericType 1 vcDefFieldDateTimeType 4 vcDefFieldIntegerType 2	 Data type **alphanumeric**: "" Data type **date** : DD.MM.YYYY Data type **integer** (32 bits): ""

Example Code

```
Dim dataDefTable As VcDataDefinitionTable
Dim dataDefField As VcDefinitionField

Set dataDefTable = VcGantt1.DataDefinition.DefinitionTable(vcMaindata)
Set dataDefField = dataDefTable.CreateDataField("Start")
dataDefField.Type = vcDefFieldDateTimeType
```

7.33 VcField



An object of the type VcField represents a field in a data record. A field can be referred to by its ID.

Properties

- DataFieldID

Properties

DataFieldID

Read Only Property of VcField

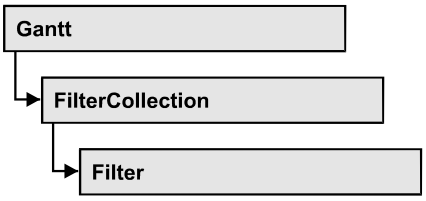
This property lets you retrieve the ID of a data field.

	Data Type	Explanation
Property value	Integer	Data field ID
	Possible Values:	Data field index

Example Code

```
Private Sub VcGantt1_OnNodeLClick(ByVal node As VcGanttLib.VcNode, _
            ByVal location As VcGanttLib.LocationEnum, ByVal x As Long, _
            ByVal y As Long, returnStatus As Variant)

    Dim identifiedObject As Object
    Dim identifiedObjectType As VcObjectTypeEnum
    Dim dataField As VcField

    If location = vcInTable Then
        Call VcGantt1.IdentifyObjectAt(x, y, identifiedObject,  _
                                       identifiedObjectType)
        Set dataField = VcGantt1.IdentifyField(x, y, identifiedObjectType)
        MsgBox dataField.DataFieldID
    End If

End Sub
```

7.34 VcFilter



An object of the type VcFilter contains subconditions (VcFilterSubCondition), e.g. permitted values to be compared to the data fields of a node or a link, so that the filter conditions may or may not apply to an object. Filters are used p.e. to assign a format to an activity. Only if the filter is valid after the subconditions have been modified, the modified subconditions will become valid. Otherwise the former filter subconditions will remain valid. This can be controlled via the methods VcFilter.IsValid and VcFilterSubCondition.IsValid.

Properties

- _NewEnum
- DataDefinitionTable
- DatesWithHourAndMinute
- Name
- Specification
- StringsCaseSensitive
- SubCondition
- SubConditionCount

Methods

- AddSubCondition
- CopySubCondition
- Evaluate
- IsValid
- RemoveSubCondition

Properties

_NewEnum

Read Only Property of VcFilter

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all filter condition objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method **GetEnumerator** is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim fiSuCo As VcFilterSubCondition

For Each fiSuCo In filter
    Debug.Print fiSuCo.Index
Next
```

DataDefinitionTable

Property of VcFilter

This property lets you enquire whether the filter is a filter for nodes (vcMainData) or for links (vcRelations). This property can be modified only if the filter does not contain subconditions.

	Data Type	Explanation
Property value	DataTableEnum	Type of data definition table
	Possible Values:	
	vcMaindata 0	Table type **vcMaindata** (for nodes)
	vcRelations 1	Table type **vcRelations** (for links)

DatesWithHourAndMinute

Property of VcFilter

This property lets you enquire/set whether the comparison of subconditions that contain dates checks the information on hours and minutes. The setting

can only be modified when there is at least one subcondition containing a date comparison. Otherwise the property value is always False.

	Data Type	Explanation
Property value	Boolean	hours and minutes are compared (True)/ not compared (False)
	Possible Values:	Group invisible/visible group nodes are/are not visible

Name

Property of VcFilter

This property lets you enquire/set the name of the filter.

	Data Type	Explanation
Property value	String	Name of the filter
	Possible Values:	Name of the color map

Example Code

```
Dim filterCltn As VcFilterCollection
Dim filter As VcFilter

Set filterCltn = VcGantt1.FilterCollection

For Each filter In filterCltn
    ListBox.AddItem filter.name
Next filter
```

Specification

Read Only Property of VcFilter

This property lets you retrieve the specification of a filter. A specification is a string that contains legible ASCII characters from 32 to 127 only, so it can be stored without problems to text files or data bases. This allows for persistency. A specification can be used to create a filter by the method **VcFilterCollection.AddBySpecification**.

	Data Type	Explanation
Property value	String	Specification of the filter
	Possible Values:	Name of the color map

StringsCaseSensitive

Property of VcFilter

This property lets you enquire/set whether subconditions that contain strings are case-sensitive.

	Data Type	Explanation
Property value	Boolean	case-sensitive (True)/not case-sensitive (False)
	Possible Values:	Group invisible/visible group nodes are/are not visible

SubCondition

Property of VcFilter

This property lets you access a VcFilterSubCondition object by its index.

	Data Type	Explanation
Parameter:		
⇨ index	Integer	index of the filter subcondition {0 ... VcFilter.SubConditionCount-1}
	Possible Values:	Data field index
Property value	VcFilterSubCondition	filter subcondition object

SubConditionCount

Read Only Property of VcFilter

This property lets you enquire the number of filter subconditions.

	Data Type	Explanation
Property value	Integer	number of filter subconditions
	Possible Values:	Data field index

Methods

AddSubCondition

Method of VcFilter

This method lets you create a new filter condition in the collection of the filter conditions. Its position is specified by the index. The corresponding VcFilterSubCondition object will be returned.

Default properties of this object:

- DataFieldIndex: -1
- Operator: vcInvalidOp
- ComparisonValueAsString: "<INVALID>"
- ConnectionOperator: vcInvalidConnOp.

	Data Type	Explanation
Parameter:		
⇨ atIndex	Integer	Index of the new filter subcondition {0 to VcFilter.SubConditionCount and –1 for "at the end of the Collection" (identical with the value VcFilter.SubConditionCount)}
	Possible Values:	Data field index
Return value	VcFilterSubCondition	Filter subcondition object

CopySubCondition

Method of VcFilter

This method lets you copy a filter subcondition by its index. The new filter subcondition will be inserted into the collection at the position specified by the index. It will be returned as a VcFilterSubCondition object.

	Data Type	Explanation
Parameter:		
⇨ fromIndex	Integer	Index of the filter subcondition to be copied

	Possible Values:	Data field index
⇨ atIndex	Integer	Index of the new filter subcondition {0 to VcFilter.SubConditionCount and –1 for "at the end of the Collection" (identical with the value VcFilter.SubConditionCount)}
	Possible Values:	Data field index
Return value	VcFilterSubCondition	Filter subcondition object

Evaluate

Method of VcFilter

This methods lets you check whether the specified filter applies for a certain data record or not. You should only pass objects that are internally linked with data records of the data tables. Those are **VcNode, VcLink, VcGroup, VcDataRecord**. If an object is passed that is not listed, an exception will be triggered.

	Data Type	**Explanation**
Parameter:		
⇨ dataObjectParam	Variant	Data record object
Return value	Boolean	Filter applies for data record (True)/does not apply (False)

IsValid

Method of VcFilter

This property checks whether all filter subconditions are correct. The correctness of all subconditions is the condition that changed filter subconditions become valid. Otherwise the former subconditons will remain valid.

	Data Type	**Explanation**
Return value	Boolean	Filter subconditions correct (True)/ not correct (False)

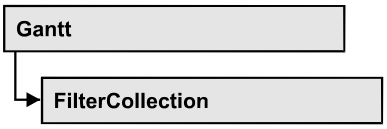
RemoveSubCondition

Method of VcFilter

This method lets you delete a filter subcondition by its index.

	Data Type	Explanation
Parameter:		
⇨ index	Integer	index of the filter subcondition to be removed
	Possible Values:	Data field index

7.35 VcFilterCollection



An object of the type VcFilterCollection automatically contains all available filters. You can access all objects in an iterative loop by **For Each filter In FilterCollection** or by the methods **First...** and **Next...**. You can access a single filter using the methods **FilterByName** and **FilterByIndex**. The number of filters in the collection object can be retrieved by the property **Count**. The methods **Add**, **Copy** and **Remove** allow to handle the filters in the corresponding way.

Properties

- _NewEnum
- Count
- MarkedNodesFilter

Methods

- Add
- AddBySpecification
- Copy
- FilterByIndex
- FilterByName
- FirstFilter
- NextFilter
- Remove

Properties

_NewEnum

Read Only Property of VcFilterCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all filter objects contained. In Visual Basic this property never is displayed, but it can be addressed by the command **For Each** *element* **In** *collection*. In .NET

languages the method GetEnumerator is offered instead. Some development environments replace this property by own language constructs.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim filter As VcFilter

For Each filter In VcGantt1.FilterCollection
   Debug.Print filter.Name
Next
```

Count

Read Only Property of VcFilterCollection

This property lets you retrieve the number of filters in the filter collection.

	Data Type	Explanation
Property value	Long	Number of filters

Example Code

```
Dim filterCltn As VcFilterCollection
Dim numberOfFilters As Long

Set filterCltn = VcGantt1.FilterCollection
numberOfFilters = filterCltn.Count
```

MarkedNodesFilter

Read Only Property of VcFilterCollection

This property lets you retrieve a constant pseudo-filter that can be used only for **ActiveNodeFilter** for filtering the nodes currently marked (sub-diagram).

	Data Type	Explanation
Property value	VcFilter	Pseudo filter

Example Code

```
Set VcGantt1.ActiveNodeFilter = VcGantt1.FilterCollection.MarkedNodesFilter
```

Methods

Add

Method of VcFilterCollection

By this method you can create a filter as a member of the FilterCollection. If the name has not been used before, the new filter object will be returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

The new filter automatically refers to the data definition table vcMainData (see VcFilter.DataDefinitionTable). You can select vcRelations instead, as long as the filter does not contain any subconditions.

	Data Type	Explanation
Parameter: ⇨ newName	String **Possible Values:**	Filter name Name of the color map
Return value	VcFilter	New filter object

Example Code

```
Set newFilter = VcGantt1.FilterCollection.Add("foo")
```

AddBySpecification

Method of VcFilterCollection

This method lets you create a filter by using filter specification. This way of creating allows filter objects to become persistent. The specification of a filter can be saved and re-loaded (see VcFilter property **Specification**). In a subsequent the filter can be created again from the specification and is identified by its name.

	Data Type	Explanation
Parameter: ⇨ filterSpecification	String **Possible Values:**	Filter specification Name of the color map
Return value	VcFilter	New filter object

Copy

Method of VcFilterCollection

By this method you can copy a filter. If the filter that is to be copied exists, and if the name for the new filter does not yet exist, the new filter object is returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
Parameter:		
⇨ fromName	String **Possible Values:**	Name of the filter to be copied Name of the color map
⇨ newName	String **Possible Values:**	Name of the new filter Name of the color map
Return value	VcFilter	Filter object

FilterByIndex

Method of VcFilterCollection

This method lets you access a filter by its index. If a filter of the specified index does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter:		
⇨ index	Integer **Possible Values:**	Index of the filter Data field index
Return value	VcFilter	Filter object returned

FilterByName

Method of VcFilterCollection

By this method you can retrieve a filter by its name. If a filter of the specified name does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter:		
⇨ filterName	String	Filter name
	Possible Values:	Name of the color map
Return value	VcFilter	Filter

Example Code

```
Dim filterCltn As VcFilterCollection
Dim filter As VcFilter

Set filterCltn = VcGantt1.FilterCollection
Set filter = filterCltn.FilterByName("Department A")
```

FirstFilter

Method of VcFilterCollection

This method can be used to access the initial value, i.e. the first filter of a filter collection, and then to continue in a forward iteration loop by the method **NextFilter** for the filters following. If there is no filter in the FilterCollection object, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcFilter	First filter

Example Code

```
Dim filterCltn As VcFilterCollection
Dim filter As VcFilter

Set filterCltn = VcGantt1.FilterCollection
Set filter = filterCltn.FirstFilter
```

NextFilter

Method of VcFilterCollection

This method can be used in a forward iteration loop to retrieve subsequent filters from a curve collection after initializing the loop by the method **FirstFilter**. If there is no filter left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcFilter	Subsequent filter

Example Code

```
Dim filterCltn As VcFilterCollection
Dim filter As VcFilter

Set filterCltn = VcGantt1.FilterCollection
Set filter = filterCltn.FirstFilter

While Not filter Is Nothing
    Listbox.AddItem filter.Name
    Set filter = filterCltn.NextFilter
Wend
```

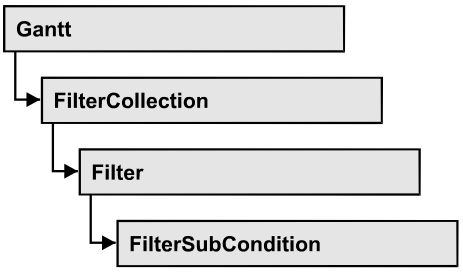
Remove

Method of VcFilterCollection

This method lets you delete a filter. If the filter is used in another object, it cannot be deleted. Then False will be returned, otherwise True.

	Data Type	Explanation
Parameter:		
⇨ name	String **Possible Values:**	Filter name Name of the color map
Return value	Boolean	Filter deleted (True)/not deleted (False)

7.36 VcFilterSubCondition



An object of the type VcFilterSubCondition contains a single filter subcondition. It does not have a name, but only an index that specifies its position in the filter.

In the **Edit Filter** dialog each line corresponds to a subcondition. The properties specified at design time in that dialog can be modified via the API at runtime.

Properties

- ComparisonValueAsString
- ConnectionOperator
- DataFieldIndex
- FilterName
- Index
- Operator

Methods

- IsValid

Properties

ComparisonValueAsString

Property of VcFilterSubCondition

This property lets you enquire/set the comparison value. This string must have the following format:

- String: included by double quotation marks. Example in VB: """Aachen"""; Example in C/C++: "\"Aachen\""

- Date: included by # signs. Example: "#18.06.2015; 12:34:56;#" (as this is the control's default format that is independent of the operating system and its local settings the date format is always "DD.MM.YYYY;hh:mm:ss;". A special date comparison value is "<TODAY>".

- Date field: included by square brackets. Example: "[ID]"

- Number: entered directly. Example: "52076"

- List: for a vc...In operator: included by {} brackets. All values included must have the same type (string, date or number). They may have one of the formats mentionned above. Example: "{"NETRONIC", [Name]}"

- Invalid (e.g. after creating a subcondition): "<INVALID>"

The type of the comparison value has to match the type of the data field and the operator type.

	Data Type	Explanation
Property value	String **Possible Values:**	Comparison value Name of the color map

ConnectionOperator

Property of VcFilterSubCondition

This property lets you enquire/set the operator for the connetion with the following subcondition. **vcAnd** binds stronger than **vcOr**.

	Data Type	Explanation
Property value	ConnectionOperatorEnum **Possible Values:** vcAnd 1 vcInvalidConnOp 0 vcOr 2	Operator to connect to the subsequent condition And operator invalid operator Or operator

DataFieldIndex

Property of VcFilterSubCondition

This property lets you set or retrieve the index of the data field the content of which is to be compared. The data field type has to match the type of the comparison value and the operator.

Special values:

- -1: no data field (invalid)
- vcBarGroupLevel: variable for the group level number
- vcGroupCollapsed: entry for collapsed groups
- vcGroupNodeOrSummaryNode: entry for summary bars
- vcNodesInSeparateRows: entry for displaying all nodes in separate rows
- vcNodesOverlaid: entry for displaying nodes overlaid, if necessary
- vcRowNumber: entry to define filters for special rows
- vcSumBarLevel: variable for the level number of the summary bar

This property can also bet set in the **Edit filter** dialog.

	Data Type	Explanation
Property value	Long	Index of the data field to be compared

FilterName

Read Only Property of VcFilterSubCondition

This property lets you enquire the name of the filter to which this subcondition belongs to.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the filter Name of the color map

Index

Read Only Property of VcFilterSubCondition

This property lets you enquire the index of this subcondition in the corresponding filter.

	Data Type	Explanation
Property value	Integer	Index of the subcondition in the filter
	Possible Values:	
		Data field index

Operator

Property of VcFilterSubCondition

This property lets you set or retreive the comparison operator. The operators that are available in the API correspond to the operators in the **Edit Filter** dialog. The operator type has to match the types of the data field and of the comparison value.

	Data Type	Explanation
Property value	OperatorEnum	comparison operator
	Possible Values:	
	vcDateEarlier 27	date earlier than
	vcDateEarlierOrEqual 28	date earlier than or equal
	vcDateEqual 25	date equal
	vcDateIn 31	date in
	vcDateLater 29	date later than
	vcDateLaterOrEqual 30	date later than or equal
	vcDateNotEqual 26	date not equal
	vcDateNotIn 32	date not in
	vcIntEqual 9	integer equal
	vcIntGreater 13	integer greater
	vcIntGreaterOrEqual 14	integer greater or equal
	vcIntIn 15	integer in
	vcIntLess 11	integer smaller than
	vcIntLessOrEqual 12	integer smaller than or equal
	vcIntNotEqual 10	integer not equal
	vcIntNotIn 16	integer not in
	vcInvalidOp 0	invalid operator
	vcStringBeginsWith 3	string begins with
	vcStringContains 5	string contains
	vcStringEqual 1	string equal
	vcStringIn 7	string contains
	vcStringNotBeginsWith 4	string does not begin with
	vcStringNotContains 6	string does not contain
	vcStringNotEqual 2	string is not equal

vcStringNotIn 8	string is not in

Methods

IsValid

Method of VcFilterSubCondition

This property checks whether the filter subcondition is correct.

	Data Type	Explanation
Return value	Boolean	Filter subcondition correct (True)/ not correct (False)

7.37 VcGantt

Gantt

A VcGantt object is the VARCHART XGantt control. You use events to control interactions with the VcGantt object. It can be customized by a number of properties and methods to meet your demands.

Properties

- ActiveNodeFilter
- AllowMultipleBoxMarking
- AllowNewBoxes
- AllowNewNodes
- AllowNumericScaleRescale
- AllowPanningMode
- AllowSelectionViaRubberRect
- AllowTableColumnWidthOptimization
- AllowTimescaleRescale
- AllowVerticalNodeMovement
- AllowVerticalNodeMovementViaTable
- Arrangement
- ArrowKeyMode
- ArrowKeyStepSizeMultiplier
- AssignCalendarToNodes
- BarSeparationGroupBy
- BorderArea
- BoxCollection
- BoxFormatCollection
- CalendarCollection
- CalendarGridCollection
- CalendarProfileCollection
- ConfigurationName
- ConsiderLinkRelationTypesOnNodeDragging
- ContextMenuForBoxesEnabled
- CtrlCXVProcessing
- CurrentVersion
- DataDefinition
- DataTableCollection
- DateLineCollection
- DateLineGridCollection

- DateOutputFormat
- DiagramAlternatingRowBackColor
- DiagramBackColor
- DiagramHistogramHeightRatio
- DiagramHistogramHeightRatioEx
- DiagramVisible
- DialogFont
- DirectDataWritingModeEnabled
- DoubleOutputFormat
- EditNewNode
- Enabled
- EnableSupplyTextEntryEvent
- EventReturnStatus
- EventsSecurityCheck
- EventText
- ExtendedDataTables
- ExtendedEditingBehavior
- FilePath
- FilterCollection
- FontAntiAliasingEnabled
- GroupCollection
- GroupingField
- GroupingModificationsAllowed
- GroupingOrderField
- GroupingSortOrder
- GroupLevelLayoutCollection
- GroupOptimizationOnInteractionsEnabled
- HierarchyDataFieldIndex
- HierarchyLevelLayout
- HistogramCollection
- HistogramSeparationLineColor
- hWnd
- InfoWindow
- InInteractionEventsEnabled
- InPlaceEditingOnGroupsInDiagramEnabled
- InPlaceEditingOnGroupsInTableEnabled
- InPlaceEditingOnNodesInDiagramEnabled
- InPlaceEditingOnNodesInTableEnabled
- InteractionMode
- LayerCollection

- LegendView
- LineFormatCollection
- LinkAppearanceCollection
- LinkCollection
- LinkPredecessorDataFieldIndex
- LinksDataTableName
- LinkSuccessorDataFieldIndex
- LinkTypeDataFieldIndex
- MapCollection
- MinimumRowHeight
- MouseProcessingEnabled
- MoveAllSelectedNodes
- MoveLayersAsNodeWithShiftKey
- MoveNodeAlways
- MoveNodeWhenMarked
- NewNodesViaDoubleClick
- NodeCalendarNameDataFieldIndex
- NodeCollection
- NodeDurationDataFieldIndex
- NodeEndDateDataFieldIndex
- NodeLevelLayout
- NodeRowNumberDataFieldIndex
- NodesDataTableName
- NodeStartDateDataFieldIndex
- NodeTooltipTextField
- NoOfInitialRows
- OLEDragHorizontalMovementAllowed
- OLEDragMode
- OLEDragViaDiagram
- OLEDragViaTable
- OLEDragWithOwnMouseCursor
- OLEDragWithPhantom
- OLEDropMode
- OverlapLayerEnabled
- OverlapLayerName
- PartialLoadThreshold
- PhantomLayerHeight
- Printer
- ResourceScheduler2
- RightTable

- RightTableDiagramWidthRatio
- RightTableDiagramWidthRatioEx
- RoundedLinkSlantsEnabled
- RowHeightReductionEnabled
- RowMargins
- Sash3DStyleEnabled
- SashThickness
- Scheduler
- ScrollEventsEnabled
- SelectedRowBackColorAsARGB
- ShowNonWorkInterval
- ShowSnapLines
- ShowSnapMarkings
- ShowTimeScaleDialog
- ShowToolTip
- SnapTargetNodesSelectionMode
- SortField
- SortOrder
- SubRowMargins
- SummaryBarsVisible
- Table
- TableCollection
- TableDiagramWidthRatio
- TableDiagramWidthRatioEx
- TimeScaleCollection
- TimeScaleEnd
- TimeScaleStart
- TimeUnit
- TimeUnitsPerStep
- ToolTipChangeDuration
- ToolTipDuration
- ToolTipPointerDuration
- ToolTipShowAfterClick
- TrackingSpaceBackColorAsARGB
- TrackingSpacePattern
- TrackingSpacePatternColorAsARGB
- UpdateBehaviorCollection
- UseHigherDiagramHistogramHeightRatioPrecision
- UseHigherTableDiagramWidthRatioPrecision
- UseSnapTargetsInInteractions

- UseTwinLineSashPhantom
- ViewComponentsBackColor
- ViewComponentsBorderColor
- WaitCursorEnabled
- WorldView
- ZoomFactor
- ZoomingPerMouseWheelAllowed

Methods

- AboutBox
- Clear
- ClearAll
- ConvertDistance
- DeleteLinkRecord
- DeleteNodeRecord
- DetectDataTableFieldName
- DetectDataTableName
- DetectFieldIndex
- DumpConfiguration
- EditGroup
- EditLink
- EditNode
- EndLoading
- ExportGraphicsToFile
- FitChartIntoView
- FitHistogramsIntoView
- FitRangeIntoView
- GetAValueFromARGB
- GetBValueFromARGB
- GetCurrentComponentStart
- GetCurrentViewDates
- GetCurrentViewDatesAsString
- GetCurrentViewDatesAsVariant
- GetDate
- GetDateAsString
- GetGValueFromARGB
- GetLinkByID
- GetLinkByIDs
- GetNodeByID
- GetRValueFromARGB

- GetViewComponentSize
- GetViewComponentSizeAsVariant
- GroupNodes
- HistogramSetMaxYValue
- IdentifyField
- IdentifyLayerAt
- IdentifyLayerAtAsVariant
- IdentifyObject
- IdentifyObjectAt
- IdentifyObjectAtAsVariant
- InsertLinkRecord
- InsertNodeRecord
- MakeARGB
- Open
- OptimizeTimeScaleStartEnd
- PageLayout
- PrintDirectEx
- PrinterSetup
- PrintIt
- PrintPreview
- PrintToFile
- RecalculateAllStructureCodes
- Reset
- SaveAsEx
- Schedule
- ScrollComponentStartTo
- ScrollToDate
- ScrollToGroupLine
- ScrollToNode
- ScrollToNodeLine
- ShowExportGraphicsDialog
- SortGroups
- SortNodes
- SuspendUpdate
- UpdateLinkRecord
- UpdateNodeRecord
- UpdateRowNumberFields
- Zoom

Events

- Error
- ErrorAsVariant
- KeyDown
- KeyPress
- KeyUp
- OLECompleteDrag
- OLEDragDrop
- OLEDragOver
- OLEGiveFeedback
- OLESetData
- OLEStartDrag
- OnBoxCreate
- OnBoxCreateComplete
- OnBoxLClick
- OnBoxLDblClick
- OnBoxModify
- OnBoxModifyCompleteEx
- OnBoxRClick
- OnCalendarGridRClick
- OnCurveLClick
- OnCurveLDblClick
- OnCurveModifyComplete
- OnCurveModifyEx
- OnCurveModifyEx2
- OnCurveModifyExAsString
- OnCurveRClick
- OnDataRecordCreate
- OnDataRecordCreateComplete
- OnDataRecordDelete
- OnDataRecordDeleteComplete
- OnDataRecordModify
- OnDataRecordModifyComplete
- OnDataRecordNotFound
- OnDateLineModify
- OnDateLineRClick
- OnDeleteCurvePoint
- OnDeleteCurvePointEx
- OnDiagramLClick
- OnDiagramLDblClick
- OnDiagramRClick

- OnGroupDelete
- OnGroupLClick
- OnGroupLDblClick
- OnGroupModify
- OnGroupModifyComplete
- OnGroupModifyEx
- OnGroupRClick
- OnGroupsMark
- OnGroupsMarkComplete
- OnHelpRequested
- OnHistogramLClick
- OnHistogramLDblClick
- OnHistogramRClick
- OnHistogramsHeight
- OnHistogramsHeightChanged
- OnHistogramsHeightModifyEx
- OnInsertCurvePoint
- OnInsertCurvePointEx
- OnInteractionEndComplete
- OnInteractionModeChange
- OnInteractionModeChangeComplete
- OnInteractionObjectChangingComplete
- OnInteractionStartComplete
- OnLegendViewClosed
- OnLinkCreate
- OnLinkCreateComplete
- OnLinkDelete
- OnLinkDeleteComplete
- OnLinkLClickCltn
- OnLinkLDblClickCltn
- OnLinkRClickCltn
- OnModifyComplete
- OnMouseDblClk
- OnMouseDown
- OnMouseMove
- OnMouseUp
- OnNodeCreate
- OnNodeCreateCompleteEx
- OnNodeDelete
- OnNodeDeleteCompleteEx

- OnNodeLClick
- OnNodeLDblClick
- OnNodeModifyComplete
- OnNodeModifyCompleteEx
- OnNodeModifyEx
- OnNodeRClick
- OnNodeResizeStart
- OnNodesMarkComplete
- OnNodesMarkEx
- OnNumericScaleLClick
- OnNumericScaleLDblClick
- OnNumericScaleRClick
- OnNumericScaleRescale
- OnObjectDrawCompleteEx
- OnObjectDrawEx
- OnOptimizeTableColumnWidth
- OnPreScrollComponent
- OnPreScrollDiagramHor
- OnResourceSchedulingProgress
- OnResourceSchedulingWarning
- OnScrollComponent
- OnScrollDiagramHor
- OnSelectField
- OnShowCurveNameInMenu
- OnShowDate
- OnShowInPlaceEditor
- OnStatusLineText
- OnSupplyTextEntry
- OnSupplyTextEntryAsVariant
- OnTableCaptionLClick
- OnTableCaptionLDblClick
- OnTableCaptionRClick
- OnTableColumnWidth
- OnTableColumnWidthModifyComplete
- OnTableWidth
- OnTableWidthModifyEx
- OnTimeScaleChangeComplete
- OnTimeScaleEndModifyComplete
- OnTimeScaleLClick
- OnTimeScaleLDblClick

- OnTimeScaleRClick
- OnTimeScaleSectionRescaleCompleteEx
- OnTimeScaleSectionRescaleEx
- OnTimeScaleSectionStartModify
- OnTimeScaleStartModifyComplete
- OnToolTipText
- OnToolTipTextAsVariant
- OnViewComponentsSizeModifyComplete
- OnWorldViewClosed
- OnZoomFactorModifyComplete

Properties

ActiveNodeFilter

Property of VcGantt

This property lets you set or retrieve a filter that selects the nodes to be displayed.

	Data Type	Explanation
Property value	VcFilter	Filter object **Default value:** Nothing

Example Code

```
Set VcGantt1.ActiveNodeFilter = VcGantt1.FilterCollection. _
                    FilterByName("Milestone")
```

AllowMultipleBoxMarking

Property of VcGantt

This property lets you specify or retrieve whether at run time several boxes can be marked simultaneously. If the property is not activated, the user has to keep the CTRL key pressed in order to mark several boxes. You can also set this property on the **General** property page

	Data Type	Explanation
Property value	Boolean	Multiple box marking enabled / not enabled **Default value:** True
	Possible Values:	Group invisible/visible

		group nodes are/are not visible

Example Code

```
VcGantt1.AllowMultipleBoxMarking = True
```

AllowNewBoxes

Property of VcGantt

This property permits (True) or prohibits (False) the user to create new boxes. If this property is set to **False**, the user cannot activate the **Mode: Create box** and it is not possible to set the **InteractionMode** to **VcCreateBox**. This property also can be set on the **General** property page.

	Data Type	Explanation
Property value	Boolean	Property active (True)/ not active (False)
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.AllowNewBoxes = False
```

AllowNewNodes

Property of VcGantt

This property permits (True) or prohibits (False) the user to create new nodes. If this property is set to **False**, the user cannot activate the **CreateNode** mode. This property also can be set on the **Nodes** property page.

	Data Type	Explanation
Property value	Boolean	Generating new nodes enabled/disabled **Default value:** True
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.AllowNewNodes = False
```

AllowNumericScaleRescale

Property of VcGantt

This property lets you set or retrieve whether the numeric scale resolution can be modified at run time.

This property can also be set on the **General** property page.

	Data Type	Explanation
Property value	Boolean	Numerical scale can be rescaled (True)/ cannot be rescaled (False)
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.AllowNumericScaleRescale = True
```

AllowPanningMode

Read Only Property of VcGantt

This property lets you move a screen section below a handcursor.

This property can also be set on the **General** property page.

	Data Type	Explanation
Property value	Boolean	Moving screen by mouse allowed (true)/not allowed (false) **Default value:** False
	Possible Values:	Group invisible/visible group nodes are/are not visible

AllowSelectionViaRubberRect

Property of VcGantt

This property lets you set/retrieve whether nodes can be selected in the empty diagram area by a rubber rectangle, drawn by mouse.

This property can also be set on the **General** property page.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	Zooming allowed (true)/not allowed (False) **Default value:** False Group invisible/visible group nodes are/are not visible

AllowTableColumnWidthOptimization

Property of VcGantt

This property permits (True) or prohibits (False) the user to let the column width rescale automatically. The optimization will be triggered when the user double-clicks on the separation line between the column to be optimized and the column on its right. Thereafter the event **OnOptimize-TableColumnWidth** is triggered. When the column width was modified, the event **OnTableColumnWidth** will occur.

This property can also be set on the **General** property page.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	Optimizing enabled/disabled **Default value:** True Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.AllowTableColumnWidthOptimization = False
```

AllowTimescaleRescale

Property of VcGantt

This property permits (True) or prohibits (False) the user to rescale the time scale. If the user is allowed to rescale the time scale, the event **OnTimeScaleRescale** is triggered after rescaling the time scale.

This property can also be set on the **General** property page.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	Re-scaling enabled/disabled **Default value:** True Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.AllowTimescaleRescale = False
```

AllowVerticalNodeMovement

Property of VcGantt

This property lets you set or retrieve whether nodes are allowed to be moved vertically in the diagram. This property also can be set on the **Nodes** property page.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	Vertical node movement in diagram enabled/disabled **Default value:** True Group invisible/visible group nodes are/are not visible

AllowVerticalNodeMovementViaTable

Property of VcGantt

This property lets you set or retrieve whether nodes are allowed to be moved vertically in the table. This property also can be set on the **Nodes** property page.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	Vertical node movement in table enabled/disabled **Default value:** true Group invisible/visible group nodes are/are not visible

Arrangement

Property of VcGantt

By this property you can set or retrieve whether the activities are arranged in a hierarchy or in groups. You can also set this property on the **Sorting** property page, by ticking the check box **Hierarchy**. This property is only effective if the property **HierarchyDataFieldIndex** or **GroupDataFieldIndex** was set, respectively.

	Data Type	Explanation
Parameter:		
Property value	VcArrangementType	Arrangement of activities groupwise or hierarchical **Default value:** vcArrangementTypeGroupwise
	Possible Values: vcArrangementTypeGroupwise 1 vcArrangementTypeHierarchical 2	 Groupwise Arrangement of activities Hierarchical Arrangement of activities

Example Code

```
VcGantt1.GroupLevelLayoutCollection.FirstGroupLevelLayout().GroupDataFieldIndex
= VcGantt1.DetectFieldIndex("Maindata", "Department")
VcGantt1.Arrangement = VcArrangementType.vcArrangementTypeGroupwise
VcGantt1.GroupNodes(True)

// alternativ:

VcGantt1.HierarchyDataFieldIndex = VcGantt1.DetectFieldIndex("Maindata",
"StructureCode")
VcGantt1.Arrangement = VcArrangementType.vcArrangementTypeHierarchical
VcGantt1.GroupNodes(True)
```

ArrowKeyMode

Property of VcGantt

By this property you can set the mode of the <left> and <right> arrow keys. Usually, the arrow keys are reserved for various interactions, such as scrolling the diagram, moving a marked field within a node or within the table. These navigating functions you can change by this property into modifying functions, so the user can move, enlarge or reduce the size of a node by them. A window displaying information on the position will remain on the screen for a few more seconds after the interaction finished to let the user read its content.

If the node being moved arrives at a border of the view, the diagram will automatically start scrolling (autoscroll).

By simply striking the arrow keys, a node will move; if the user in addition presses the <Shift> key, he or she can change the size of the node.

Assigning modifying functions to the arrow keys is very useful in low-resolution charts, since moving or resizing a node by mouse may be imprecise. Positioning a node by arrow keys is more precise, because each single step of the motion is indicated in the information window, representing a much higher resolution than is offered by the time scale. The step size is controlled by the properties **VcGantt.TimeUnit**, **VcGantt.TimeUnitsPer-Step** and **VcGantt.ArrowKeyStepSizeMultiplier**.

	Data Type	Explanation
Property value	Integer	Mode of the <left> and <right> arrow keys **Default value:** 0
	Possible Values:	
	vcStandard 127	The arrow keys <left> and <right> are in their default mode
	vcResizeOrMoveNode 384	The arrow keys <left> and <right> are in the mode to modify nodes

Example Code

```
'Assigning the function to an option button
Private Sub OptionEditNode_Click()
    If OptionStandard.Value = True Then
        VcGantt1.ArrowKeyMode = vcStandard
    Else
        VcGantt1.ArrowKeyMode = vcResizeOrMoveNode
    End If
End Sub
```

ArrowKeyStepSizeMultiplier

Property of VcGantt

This property lets you set or retrieve the value of the arrow keys step size multiplier. When moving the cursor by mouse or by arrow keys (see property **VcGantt.ArrowKeyMode**), the properties **VcGantt.TimeUnit** and **VcGantt.TimeUnitsPerStep** will determine the step size, multiplying their values. If for example the time unit was set to a day and the units per step were set to 2, the step size will be 2 days. Since by the mouse farther motion can be obtained simply by continued dragging, but keys do not offer anything comparable, this additional multiplier exists for the arrow keys. The user can activate it by pressing the <Ctrl> key in addition to the arrow key. If you set the value of the multiplier to 10, the step size in the above example will be 20 days per key stroke.

	Data Type	Explanation
Property value	Integer **Possible Values:**	Value of the multipier Data field index

Example Code

```
'Reducing the time scale resolution and enlarging the step size
Private Sub CommandExtendScale_Click()
    'Filling up the available space for the Gantt graph by extending the time
scale
    VcGantt1.TimeScaleEnd = DateSerial(2012, 1, 1)
    ‘ Reducing the resolution of the time scale by the factor 10
     VcGantt1.TimeScaleCollection.Active.Section(0).UnitWidth =
VcGantt1.TimeScaleCollection.Active.Section(0).UnitWidth / 10
    ' Increasing the multiplier for the arrow keys to advance in larger steps
    VcGantt1.ArrowKeyStepSizeMultiplier = 25
End Sub
```

AssignCalendarToNodes

Property of VcGantt

This property specifies whether a calendar is assigned to the nodes. Due to the calendar, the beginning/end of an activity will not be placed on a workfree day when shifted. Also, when calculating durations for activities, workfree days will be considered. A five-day-calendar is the default calendar. Beside, you can to define your own calendars. This property also can be set on the **Nodes** property page.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	A calendar is assigned (True) / is not assigned (False) **Default value:** False Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.AssignCalendarToNodes = False
```

BarSeparationGroupBy

Property of VcGantt

This property lets you set or retrieve the data field that controls the node separation of groups.This property only is available if nodes are grouped (property page **objects**, button **Grouping**, frame **groupwise**), where the

grouping options **nodes in separate lines** and **nodes optimized** were activated. Then you can select a data field to control the separation. Consequently, all nodes of a group that have the same value in this data field will be put in one line, even if they overlap each other.

Tip: Please note that in order to achieve a satisfactory result, the fields have to have the data type **Integer** or **Alphanumeric** and have to lie within the range of 1 - long_MAX (2147483647). If a field has the value 0 the node will not be kept together with the other nodes.

	Data Type	Explanation
Property value	Integer	Number of the field that should be used for the separation of nodes in groups
	Possible Values:	Data field index

Example Code

```
VcGantt1.BarSeparationGroupBy = 3
```

BorderArea

Read Only Property of VcGantt

This property gives access to the BorderArea object, i. e. the title and legend area.

	Data Type	Explanation
Property value	VcBorderArea	Title and legend area

Example Code

```
Dim borderArea As VcBorderArea

Set borderArea = VcGantt1.BorderArea
```

BoxCollection

Read Only Property of VcGantt

This property gives access to the BoxCollection object that contains all boxes available.

	Data Type	Explanation
Property value	VcBoxCollection	BoxCollection object

Example Code

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

Set boxCltn = VcGantt1.BoxCollection
For Each box In boxCltn
    List1.AddItem (box.Name)
Next
```

BoxFormatCollection

Read Only Property of VcGantt

This property gives access to the BoxFormatCollection object that contains all box formats available.

	Data Type	Explanation
Property value	VcBoxFormatCollection	BoxFormatCollection object

Example Code

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormat As VcBoxFormat

Set boxFormatCltn = VcGantt1.BoxFormatCollection
For Each boxFormat In boxFormatCltn
    List1.AddItem (boxFormat.Name)
Next
```

CalendarCollection

Read Only Property of VcGantt

This property gives access to the calendar collection object that contains all calendars available.

	Data Type	Explanation
Property value	VcCalendarCollection	CalendarCollection object

Example Code

```
Dim calendarCltn As VcCalendarCollection
Dim calendar As VcCalendar

Set calendarCltn = VcGantt1.CalendarCollection
For Each calendar In calendarCltn
    List1.AddItem (calendar.Name)
Next
```

CalendarGridCollection

Read Only Property of VcGantt

This property gives access to the calendar grid collection object that contains all calendar grids available.

	Data Type	Explanation
Property value	VcCalendarCollecGridCollection	CalendarGridCollection object

Example Code

```
Dim calendarGridCltn As VcCalendarGridCollection
Dim calendarGrid As VcCalendarGrid

Set calendarGridCltn = VcGantt1.CalendarGridCollection
For Each calendarGrid In calendarGridCltn
    List1.AddItem (calendarGrid.Name)
Next
```

CalendarProfileCollection

Read Only Property of VcGantt

This property gives access to the CalenderProfileCollection object that contains all calendar profiles available.

	Data Type	Explanation
Property value	VcCalendarProfileCollection	CalendarProfileCollection object

Example Code

```
Dim calendarProfileCltn As VcCalendarProfileCollection
Dim calendarProfile As VcCalendarProfile

Set calendarProfileCltn = VcGantt1.CalendarProfileCollection
```

ConfigurationName

Property of VcGantt

This property enables a configuration file (*.ini) to be loaded that all settings are adopted from, including the corresponding data interface.

You can specify either a local file including the path or an URL.

local file: The default configuration file *vcgantt.ini* should be stored in the directory where the *vcgantt.ocx* is registered. If you specify the file name without path, *vcgantt.ini* will be expected to exist in the installation directory.

If the specified file does not exist, the default configuration will be loaded, which does not necessarily exist at the end user.

URL: A URL should be used as configuration file only if the configuration is specified during runtime by the API because only then the *ini* and *ifd* files will be loaded from the URL specified. (Otherwise, if you specify a URL as a configuration file during design time, the *ini* and *ifd* files will be downloaded, but they will be stored in the Structured Storage (VB: *frx* file). That store will be used during runtime instead of loading the files directly.) So when embedding VARCHART ActiveX into an HTML page, you can specify the *ini* and *ifd* files directly, not needing other ways to temporarily create a local file which is considered insecure by browsers anyway.

Also see "Introduction: ActiveX Controls in Browser Environment"

Note: When loading a new configuration file, existing data will be lost and may have to be re-loaded again.

	Data Type	Explanation
Property value	String	File name **Default value:** vcgantt.ini
	Possible Values:	Name of the color map

Example Code

```
VcGantt1.ConfigurationName = "c:\VARCHART\XGantt\sample.ini"
' or:
VcGantt1.ConfigurationName = "http://members.tripod.de/netronic_te/ _
                                                xgantt_sample.ini"
```

ConsiderLinkRelationTypesOnNodeDragging

Property of VcGantt

When this property is set to True, the phantom lines that represent the links will be displayed indicating their type if dragged, and if links are switched on at all. The phantom lines will not start off from the center of the node, but from the left and right side of the node.

This property can also be set on the **General** property page.

	Data Type	Explanation

Example Code

```
VcGantt1.ConsiderLinkRelationTypesOnNodeDragging = True
```

ContextMenuForBoxesEnabled

Property of VcGantt

By this property you can set or retrieve whether the context menu for boxes is enabled.

This property can also be set on the **General** property page.

	Data Type	Explanation
Property value	Boolean	Contex menu for box is/is not enabled
	Possible Values:	Group invisible/visible group nodes are/are not visible

CtrlCXVProcessing

Property of VcGantt

This property automatically translates the key combinations <Ctrl>+<C>, <Ctrl>+<X> and <Ctrl>+<V> into the clipboard commands **CopyNodesTo-Clipboard**, **CutNodesToClipboard** and **PasteNodesFromClipboard**, respectively. You can suppress this feature in order to avoid conflicts with shortcuts for menu items in e.g. Visual Basic applications. This property can also be set on the **General** property page.u commands in Visual Basic. This property can also be set on the **General** property page.

	Data Type	Explanation
Property value	Boolean	Key combinations will/will not be translated into clipboard commands **Default value:** True
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.CtrlCXVProcessing = False
```

CurrentVersion

Read Only Property of VcGantt

This property lets you retrieve the number of the current version of the VARCHART XGantt object. This is an easy way to identify the version on your customer's system at runtime, and to probably request the installation to be repaired, if a version is identified which is too old. The version number can alternatively be found by the property page of the file vcgantt.ocx in the section **version** or it can be read by the FILEVERSION resource of that file.

	Data Type	Explanation
Property value	String	Version number
	Possible Values:	Name of the color map

Example Code

```
MsgBox VcGantt1.CurrentVersion
```

DataDefinition

Read Only Property of VcGantt

This property gives access to the current data definition object, in order to e.g. enquire field names or field types. The data definition of VcGantt has got two data definition tables: **vcMaindata** and **vcRelations**.

	Data Type	Explanation
Property value	VcDataDefinition	Data definition

Example Code

```
Dim dataDefinition As VcDataDefinition

Set dataDefinition = VcGantt1.DataDefinition
```

DataTableCollection

Read Only Property of VcGantt

This property gives access to the DataTableCollection object that contains the existing data tables.

	Data Type	Explanation
Property value	VcDataTableCollection	Data table collection object returned

Example Code

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable

Set dataTableCltn = VcGantt1.DataTableCollection
For Each dataTable In dataTableCltn
    List1.AddItem (dataTable.Name)
Next
```

DateLineCollection

Read Only Property of VcGantt

This property gives access to the DateLineCollection object which contains all date lines available.

	Data Type	Explanation
Property value	VcDateLineCollection	DateLineCollection object

Example Code

```
Dim dateLineCltn As VcDateLineCollection
Dim dateLine As VcDateLine

Set dateLineCltn = VcGantt1.DateLineCollection
For Each dateLine In dateLineCltn
    List1.AddItem (dateLine.Name)
Next
```

DateLineGridCollection

Read Only Property of VcGantt

This property gives access to the DateLineGridCollection object which contains all date line grids available.

	Data Type	Explanation
Property value	VcDateLineGridCollection	DateLineGridCollection object

Example Code

```
Dim dateLineGridCltn As VcDateLineGridCollection
Dim dateLineGrid As VcDateLineGrid

Set dateLineCltn = VcGantt1.DateLineCollection
For Each dateLine In dateLineCltn
    List1.AddItem (dateLine.Name)
Next
```

DateOutputFormat

Property of VcGantt

This property lets you specify or enquire the date output format. To compose the date you can use the below codes:

D:	first letter of the day of the week (not adjustable)
TD:	Day of the Week (adjustable by using the event **OnSupplyTextEntry**)
DD:	two-digit figure for the day of the month: 01-31
DDD:	first three letters of the day of the week (not adjustable)
M:	first letter of the name of the month (not adjustable)
TM:	name of the month (adjustable by using the event **OnSupplyTextEntry**)
MM:	two-digit figure for the month: 01-12
MMM:	first three letters of the name of the month (not adjustable)
YY:	two-digit figure for the year
YYYY:	four-digit figure for the year
WW:	two-digit figure for the number of the calendar week: 01-53
TW:	text for "calendar week" (adjustable by using the event **OnSupplyTextEntry**)
Q:	one-digit figure for the quarter: 1-4
TQ:	name of quarter (adjustable by using the event **OnSupplyTextEntry**)
hh:	two-digit figure for the hour in 24 hours format: 00-23
HH:	two-digit figure for the hour in 12 hours format: 01-12
Th:	Text of "o' clock" (adjustable by using the event **OnSupplyTextEntry**)
TH:	"am" or "pm" (adjustable by using the event **OnSupplyTextEntry**)
mm	two-digit figure for the minute: 00-59
ss:	two-digit figure for the second: 00-59
TS:	short date format, as defined in the regional settings of the windows control panel
TL:	long date format, as defined in the regional settings of the windows

control panel

TT: time format, as defined in the regional settings of the windows control panel

Note: Characters which are not to be interpreted as part of the date should be preceded by a backslash '\'. '\\' for instance results in "\'. The special characters: '**:, /, -**' and **blank** don't need '\' as prefix.

This setting is valid for the table area and for layer annotations in the node area. This property also can be set on the **General** property page.

	Data Type	Explanation
Property value	String **Possible Values:**	Date {DMYhms:;/} Name of the color map

Example Code

```
VcGantt1.DateOutputFormat = "DD.MM.YY"
```

DiagramAlternatingRowBackColor

Property of VcGantt

This property lets you set or retrieve a second background color to the diagram, which forms a linewise alternating pattern with the color set by the property **DiagramBackColor**. This property also can be set on the **Layout** property page.

	Data Type	Explanation
Property value	Color	RGB color values ({0...255},{0...255},{0...255}) **Default value:** (255,255,255)

Example Code

```
VcGantt1.DiagramAlternatingRowBackColor = RGB(255, 0, 0)
```

DiagramBackColor

Property of VcGantt

This property lets you set or retrieve the diagram background color. If you combine this property with the property **DiagramAlternatingRowBack-Color** you can generate a color pattern that alternates linewise. This property also can be set on the **Layout** property page.

	Data Type	Explanation
Property value	Color	RGB color values ({0...255},{0...255},{0...255}) **Default value:** (255,255,255)

Example Code

```
VcGantt1.DiagramBackColor = RGB(255, 0, 0)
```

DiagramHistogramHeightRatio

Property of VcGantt

By this property you can set or retrieve ratio (in %) of the height of the diagram area (without histogram) to the height of the histogram at the start of the program. If the ratio is -1 or 0, the histogram will be displayed completely at the start. This property also can be set on the **Layout** property page.

	Data Type	Explanation
Property value	Integer {-1, 0, 1, ..., 1000}	Ratio between diagram height and histogram height

Example Code

```
Dim ratio As Integer

ratio = VcGantt1.DiagramHistogramHeightRatio
```

DiagramHistogramHeightRatioEx

Property of VcGantt

This property lets you set or retrieve the ratio between the total height of the diagram (in %) and the height of the histogram.

In contrast to the **DiagramHistogramHeightRatio** property this property returns a "Double" value, thus achieving a higher level of accuracy. The use of this property has to be enabled by the **UseHigherDiagramHistogram-**

HeightRatioPrecision property or by activating the corresponding option on the **General** property page.

	Data Type	Explanation
Property value	Double	Height ratio

Example Code

```
VcGantt1.DiagramHistogramHeightRatioEx = 40
```

DiagramVisible

Property of VcGantt

This property lets you set or retrieve whether the diagram section (table and Gantt graph) should be visible. This property also can be set on the **Layout** property page.

	Data Type	Explanation
Property value	Boolean	Diagram section visible (True) / invisible (False)
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.DiagramVisible = False
```

DialogFont

Property of VcGantt

This property lets you set or retrieve the font name and font size in the dialogs of the VARCHART XGantt control that appear at run time. The object expected is a font object of your programming environment, e.g. in Visual Basic an object of the class **Stdfont**.

	Data Type	Explanation
Property value	StdFont	Font attributes

Example Code

```
Dim newFont As New StdFont

newFont.Size = 14
newFont.Name = "Verdana"

Set VcGantt1.DialogFont = newFont
```

DirectDataWritingModeEnabled

Property of VcGantt

If this property is set to "True", data modifications that are carried out by using **VcNode/VcLink/VcDataRecord/.set_DataField** or **.AllData** are directly stored to the data pool WITHOUT being evaluated (e.g. by filter analysis, mapping etc.).

Thus a better performance is achieved.

	Data Type	Explanation
Property value	Boolean	Data modifications without analysis are (True)/are not (False) carried out **Default value:** False
	Possible Values:	Group invisible/visible group nodes are/are not visible

DoubleOutputFormat

Property of VcGantt

This property lets you set or retrieve the output format of numbers as a double value in the Gantt diagram. The format is composed by the below characters:

- Text
- I
- D

plus the separators **comma** and **period**. **Text** represents a character string; **I** represents the figures before the decimal separator and **D** represents the figures after the decimal separator. The overall sequence is **Text I D Text**, where a comma and a period can be inserted in the places desired. An example be the number -284901,3458. By the format **I,DDDD ppm** it will be output as **-284901,3458 ppm**. By the format **$I,III.DD** it will be output as **$-284,901.35**.

This property can also be set on the **General** property page.

	Data Type	Explanation
Property value	String **Possible Values:**	Character string which describes the double format, for example "I,DDDD ppm" Name of the color map

Example Code

```
VcGantt1.DoubleOutputFormat = "I,DDDD ppm"
```

EditNewNode

Property of VcGantt

This property specifies whether or not the **Edit Data** dialog box appears when a new node is created. The **AllowNewNodes** property must be set to **True** to enable the user to create new nodes. This property also can be set on the **Nodes** property page.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	the **Edit Data** dialog appears/does not appear. **Default value:** False Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.EditNewNode = True
```

Enabled

Property of VcGantt

This property lets you disable the VARCHART XGantt control so that it will not react to mouse or keyboard commands.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	VARCHART ActiveX control enabled/disabled Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.Enabled = False
```

EnableSupplyTextEntryEvent

Property of VcGantt

This property lets you activate the **OnSupplyTextEntry** event. This event lets you modify the texts of context menus, dialog boxes, error messages, months' and days' names etc. that occur during run time, for example for translation into different languages.

This property can also be set on the **General** property page.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	Property active/not active Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.EnableSupplyTextEntryEvent = True
```

EventReturnStatus

Property of VcGantt

You will need this property only in a development environment which does not allow the setting of a return value in an event procedure as e.g. javascript.

With this property the default returnStatus is overwritten within the event method by the desired value.The setting is valid only for the event in which it was made.

	Data Type	Explanation
Property value	ReturnStatusEnum **Possible Values:** vcRetStatDefault 2 vcRetStatFalse 0 vcRetStatNoPopup 4 vcRetStatOK 1	Return value of the event **Default value:** vcRetStatOK The default behavior remains unchanged. The default behavior will not be performed. The popup of the right-click mouse menu is inhibited. The default behavior will be performed.

Example Code

```
Private Sub VcGantt1_OnDiagramRClick(ByVal x As Long, ByVal y As Long,
returnStatus As Variant)

    VcGantt1.EventReturnStatus = vcRetStatNoPopup
```

```
End Sub
```

EventsSecurityCheck

Property of VcGantt

This property lets you activate/deactivate the event security check. You also can set this property on the **General** property page.

	Data Type	Explanation
Property value	Boolean	Event security check on/out
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.EventsSecurityCheck = False
```

EventText

Read Only Property of VcGantt

You will need this property only in a development environment which does not allow the setting of the delivery parameter in an event procedure as e.g. javascript.

This property sets the ToolTipText. The setting is only valid for the corresponding event.

	Data Type	Explanation
Property value	String	Tool Tip
	Possible Values:	Name of the color map

Example Code

```
Private Sub VcGantt1_OnSupplyTextEntry(ByVal controlIndex As
VcGanttLib.TextEntryIndexEnum, TextEntry As String, returnStatus As Variant)

        VcGantt1.EventText = "Order189"
End Sub
```

ExtendedDataTables

Property of VcGantt

This property allows to choose between using merely two data tables (Maindata and Relations) and the advanced use of up to 90 data tables. The latter option is recommended. This property needs to be set at the beginning of your program, before data tables and data records are created.

This property can also be set on the **General** property page.

	Data Type	Explanation
Property value	Boolean	**True**: only two data tables (Maindata and Relations) **False**: up to 99 data tables **Default value:** False
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.ExtendedDataTables = True
```

ExtendedEditingBehavior

Property of VcGantt

This property lets you set or retrieve whether at run time the user is allowed to apply enhanced options for editiing the table. You can also set this property on the **General** property page.

	Data Type	Explanation
Property value	Boolean	Extended table editing enabled/disabled **Default value:** False
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.ExtendedEditingBehavior = True
```

FilePath

Property of VcGantt

This property lets you set the file path so that graphics files will be found in the directory specified, even if only a relative file name was specified. Otherwise the file will be searched in the current directory of the application and in the installation directory of the VARCHART ActiveX control.

This property should be set when the application is started during the initializing procedure of the VARCHART ActiveX control. We recommend to set the file path to the path of the application or to a subdirectory of the application. The advantage of this action is that the application can be stored in any directory.

	Data Type	Explanation
Property value	String **Possible Values:**	File path **Default value:** " " Name of the color map

Example Code

```
Dim graphicsPath As String

graphicsPath = App.Path & "\bitmaps"
VcGantt1.FilePath = graphicsPath
```

FilterCollection

Read Only Property of VcGantt

This property gives access to the FilterCollection object that contains all filters available.

	Data Type	Explanation
Property value	VcFilterCollection	FilterCollection object

Example Code

```
Dim filterCltn As VcFilterCollection
Dim filter As VcFilter

Set filterCltn = VcGantt1.FilterCollection
For Each filter In filterCltn
    List1.AddItem (filter.Name)
Next
```

FontAntiAliasingEnabled

Property of VcGantt

This property lets you set or retrieve whether fonts can be anti-aliased with GDI+. If the legibility of certain fonts - in particular non- latin ones - changes for the worse, the property should be set to **False**.

The anti-aliasing with GDI+ has yet another effect: regardless of the selected zoom factor, texts keep their relative dimension so that the number of characters that fits in a table field will always be the same. If the option is switched off the settings of the operating system are applied instead (the settings can be found in the **Control Panel**, dialog box **Display**, Tab **Appearance: Effects**). Thus, if the option **Smooth edges** is switched on in the **Control Panel**, the texts might still be anti-aliased, notwithstanding the settings of the **General** property page. In this case, at some zoom levels more text could be visible than at others, since the native edge smoothing does not guarantee that the same relative dimension is always kept.

This property also can be set on the **General** property page.

	Data Type	Explanation
Property value	Boolean	Characters will/will not be anti-aliased **Default value:** True
	Possible Values:	Group invisible/visible group nodes are/are not visible

GroupCollection

Read Only Property of VcGantt

If activities were grouped in a chart, this property gives access to the GroupCollection object that contains all groups available.

	Data Type	Explanation
Property value	VcGroupCollection	GroupCollection object

Example Code

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup

Set groupCltn = VcGantt1.GroupCollection
For Each group In groupCltn
    List1.AddItem (group.Name)
```

```
Next
```

GroupingField

Property of VcGantt

This property lets you set or retrieve the field in the data definition table that is to be used as a grouping criterion of a defined level. The groups by default will be sorted in the order of reading, by which the first activity of the group was loaded. The sorting order can be modified by the property **GroupingOrderField**).

This property also can be set in the **Grouping** dialog.

	Data Type	Explanation
Parameter:		
⇨ GroupingLevel	Integer	Grouping level (starting by 0)
	Possible Values:	Data field index
Property value	Integer	Field ID of the data definition table
	Possible Values:	Data field index

Example Code

```
Dim definitionTable As VcDataDefinitionTable

Set definitionTable = VcGantt1.DataDefinition.DefinitionTable(vcMaindata)
VcGantt1.GroupingField(0) = definitionTable.FieldByName("Code 1").ID
VcGantt1.GroupNodes (True)
```

GroupingModificationsAllowed

Property of VcGantt

This property lets you specify whether the user can collapse expanded groups or expand collapsed groups. The user can collapse/expand groups by double-clicking on the group heading in the table section, by clicking on the minus or plus symbol next to the group heading or by the context menu for groups. This property also can be set in the **Grouping** dialog.

	Data Type	Explanation
Parameter:		
⇨ groupingLevel	Integer	Grouping level
	Possible Values:	

		Data field index
Property value	Boolean **Possible Values:**	Modifications allowed (True)/ not allowed (False) Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.GroupingModificationsAllowed(0) = False
```

GroupingOrderField

Property of VcGantt

This property lets you specify what field of the data definition table is to be used for sorting the groups. By using **GroupingOrderField**, the groups will be sorted in ascending or descending alphabetical order by this field. This property also can be set in the **Grouping** dialog.

	Data Type	**Explanation**
Parameter: ⇨ groupingLevel	Integer **Possible Values:**	Grouping level Data field index
Property value	Long	Field index of the data definition table

Example Code

```
VcGantt1.GroupingOrderField (0) = 12
VcGantt1.GroupingSortOrder (0) = vcDescending
VcGantt1.SortGroups
```

GroupingSortOrder

Property of VcGantt

This property lets you specify the sorting order of groups (ascending or descending). By the property **GroupingOrderField** you can specify the field by that the groups are sorted. This property also can be set in the **Grouping** dialog.

	Data Type	**Explanation**
Parameter: ⇨ groupingLevel	Integer **Possible Values:**	Grouping level

		Data field index
Property value	SortOrderEnum **Possible Values:** vcAscending 1 vcDescending 2	Ascending or descending order **Default value:** vcAscending ascending order descending order

Example Code

```
VcGantt1.GroupingOrderField (0) = 12
VcGantt1.GroupingSortOrder (0) = vcAscending
VcGantt1.SortGroups
```

GroupLevelLayoutCollection

Read Only Property of VcGantt

This property gives access to the GroupLevelLayoutCollection object which contains all group level layouts available.

	Data Type	Explanation
Property value	VcGroupLevelLayoutCollection	GroupLevelLayoutCollection object

Example Code

```
Dim groupLevelLayoutCltn As VcGroupLevelLayoutCollection
Dim groupLevelLayout As VcGroupLevelLayout

Set groupLevelLayoutCltn = VcGantt1.GroupLevelLayoutCollection
For Each groupLevelLayout In groupLevelLayoutCltn
    List1.AddItem (groupLevelLayout.Name)
Next
```

GroupOptimizationOnInteractionsEnabled

Property of VcGantt

If this property is set to **True**, the nodes of the target group automatically are optimized on interactions such as creating nodes, moving nodes or modifying their start or end date, if they had been in the optimized state of display before. If this property is set to **False**, on the interactions mentioned the node will be placed at the cursor, if this doesn't cause nodes to overlap. If it does, the node will be placed with other nodes in the next line, if this doesn't cause overlaps. If it does, a new line will be created below the one where the cursor is and the node will be put there.

This property can also be set at design time on the **General** property page.

Also see the method **VcGroup.ReOptimizeNodes**.

	Data Type	Explanation
Property value	Boolean	The **Optimized** re-arrangement of nodes will (True) / will not (False) be performed on interaction **Default value:** True
	Possible Values:	Group invisible/visible group nodes are/are not visible

HierarchyDataFieldIndex

Property of VcGantt

This property lets you set or retrieve the index of the data field which defines the hierarchical order of activities. This can be done even **after** having loaded data already. The modifications only become effective after having set the arrangement of activities to **hierarchical** with the property **VcGantt.Arrangement** (**vcArrangementTypeHierarchical** and having carried out an update with the method **VcGantt.GroupNodes**.

	Data Type	Explanation
Property value	Long	Data field which defines the hierarchical order of activities

Example Code

```
VcGantt1.HierarchyDataFieldIndex = VcGantt1.DetectFieldIndex("Maindata",
"Hierarchy")
VcGantt1.Arrangement = vcArrangementTypeHierarchical
VcGantt1.GroupNodes True
```

HierarchyLevelLayout

Read Only Property of VcGantt

This property gives access to the HierarchyLevelLayout object. This object lets you set or retrieve the properties of the hierarchical arrangement of activities.

	Data Type	Explanation
Property value	VcHierarchyLevelLayout	HierarchyLevelLayout object

HistogramCollection

Read Only Property of VcGantt

This property gives access to the HistogramCollection object that contains all histograms available.

	Data Type	Explanation
Property value	VcHistogramCollection	HistogramCollection object

Example Code

```
Dim histogramCltn As VcHistogramCollection
Dim histogram As VcHistogram

Set histogramCltn = VcGantt1.HistogramCollection
For Each histogram In histogramCltn
    List1.AddItem (histogram.Name)
Next
```

HistogramSeparationLineColor

Property of VcGantt

This property lets you set/retrieve the color of the separation lines between histograms. This property also can be set on the **Layout** property page.

	Data Type	Explanation
Property value	Color	RGB color values ({0...255},{0...255},{0...255}) **Default value:** (255,255,255)

hWnd

Read Only Property of VcGantt

This property returns a handle. The Microsoft Windows operating environment identifies each form and control in an application by assigning it a handle, or **hWnd**. The **hWnd** property is used with Windows API calls. Many Windows operating environment functions require the **hWnd** of the active window as an argument.

Note: Because the value of this property can change while a program is running, never store the **hWnd** value in a variable.

	Data Type	Explanation
Property value	Long	Handle

Example Code

```
MsgBox (Me.hWnd)
```

InfoWindow

Read Only Property of VcGantt

This property gives access to the InfoWindow object that designates the information window of a node appearing in a Gantt chart when a node is created or modified.

	Data Type	Explanation
Property value	VcInfoWindow	InfoWindow object

InInteractionEventsEnabled

Property of VcGantt

This property lets you enable or disable the InInteractionEvents. This property can also be set on the **General** property page.

	Data Type	Explanation
Property value	Boolean	InInteractionEvents enabled (True) or disabled (False) **Default value:** False
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.InInteractionEventsEnabled = True
```

InPlaceEditingOnGroupsInDiagramEnabled

Property of VcGantt

This property lets you set or retrieve whether at run time the in-line editing of group data fields in the diagram should be permitted to the user. For this, the group data have to use their own data tables. You also can set this property on the **General** property page.

Note: If certain data fields are not to be editable, the **Editable** check box in the **Administrate Data Tables** dialog must not be ticked.

Also see **InPlaceEditingOnNodesInDiagramEnabled**, **InPlaceEditingOn-NodesInTableEnabled** and **InPlaceEditingOnGroupsInTableEnabled**.

	Data Type	Explanation
Property value	Boolean	In-line editing enabled (True) / not enabled (False) **Default value:** True
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.InPlaceEditingOnGroupsInDiagramEnabled = True
```

InPlaceEditingOnGroupsInTableEnabled

Property of VcGantt

This property lets you set or retrieve whether at run time the in-line editing of group data fields in the table should be permitted to the user. c You also can set this property on the **General** property page.

Note: If certain data fields are not to be editable, the **Editable** check box in the **Administrate Data Tables** dialog must not be ticked.

Also see **InPlaceEditingOnNodesInDiagramEnabled**, **InPlaceEditingOn-NodesInTableEnabled** and **InPlaceEditingOnGroupsInDiagramEnabled**.

	Data Type	Explanation
Property value	Boolean	In-line editing enabled (True) / not enabled (False) **Default value:** True
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.InPlaceEditingOnGroupsInTableEnabled = True
```

InPlaceEditingOnNodesInDiagramEnabled

Property of VcGantt

This property lets you set or retrieve whether at run time the in-line editing of node data fields in the diagram should be permitted to the user. You also can set this property on the **General** property page.

Note: If certain data fields are not to be editable, the **Editable** check box in the **Administrate Data Tables** dialog must not be ticked.

Also see **InPlaceEditingOnNodesInTableEnabled**, **InPlaceEditingOn-GroupsInTableEnabled** and **InPlaceEditingOnGroupsInDiagram-Enabled**.

	Data Type	Explanation
Property value	Boolean	In-line editing enabled (True) / not enabled (False) **Default value:** True
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.InPlaceEditingOnNodesInDiagramEnabled = True
```

InPlaceEditingOnNodesInTableEnabled

Property of VcGantt

This property lets you set or retrieve whether at run time the in-line editing of node data fields in the table should be permitted to the user. You also can set this property on the **General** property page.

Note: If certain data fields are not to be editable, the **Editable** check box in the **Administrate Data Tables** dialog must not be ticked.

Also see **InPlaceEditingOnNodesInDiagramEnabled**, **InPlaceEditingOn-GroupsInTableEnabled** and **InPlaceEditingOnGroupsInDiagram-Enabled**.

	Data Type	Explanation
Property value	Boolean	In-line editing enabled (True) / not enabled (False) **Default value:** True
	Possible Values:	

		Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.InPlaceEditingOnNodesInTableEnabled = True
```

InteractionMode

Property of VcGantt

This property activates/retrieves one of the available modes of interaction.

	Data Type	Explanation
Property value	InteractionModeEnum	Modes **create link, delete link, create node, delete node, pointer** **Default value:** vcPointer
	Possible Values:	
	vcCreateBox 36	Box creating mode
	vcCreateLink 4	Link creating mode
	vcCreateNode 2	Node creating mode
	vcDeleteLink 5	Link deleting mode
	vcDeleteNode 3	Node deleting mode
	vcPanning 6	Panning mode
	vcPointer 0	Select mode

Example Code

```
VcGantt1.InteractionMode = vcCreateNode
```

LayerCollection

Read Only Property of VcGantt

This property gives access to the layer collection that contains all layers available.

	Data Type	Explanation
Property value	VcLayerCollection	LayerCollection object

Example Code

```
Dim layerCltn As VcLayerCollection
Dim layer As VcLayer

Set layerCltn = VcGantt1.LayerCollection
For Each layer In layerCltn
    List1.AddItem (layer.Name)
Next
```

LegendView

Read Only Property of VcGantt

This property gives access to the LegendView object that lets you define the legend view of the diagram.

	Data Type	Explanation
Property value	VcLegendView	LegendView object

Example Code

```
Dim legendview As VcLegendView

Set legendview = VcGantt1.LegendView
legendview.Visible = True
```

LineFormatCollection

Read Only Property of VcGantt

This property gives access to the LineFormatCollection object that contains all line formats available.

	Data Type	Explanation
Property value	VcLineFormatCollection	LineFormatCollection object

LinkAppearanceCollection

Read Only Property of VcGantt

This property gives access to the LinkAppearanceCollection object that contains all link appearance objects available.

	Data Type	Explanation
Property value	VcLinkAppearanceCollection	LinkAppearanceCollection object

Example Code

```
Dim linkAppCltn As VcLinkAppearanceCollection
Dim linkApp As VcLinkAppearance

Set linkAppCltn = VcGantt1.LinkAppearanceCollection
For Each linkApp In linkAppCltn
    List1.AddItem (linkApp.Name)
Next
```

LinkCollection

Read Only Property of VcGantt

This property gives access to the link collection that contains all links defined.

	Data Type	Explanation
Property value	VcLinkCollection	LinkCollection object

Example Code

```
Dim linkCltn As VcLinkCollection
Dim link As VcLink

Set linkCltn = VcGantt1.LinkCollection
For Each link In linkCltn
    List1.AddItem (link.AllData)
Next
```

LinkPredecessorDataFieldIndex

Property of VcGantt

This property lets you set or retrieve the index of the data field which holds the identification of the predecessor node of the link. You can only set this property if data was not yet loaded.

This property can also be set on the **Links** property page.

	Data Type	Explanation
Parameter:		
identifierIndex	Integer	Index of predecessor node {0...2}
	Possible Values:	Data field index
Property value	Long	Field index of the data table

Example Code

```
Dim dataTable As VcDataTable
Dim dataRecord As VcDataRecord

'create Link DataTable
Set dataTable = VcGantt2.DataTableCollection.Add("LinkDataTable")
VcGantt1.LinksDataTableName = dataTable.Name
dataTable.DataTableFieldCollection.Add("Id").PrimaryKey = True
dataTable.DataTableFieldCollection.Add ("Predecessor")
dataTable.DataTableFieldCollection.Add ("Successor")
VcGantt1.DataTableCollection.Update

VcGantt1.LinkPredecessorDataFieldIndex(0) =
VcGantt1.DetectFieldIndex("LinkDataTable", "Id")
```

```
VcGantt1.LinkSuccessorDataFieldIndex(0) =
VcGantt1.DetectFieldIndex("LinkDataTable", "Id")

'Load Data
Set dataTable = VcGantt1.DataTableCollection.DataTableByName("LinkDataTable")
Set dataRecord = dataTable.DataRecordCollection.Add("1;1;2;")
VcGantt1.EndLoading
```

LinksDataTableName

Property of VcGantt

This property lets you set or retrieve the name of the data table which contains the fields for the links. This is only possible as long as no data has been loaded.

This property can also be set on the **Links** property page.

	Data Type	Explanation
Property value	String	Name of the data table which provides the fields for the links
	Possible Values:	Name of the color map

Example Code

```
Dim dataTable As VcDataTable
Dim dataRecord As VcDataRecord

'create Link DataTable
Set dataTable = VcGantt2.DataTableCollection.Add("LinkDataTable")
VcGantt1.LinksDataTableName = dataTable.Name
dataTable.DataTableFieldCollection.Add("Id").PrimaryKey = True
dataTable.DataTableFieldCollection.Add ("Predecessor")
dataTable.DataTableFieldCollection.Add ("Successor")
VcGantt1.DataTableCollection.Update

VcGantt1.LinkPredecessorDataFieldIndex(0) =
VcGantt1.DetectFieldIndex("LinkDataTable", "Id")
VcGantt1.LinkSuccessorDataFieldIndex(0) =
VcGantt1.DetectFieldIndex("LinkDataTable", "Id")

'Load Data
Set dataTable = VcGantt1.DataTableCollection.DataTableByName("LinkDataTable")
Set dataRecord = dataTable.DataRecordCollection.Add("1;1;2;")
VcGantt1.EndLoading
```

LinkSuccessorDataFieldIndex

Property of VcGantt

This property lets you set or retrieve the index of the data field which holds the identification of the successor node of the link. This is only possible as long as no data has been loaded.

This property can also be set on the **Links** property page.

	Data Type	Explanation
Parameter: identifierIndex	Integer **Possible Values:**	Index of predecessor node {0...2} Data field index
Property value	Long	Field index of the data table

Example Code

```
Dim dataTable As VcDataTable
Dim dataRecord As VcDataRecord

'create Link DataTable
Set dataTable = VcGantt2.DataTableCollection.Add("LinkDataTable")
VcGantt1.LinksDataTableName = dataTable.Name
dataTable.DataTableFieldCollection.Add("Id").PrimaryKey = True
dataTable.DataTableFieldCollection.Add ("Predecessor")
dataTable.DataTableFieldCollection.Add ("Successor")
VcGantt1.DataTableCollection.Update

VcGantt1.LinkPredecessorDataFieldIndex(0) =
VcGantt1.DetectFieldIndex("LinkDataTable", "Id")
VcGantt1.LinkSuccessorDataFieldIndex(0) =
VcGantt1.DetectFieldIndex("LinkDataTable", "Id")

'Load Data
Set dataTable = VcGantt1.DataTableCollection.DataTableByName("LinkDataTable")
Set dataRecord = dataTable.DataRecordCollection.Add("1;1;2;")
VcGantt1.EndLoading
```

LinkTypeDataFieldIndex

Property of VcGantt

This property lets you set or retrieve the index of the data field which contains the link type. This is only possible as long as no data has been loaded.

This property can also be set on the **Links** property page.

	Data Type	Explanation
Property value	Long	Data field which contains the link type

Example Code

```
Dim dataTable As VcDataTable

'create Link DataTable
Set dataTable = VcGantt1.DataTableCollection.Add("LinkDataTable")
VcGantt1.LinksDataTableName = dataTable.Name
dataTable.DataTableFieldCollection.Add("Id").PrimaryKey = True
dataTable.DataTableFieldCollection.Add ("Predecessor")
```

```
dataTable.DataTableFieldCollection.Add ("Successor")
dataTable.DataTableFieldCollection.Add("LinkType")
VcGantt1.DataTableCollection.Update
```

MapCollection

Read Only Property of VcGantt

This property gives access to the map collection that contains a defined number of maps. The maps contained are selected by the method **VcMap-Collection.SelectMaps**.

	Data Type	Explanation
Property value	VcMapCollection	MapCollection object

Example Code

```
Dim mapCltn As VcMapCollection

Set mapCltn = VcGantt1.MapCollection
mapCltn.SelectMaps vcAnyMap
```

MinimumRowHeight

Property of VcGantt

By this property you can assign a minimum height (unit: 1/100 mm) to a row. The height chosen should correspond to the average height of an activity. This property can also be set on the **Layout** property page.

The minimum row height only becomes effective if there is no activity in the row or if existing activities do not exceed the minimum row height. In all other cases the row height automatically adapts to the space required by the activities. The values permitted range between 2 and 1000.

	Data Type	Explanation
Property value	Long	Minimum row heigt

Example Code

```
VcGantt1.MinimumRowHeight = 100
```

MouseProcessingEnabled

Property of VcGantt

This property allows you to process mouse events in your own way. If you want your own processing method between the **OnMouseDown** event and

the **OnMouseUp** event, then set the **MouseProcessingEnabled** property to False for this time interval. Then VARCHART XGantt will ignore all mouse movements and clicks until this property is set to True again.

This property also can be set in the OnMouse* events.

	Data Type	Explanation
Property value	Boolean	Property active (True)/ not active (False) **Default value:** True
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
Private Sub VcGantt1_OnMouseDown(ByVal button As Integer, ByVal Shift As
Integer, ByVal x As Long, ByVal y As Long)

    VcGantt1.MouseProcessingEnabled = False

End Sub
```

MoveAllSelectedNodes

Property of VcGantt

This property lets you set or retrieve whether the user can move the marked nodes collectively. If it is disabled, only single nodes (depending on whether on the **Nodes** property page the **Move node when marked** check box was ticked) or layers can be moved by the mouse, even if several nodes where marked.

This property also can be set on the **Nodes** property page.

	Data Type	Explanation
Property value	Boolean	All marked nodes can be moved together (True)./Only single layers or nodes can be moved by the mouse, even if several nodes where marked (False).
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.MoveAllSelectedNodes = True
```

MoveLayersAsNodeWithShiftKey

Property of VcGantt

This property lets you set or retrieve whether the layers of a marked node are moved as a whole when the shift key is being pressed while dragging (True). Otherwise the layers can be moved individually only (False). This property also can be set on the **Nodes** property page.

	Data Type	Explanation
Property value	Boolean	Moving of all layers of a node with shift enabled/disabled **Default value:** true
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.MoveLayersAsNodeWithShiftKey = False
```

MoveNodeAlways

Property of VcGantt

This property lets you set or retrieve whether a node and hence all ist layers can be moved without having to be marked before. This property also can be set on the **Nodes** property page.

	Data Type	Explanation
Property value	Boolean	Moving of nodes as a whole without marking them before is switched on (true) or off (false)
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.MoveNodeAlways = True
```

MoveNodeWhenMarked

Property of VcGantt

This property lets you set or retrieve whether a marked node can be interactively moved as a whole (True). Otherwise single layers can be moved only (False). This property also can be set on the **Nodes** property page.

	Data Type	Explanation
Property value	Boolean	A marked node can be interactively moved as a whole (True)/only single layers can be moved (False)
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.MoveNodeWhenMarked = True
```

NewNodesViaDoubleClick

Property of VcGantt

This property lets you enable the user to create a new node by double-clicking in the diagram area. Note: The **AllowNewNodes** property must be set to True. This property also can be set on the **Nodes** property page.

	Data Type	Explanation
Property value	Boolean	Generating new nodes via double-click enabled/disabled **Default value:** False
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.NewNodesViaDoubleClick = True
```

NodeCalendarNameDataFieldIndex

Property of VcGantt

This property lets you set or retrieve the index of the data field to store the name of the calendar if you wish to use an individual calendar for a node. This is only possible as long as no data has been loaded. This property also can be set on the **Nodes** property page.

	Data Type	Explanation
Property value	Long	Index of the data field which contains the name of a node calendar

NodeCollection

Read Only Property of VcGantt

This property gives access to the NodeCollection object, that contains a defined number of nodes. The number of nodes is defined by the method **VcNodeCollection.SelectMaps**

	Data Type	Explanation
Property value	VcNodeCollection	NodeCollection object

Example Code

```
Dim nodeCltn As VcNodeCollection

Set nodeCltn = VcGantt1.NodeCollection
nodeCltn.SelectNodes (vcAll)
```

NodeDurationDataFieldIndex

Property of VcGantt

This property lets you set or retrieve the the index of the data field that contains the duration of an interactively created node. This is only possible as long as no data has been loaded. This property also can be set on the **Nodes** property page.

	Data Type	Explanation
Property value	Long	Index of the data field which holds the duration of an interactively created node

NodeEndDateDataFieldIndex

Property of VcGantt

This propery lets you set or retrieve the the index of the data field to store the end date of an interactively created activity. This is only possible as long as no data has been loaded. This property also can be set on the **Nodes** property page.

	Data Type	Explanation
Property value	Long	Index of the data field which holds the end date of an interactively created activity

NodeLevelLayout

Read Only Property of VcGantt

This property gives access to the NodeLevelLayout object. This object lets you set or retrieve the properties of the hierarchical arrangement of activites.

	Data Type	Explanation
Property value	VcNodeLevelLayout	NodeLevelLayout object

NodeRowNumberDataFieldIndex

Property of VcGantt

This property lets you set or retrieve the index of the data field which stores the row number of each activity. The modifications only become effective after having carried out an update with the method **VcGantt.UpdateRowNumberFields**.

This property also can be set on the **Nodes** property page.

	Data Type	Explanation
Property value	Long	Index of the data field which holds the row number of an activity

Example Code

```
Private Sub Form_Load()

VcGantt1.NodeRowNumberDataFieldIndex =
VcGantt1.DetectFieldIndex("NodeDataTable", "SortNumber")

   'Load data
   Call loadData

   VcGantt1.UpdateRowNumberFields
   VcGantt1.SaveAsEx "C:\ProjectData.txt", vcUnicodeEncoding
End Sub
```

NodesDataTableName

Property of VcGantt

This property lets you set or retrieve the name of the data table which provides the fields for the nodes. This is only possible as long as no data has been loaded. This property also can be set on the **Nodes** property page.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the data table which provides the fields for the nodes Name of the color map

Example Code

```
Dim dataTable As VcDataTable
Dim dataRecord As VcDataRecord

'create Node DataTable
 Set dataTable = VcGantt1.DataTableCollection.Add("NodeDataTable")
VcGantt1.NodesDataTableName = dataTable.Name
dataTable.DataTableFieldCollection.Add("Id").PrimaryKey = True
'Load Data
Set dataTable = VcGantt1.DataTableCollection.DataTableByName("NodeDataTable")
Set dataRecord = dataTable.DataRecordCollection.Add("1;Node One;")
Set dataRecord = dataTable.DataRecordCollection.Add("2;Node Two;")

Set dataTable = VcGantt1.DataTableCollection.DataTableByName("LinkDataTable")
Set dataRecord = dataTable.DataRecordCollection.Add("1;1;2;")
VcGantt1.EndLoading
```

NodeStartDateDataFieldIndex

Property of VcGantt

This property lets you set or retrieve the index of the data field to store the start date of an interactively created activity. This is only possible as long as no data has been loaded. This property also can be set on the **Nodes** property page.

	Data Type	Explanation
Property value	Long	Index of the data field which holds the start date of an interactively created activity

NodeTooltipTextField

Property of VcGantt

This property lets you require/set the index of the data field of a node to store the tooltip texts for VMF files. This text appears when in the WebViewer the right mouse button is pressed.

This property also can be set on the **Nodes** property page.

	Data Type	Explanation
Property value	Integer **Possible Values:**	Index of the node data field for tooltip texts **Default value:** 4 Data field index

Example Code

```
VcGantt1.NodeTooltipTextField = 1
```

NoOfInitialRows

Property of VcGantt

This property lets you set or retrieve the number of node rows at the program start. This property also can be set on the **Layot** property page.

	Data Type	Explanation
Property value	Long	Number of node rows at the program start

Example Code

```
VcGantt1.NoOfInitialRows = 1
```

OLEDragHorizontalMovementAllowed

Property of VcGantt

This property lets you set or retrieve whether a node can be moved if the control is the target component of an ongoing OLE Drag&Drop action. The property does not affect activities moved within the same Gantt chart.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	OLE drag&drop action allowed (true) / not allowed (False) **Default value:** False Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.OLEDragHorizontalMovementAllowed = True
```

OLEDragMode

Property of VcGantt

By this property you can set or retrieve, whether dragging a node beyond the limits of the current VARCHART XGantt control is allowed.

If the OLEDragMode was set to **vcOLEDragManual**, OLE dragging is not possible. If the property was set to **vcOLEDragAutomatic**, dragging a node beyond control limits will be started automatically.

On the start, the source component will fill the DataObject with the data it contains and will set the **effects** parameter before initiating the OLEStartDrag event, as well as other source-level OLE Drag & Drop events. This gives you control over the drag/drop operation and allows you to intercede by adding other data formats.

VARCHART XGantt by default uses the clipboard format CF_TEXT (corresponding to the vbCFText format in Visual Basic), that can be retrieved easily.

While dragging, the user can decide whether to shift or to copy the object by using the Ctrl key.

This property can also be set on the **Nodes** property page.

OLE Drag & Drop operations in VARCHART XGantt are compatible to the ones in Visual Basic. Methods, properties and events have the same names and results as the default objects of Visual Basic.

	Data Type	Explanation
Property value	OLEDragModeEnum	Dragging mode for objects to leave the VARCHART ActiveX control **Default value:** vcOLEDragManual
	Possible Values:	
	vcOLEDragAutomatic 1	Method OLEDrag is invoked automatically
	vcOLEDragManual 0	Method OLEDrag needs to be invoked separately.

Example Code

```
VcGantt1.OLEDragMode = vcOLEDragAutomatic
```

OLEDragViaDiagram

Property of VcGantt

This property lets you specify or retrieve whether OLE-DragDrop is enabled in the diagram area.

	Data Type	Explanation
Property value	Boolean	OLE DragDrop enabled/not enabled in diagram **Default value:** True
	Possible Values:	Group invisible/visible group nodes are/are not visible

OLEDragViaTable

Property of VcGantt

This property lets you specify or retrieve whether OLE-DragDrop is enabled in the table area.

	Data Type	Explanation
Property value	Boolean	OLE DragDrop enabled/not enabled in table **Default value:** True
	Possible Values:	Group invisible/visible group nodes are/are not visible

OLEDragWithOwnMouseCursor

Property of VcGantt

This property lets you disable the mouse cursor in the target control during an OLE drag operation. OLE Drag & Drop allows to set the cursor in the source control by the event **OLEGiveFeedback**. If you do this, two competing cursors will exist in the target control, that may appear to flicker. You can avoid the flickering by disabling the target cursor by this property. This property also can be set on the **Nodes** property page.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	Cursor occurs/does not occur in the target control **Default value:** True Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.OLEDragWithOwnMouseCursor = False
```

OLEDragWithPhantom

Property of VcGantt

This property lets you disable the display of an OLE drag phantom. Disabling the phantom is useful when generating a new object is omitted but merely the attributes of the object in the target control are modified. This property also can be set on the **Nodes** property page.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	Phantom occurs/does not occur **Default value:** True Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.OLEDragWithPhantom = False
```

OLEDropMode

Property of VcGantt

By this property you can set or retrieve, whether a node from a different VARCHART XGantt control can be dropped in the current control.

Dropping will not be allowed if you set the property to **OLEDropNone**. If you set it to **vcOLEDropManual**, you will receive the **OLEDragDrop** event that enables you to process the data received by the object dropped, e.g. to generate a node or to load a file. If the source and the target component are identical, you will receive either the event **OnNodeModifyEx** or **OnNodeCreate** as with OLE Drag&Drop switched off. If you set the property to **vcOLEDropAutomatic**, the dropping will automatically be

processed by the control, displaying a node in the place of the dropping action, if possible.

This property can be also set on the **Nodes** property page.

OLE Drag & Drop operations in VARCHART XGantt are compatible to the ones in Visual Basic. Methods, properties and events have the same names and results as the default objects of Visual Basic.

	Data Type	Explanation
Property value	OLEDropModeEnum	Dropping mode of the VARCHART ActiveX control to receiving objects from outside **Default value:** vcOLEDropNone
	Possible Values:	
	vcOLEDropAutomatic 2	The data of the object received are automatically processed and a node corresponding to the data received is displayed in the place of the dropping.
	vcOLEDropManual 1	The event **OLEDragDrop** is invoked for the programmer to process the data of the object received.
	vcOLEDropNone 0	Dropping of objects that do not originate from the current VARCHART ActiveX control is not allowed.

Example Code

```
VcGantt1.OLEDropMode = vcOLEDropAutomatic
```

OverlapLayerEnabled

Property of VcGantt

This property lets you activate the overlap layer of the diagram. Please also see the property **OverlapLayerName** and the property **UsedAsOverlap-Layer** at the layer object.

	Data Type	Explanation
Property value	Boolean	Overlap layer on (True) / off (False)
	Possible Values:	
		Group invisible/visible group nodes are/are not visible

OverlapLayerName

Property of VcGantt

This property lets you set or retrieve by ist name the layer that is designed to occur as the overlap layer in the diagram. The overlap layer needs to be created and described by methods an properties of the layer object and needs to be marked by the layer property **UsedAsOverlap Layer**. Finally, it needs to be activated by the property **OverlapLayerEnabled** of the Gantt object.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the overlap layer Name of the color map

PartialLoadThreshold

Read Only Property of VcGantt

This property lets you set or retrieve a value up to which he loading of nodes will be performed by an optimized partial update and not by a complete update of the data records.

If data records are added, a default loading cycle is started that is optimized for the loading of large amounts of data: structures as grouping and sorting, the calculating of summary bars etc. are being removed and created anew completely. This is convenient when large amounts of data are loaded into an empty chart. If, however, only few records are being loaded into an existing data strucure the reloading of only few nodes could take just as long as the loading of the existing nodes because the configuration of the above mentioned structures take up the main part of the performance.

The property **PartialLoadTreshold**offers an alternative: Only few data are inserted in an optimized form into an existing amount of data by a partial update. The value that is gvien here sets the threshold value up to which data are being inserted by a "small" update. The recommendable value depends on various factors in the respective application and has to be tested by the user:

- Number of the existing nodes
- Complexity of the Gantt(grouping, sorting, summary bars, links, mapping etc.)

The property should mainly be used when the chart contains already many nodes and only few shall be added at runtime.

This property can be also set by the properties of the control:

OLEDragWithPhantom	True
OLEDropMode	0 - vcOLEDropNone
OverlapLayerEnabled	True
OverlapLayerName	
PartialLoadThreshold	0
PhantomLayerHeight	200
RightTableDiagramWidthRatio	-1
RoundedLinkSlantsEnabled	False
RowHeightReductionEnabled	False
RowMargins	50
ScrollEventsEnabled	True
SelectedRowBackColorAsARGB	0
ShowNonWorkInterval	False
ShowTimeScaleDialog	True

Tip: The optimization can currently only be used for the **Maindata** table. Hence the setting will be ignored if data from other tables or links are being loaded in a loading cycle.

	Data Type	Explanation
Property value	Long	Number of nodes up to which loading of nodes will be performed partially **Default value:** 0

PhantomLayerHeight

Property of VcGantt

By this property you can set or retrieve the height of the layer phantom (in 1/100 mm) that appears when a node is created interactively.

	Data Type	Explanation
Property value	Integer **Possible Values:**	Height of the layer phantom Data field index

Example Code

```
Dim phantomLayerHeight As Integer

phantomLayerHeight = VcGantt1.PhantomLayerHeight
```

Printer

Read Only Property of VcGantt

This property gives access to the printer object. This object lets you set or retrieve the properties of the printer currently used.

	Data Type	Explanation
Property value	VcPrinter	Printer object

Example Code

```
Dim printerZoomfactor As Integer
Dim printerCuttingMarks As String

printerZoomfactor = VcGantt1.Printer.ZoomFactor
printerCuttingMarks = VcGantt1.Printer.CuttingMarks
```

ResourceScheduler2

Property of VcGantt

This property sets the ResourceScheduler2 object for resource scheduling.

	Data Type	Explanation
Property value	VcResourceScheduler2	ResourceScheduler2 object passed

Example Code

```
VcGantt1.ResourceScheduler2.TaskDataTableName = "Task"
VcGantt1.ResourceScheduler2.TaskDueDateFieldIndex = 1
VcGantt1.ResourceScheduler2.TaskReleaseDateFieldIndex = 2
'...
VcGantt1.ResourceScheduler2.Process
```

RightTable

Read Only Property of VcGantt

This property lets you or retrieve the table object on the right of the Gantt graph in order to access the formats used or to modify the table columns and headings.

	Data Type	Explanation
Property value	VcTable	Second table on the right

Example Code

```
Dim rightTable As VcTable

Set rightTable = VcGantt1.RightTable
```

RightTableDiagramWidthRatio

Property of VcGantt

This property lets you set or retrieve the ratio between the width of the right table and the width of the diagram (in %). If this property is set to -1, the table will always be displayed completely.

	Data Type	Explanation
Property value	Integer	Width ratio
		{-1, 1...100}
	Possible Values:	Data field index

Example Code

```
VcGantt1.RightTableDiagramWidthRatio = 40
```

RightTableDiagramWidthRatioEx

Property of VcGantt

This property lets you set or retrieve the ratio between the width of the right table and the width of the diagram (in %). If this property is set to -1, the table will always be displayed completely.

In contrast to the **RightTableDiagramWidthRatio** property this property returns a "Double" value, thus achieving a higher level of accuracy. The usage of this property has to be enabled by the **UseHigherTableDiagram-WidthRatioPrecision** property or by activating the corresponding option on the **General** property page.

	Data Type	Explanation
Property value	Double	Width ratio

Example Code

```
VcGantt1.RightTableDiagramWidthRatioEx = 40
```

RoundedLinkSlantsEnabled

Read Only Property of VcGantt

This property lets you set or retrieve whether the slants of links of the routing type **vcLRTOrthogonal** are to be displayed as quarter circles instead of straigt lines. This property can also be set on the **General** property page.

	Data Type	Explanation
Property value	System.Boolean	Slants of links are to be displayed/not displayed as quarter circles **Default value:** false

Example Code

```
VcGantt1.RoundedLinkSlantsEnabled = True
```

RowHeightReductionEnabled

Read Only Property of VcGantt

This property controls the way of calculating the row height in the diagram. If it is set to **false**, the vertical offsets of the layers are applied by using an imaginary zero line in the vertical center of a node line. To keep the zero line always in the center of the row, it thus may happen that either the top or the bottom row margin will seem rather broad. The layers with a vertical offset of 0, however, stay always vertically centered .

If this property is set to **true**, the imaginary zero line is still used, but its position is no longer necessarily in the center of the row but in a position that allows the row height to be as low as possible. Thus it may happen that layers with a vertical offset of 0 are not on the same level as the vertical centered text of the corresponding table row.

This feature can also be set on the **General** property page.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	Reduction of row height allowed (true)/not allowed (false) **Default value:** False Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.RowHeightReductionEnabled = True
```

RowMargins

Property of VcGantt

This property lets you set or retrieve the width between the upper/ lower node margins and the upper/lower margins of the node rows. This property can also be set on the **Layout** property page.

	Data Type	Explanation
Property value	Long	width between the upper/ lower node margins and the upper/lower margins of the node rows by 1/100 mm

Example Code

```
VcGantt1.RowMargins = 100
```

Sash3DStyleEnabled

Property of VcGantt

This property returns/sets whether the sash 3D style is enabled.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	3D style of sash switched on/off **Default value:** True Group invisible/visible group nodes are/are not visible

SashThickness

Property of VcGantt

This property returns/sets the sash thickness. Value range: 3 - 20 pixels.

	Data Type	Explanation

Scheduler

Read Only Property of VcGantt

This property returns the VcScheduler object.

	Data Type	Explanation
Property value	VcScheduler	Returns the VcScheduler object

ScrollEventsEnabled

Property of VcGantt

This property lets you enable or disable the scroll events. This feature can also be set by the **General** property page.

Note: The scroll events are **disabled** by default!

	Data Type	Explanation
Property value	System.Boolean	Scroll events enabled/disabled

Example Code

```
VcGantt1.ScrollEventsEnabled = True
```

SelectedRowBackColorAsARGB

Property of VcGantt

By this property you can assign a color to a selected row. You can use an alpha value that sets the degree of transparency to the color, in order to put a colored fog on the background color of the row (see properties **DiagramBackColor** and **DiagramAlternatingRowBackColor**).

The color is disabled by default since the default value is fully transparent.

The color value is composed by four parts: A (alpha), R (red), G (green) and B (blue). R, G and B cannot (!) be put together by the commonly used RGB macro. The most simple way is to use hexadecimal notation, for example in VB6 **&haarrggbb** or in C++ **0xaarrggbb**, where aa, rr, gg and bb may range between 00..FF (corresponding to the decimal values of 0..255). A value of 0 in the alpha position will result in complete transparency whereas 255 represents a completely solid color. Ascending values of R, G and B will show increasingly lightening colors, the ultimate values 0,0,0 and 255,255, 255 representing black and white, respectively.

This property also can be set on the **Layout** property page.

	Data Type	Explanation
Property value	Long	Color value **Default value:** 0

Example Code

```
VcGantt1.SelectedRowBackgroundColorAsARGB = &h77503DFF
```

ShowNonWorkInterval

Property of VcGantt

This property lets you set or retrieve whether workfree intervals are to be displayed in the nodes. This property also can be set on the **Nodes** property page.

Note: AssignCalendarToNodes has to be set to True.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	Show workfree intervals (true)/do not show workfree intervals (false/ Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.ShowNonWorkInterval = False
```

ShowSnapLines

Read Only Property of VcGantt

This property enables snap lines to be shown while nodes are being resized or dragged with the snap target mode switched on. These lines help to better recognize the defined snap targets.

This feature can also be switched on on the **Nodes** property page.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	Snap lines are/ are not shown **Default value:** false Group invisible/visible group nodes are/are not visible

ShowSnapMarkings

Read Only Property of VcGantt

This property enables snap markings to be shown at the nodes being defined as snap targets while nodes are being resized or dragged with the snap target mode switched on. These markings help to better recognize the defined snap targets.

This feature can also be switched on on the **Nodes** property page.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	Snap markings are/ are not shown **Default value:** false Group invisible/visible group nodes are/are not visible

ShowTimeScaleDialog

Property of VcGantt

This property lets you set or retrieve whether the **Edit Time scale** dialog box is to appear when the user double-clicks on the time scale. This property also can be set on the **General** property page.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	The **TimeScale** dialog box appears/does not appear. Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.ShowTimeScaleDialog = False
```

ShowToolTip

Property of VcGantt

This property lets you activate/deactivate the event **OnToolTipText**. This property also can be set on the **General** property page. The event **OnToolTipText** lets you edit the tooltip texts.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	Property active/not active **Default value:** False Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.ShowToolTip = True
```

SnapTargetNodesSelectionMode

Property of VcGantt

This property lets you specify whether nodes are selected atuomatically or manually when moving with the snap target mode switched on.

The property **VcNode.SnapTargetMode** selects the nodes as possible snap targets when manual selection is switched on.

	Data Type	Explanation
Property value	SnapTargetNodesSelectionModeEnum **Possible Values:** vcAutomatically 1 vcUserSelection 2	Nodes selection mode for moving with snap targets switched on **Default value:** vcAutomatically Autmatical selection of nodes Manual selection of nodes

SortField

Property of VcGantt

This property lets you specify the fields that the nodes are to be sorted by. Three sorting levels exist. For each one the field index can be specified. The sorting order you can specify by the **SortOrder** property. Sorting is to be triggered by the method **SortNodes**.

This property also can be set in the **Grouping** dialog.

	Data Type	Explanation
Parameter: ⇨ sortLevel	Integer **Possible Values:**	Sorting level {0...2} Data field index
Property value	Integer **Possible Values:**	Field index of the data definition table Data field index

Example Code

```
VcGantt1.SortField (0) = 11
VcGantt1.SortOrder (0) = vcDescending

VcGantt1.SortNodes
```

SortOrder

Property of VcGantt

This property specifies the sorting order (ascending or descending) for each of the three sorting levels. The sorting is triggered by the method **SortNodes**. This property also can be set in the **Grouping** dialog.

	Data Type	Explanation
Property value	Integer **Possible Values:** vcAscending 1 vcDescending 2	Ascending or descending order **Default value:** vcAscending ascending order descending order

Example Code

```
VcGantt1.SortField (0) = 11
VcGantt1.SortOrder (0) = vcDescending

VcGantt1.SortNodes
```

SubRowMargins

Property of VcGantt

This property lets you set or retrieve the vertical offset between sub rows (unit: 1/100 mm). Sub rows only come into existence if groups are displayed in an optimized way. Then nodes of the group are distributed to sub rows to

prevent them from overlapping. This property can also be set on the **Layout** property page.

	Data Type	Explanation
Property value	Long	width between sub rows by 1/100 mm ({0...200}) **Default value:** 50

Example Code

```
VcGantt1.SubRowMargins = 100
```

SummaryBarsVisible

Property of VcGantt

This property lets you set or retrieve whether summary bars are visible or not.

	Data Type	Explanation
Parameter: ⇨ GroupingLevel	Integer **Possible Values:**	*(Not for hierarchy)* grouping level (GroupingLevel = -1: reading: all levels, writing: at least one level) Data field index
Property value	Boolean **Possible Values:**	summary bars visible (True)/ invisible (False) Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.SummaryBarsVisible (-1) = False
```

Table

Read Only Property of VcGantt

This property gives access to the **table** object in order to access the formats used to modify its table columns and their headings.

	Data Type	Explanation
Property value	VcTable	Table

Example Code

```
Dim table As VcTable
```

```
Set table = VcGantt1.Table
```

TableCollection

Read Only Property of VcGantt

This property gives access to the table collection object that contains all tables available.

	Data Type	Explanation
Property value	VcTableCollection	TableCollection object

Example Code

```
Dim tableCltn As VcTableCollection
Dim table As VcTable

Set tableCltn = VcGantt1.TableCollection
For Each table In tableCltn
    List1.AddItem (table.Name)
Next
```

TableDiagramWidthRatio

Read Only Property of VcGantt

This property lets you set or retrieve the ratio between the width of the left table and the width of the diagram (in %). If this property is set to -1, the table will always be displayed completely.

	Data Type	Explanation
Property value	Integer	Width ratio
	Possible Values:	{-1, 1...100} Data field index

Example Code

```
VcGantt1.LeftTableDiagramWidthRatio = 40
```

TableDiagramWidthRatioEx

Property of VcGantt

This property lets you set or retrieve the ratio between the width of the left table and the width of the diagram (in %). If this property is set to -1, the table will always be displayed completely.

In contrast to the **LeftTableDiagramWidthRatio** property this property returns a "Double" value, thus achieving a higher level of accuracy. The usage of this property has to be enabled by the **UseHigherTableDiagram-WidthRatioPrecision** property or by activating the corresponding option on the **General** property page.

	Data Type	Explanation

Example Code

```
VcGantt1.LeftTableDiagramWidthRatioEx = 40
```

TimeScaleCollection

Read Only Property of VcGantt

This property gives access to the time scale collection that contains all time scales available.

	Data Type	Explanation
Property value	VcTimeScaleCollection	TimeScaleCollection object

Example Code

```
Dim timeScaleCltn As VcTimeScaleCollection
Dim timeScale As VcTimeScale

Set timeScaleCltn = VcGantt1.TimeScaleCollection
For Each timeScale In timeScaleCltn
    List1.AddItem (timeScale.Name)
Next
```

TimeScaleEnd

Property of VcGantt

This property lets you set or retrieve the end of the time scale. The date of the end needs to be later than the date of the start (also see the **TimeScaleStart** property), otherwise the setting will be ignored. At the same time the sequence of the statements set needs to be vice versa. We recommend to use the sequence of statements as shown in the source code sample below.

Note: The end date is not included. If you specify TimeScaleEnd = "31.12.2010" for example, the last day displayed will be the 30.12.2010.

	Data Type	Explanation
Property value	Date/Time	End date of the time scale {1.1.1980...31.12.2035}

Example Code

```
' Timescale from 1.10.2014 to 30.11.2014
VcGantt1.TimeScaleEnd = "01.12.14"
VcGantt1.TimeScaleStart = "01.10.14"
VcGantt1.TimeScaleEnd = "01.12.14"
```

TimeScaleStart

Property of VcGantt

This property lets you set or retrieve the start of the time scale. When setting, the date of the start needs to be earlier than the date of the end (also see the **TimeScaleEnd** property), otherwise the setting will be ignored by XGantt. At the same time the sequence of the statements set needs to be vice versa. We recommend to use the sequence of statements as shown in the source code example below.

	Data Type	Explanation
Property value	Date/Time	Start date of the time scale {1.1.1980...31.12.2035}

Example Code

```
' Timescale from 1.10.2014 to 30.11.2014
VcGantt1.TimeScaleEnd = "01.12.14"
VcGantt1.TimeScaleStart = "01.10.14"
VcGantt1.TimeScaleEnd = "01.12.14"
```

TimeUnit

Property of VcGantt

This property lets you set or retrieve the time unit used for the calculation of the duration (see "Layers") and for generating and modifying nodes interactively. If for example you have chosen the unit of a day, nodes can be generated or shifted by steps of days only, and the duration of nodes will also be calculated in days. This property can be set on the **General** property page.

Note: If you want to change the time unit, you should do this before reading data because later modifications are not effective.

	Data Type	Explanation
Property value	TimeUnitEnum **Possible Values:** vcDay 5 vcHour 6 vcMinute 7 vcSecond 8	Time unit **Default value:** vcDay Time unit **day** Time unit **hour** Time unit **minute** Time unit **second**

Example Code

```
Dim timeUnit As TimeUnitEnum

timeUnit = VcGantt1.TimeUnit
```

TimeUnitsPerStep

Property of VcGantt

This property lets you specify the number of time units covered by minimum interactive shifting of a node. This property also can be set on the **General** property page (**Smallest time interval**).

	Data Type	Explanation
Property value	Integer **Possible Values:**	Number of time units per step **Default value:** 1 Data field index

Example Code

```
VcGantt1.TimeUnitsPerStep = 4
```

ToolTipChangeDuration

Property of VcGantt

By this property you can set the duration that elapses before a subsequent tool tip window appears when the pointer moves to a different object. Unit: milliseconds. To reset this delay time to its default value of 98 msec, please set it to -1.

	Data Type	Explanation
Property value	Integer **Possible Values:**	Duration in milliseconds. Maximum value: 32767 msec **Default value:** -1

		Data field index

Example Code

```
VcGantt1.ToolTipText = "Object"
VcGantt1.ToolTipChangeDuration = 1000
```

ToolTipDuration

Property of VcGantt

By this property you can set the duration of the tool tip window to remain visible if the pointer is stationary within the bounding rectangle of an object. Unit: milliseconds. To reset this delay time to its default value of 5,000 msec, please set it to -1.

	Data Type	Explanation
Property value	Integer	Duration in milliseconds. Maximum value: 32767 msec **Default value:** -1
	Possible Values:	Data field index

Example Code

```
VcGantt1.ToolTipText = "Object"
VcGantt1.ToolTipDuration = 1000
```

ToolTipPointerDuration

Property of VcGantt

By this property you can set the duration during which the pointer must remain stationary within the bounding rectangle of an object before the tool tip window appears. Unit: milliseconds. To reset this delay time to its maximum value of 480 msec, please set it to -1.

	Data Type	Explanation
Property value	Integer	Duration in milliseconds **Default value:** -1
	Possible Values:	Data field index

Example Code

```
VcGantt1.ToolTipText = "Object"
VcGantt1.ToolTipPointerDuration = 1000
```

ToolTipShowAfterClick

Property of VcGantt

By this property you can set whether a tool tip window should disappear when its object is clicked (default behavior) or whether it should remain for the times set to it.

	Data Type	Explanation
Property value	Boolean	Tool tip window disappears (false) or remains (true) **Default value:** False
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.ToolTipShowAfterClick = True
```

TrackingSpaceBackColorAsARGB

Property of VcGantt

This property lets you set or retrieve the tracking space background color. This property also can be set on the **Layout** property page.

	Data Type	Explanation
Property value	Integer	RGB color values ({0...255},{0...255},{0...255}) **Default value:** (255,255,255)
	Possible Values:	Data field index

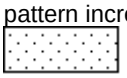
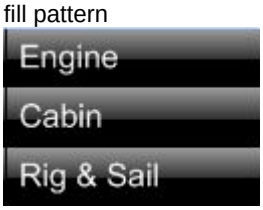
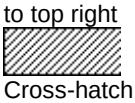
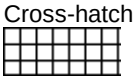
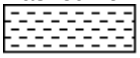
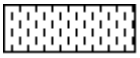
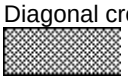
Example Code

```
VcGantt1.TrackingSpaceBackgroundColor = System.Drawing.Color.Blue
```

TrackingSpacePattern

Property of VcGantt

This property lets you set or retrieve the background pattern of the tracking space.

	Data Type	Explanation
Property value	FillPatternEnum	Pattern type
	Possible Values:	
	vc05PercentPattern... vc90PercentPattern 01 - 11	Dots in foreground color on background color, the density of the foreground pattern increasing with the percentage 
	vcAeroGlassPattern 40	Vertical color gradient in the color of the fill pattern 
	vcBDiagonalPattern 5	Diagonal lines slanting from bottom left to top right 
	vcCrossPattern 6	Cross-hatch pattern 
	vcDarkDownwardDiagonalPattern 2014	Diagonal lines slanting from top left to bottom right; spaced 50% closer than vcFDiagonalPattern and of twice the line width 
	vcDarkHorizontalPattern 2023	Horizontal lines spaced 50% closer than vcHorizontalPattern and of twice the line width 
	vcDarkUpwardDiagonalPattern 2015	Diagonal lines slanting from bottom left to top right, spaced 50% closer than vcBDiagonalPattern and of twice the line width 
	vcDarkVerticalPattern 2022	Vertical lines spaced 50% closer than vcVerticalPattern and of of twice the line width 
	vcDashedDownwardDiagonalPattern 2024	Dashed diagonal lines from top left to bottom right 
	vcDashedHorizontalPattern 2026	Dashed horizontal lines 
	vcDashedUpwardDiagonalPattern 2025	Dashed diagonal lines from bottom left to top right 
	vcDashedVerticalPattern 2027	Dashed vertical lines 
	vcDiagCrossPattern 7	Diagonal cross-hatch pattern, small 
	vcDiagonalBrickPattern 2032	Diagonal brick pattern 

vcDivotPattern 2036	Divot pattern
vcDottedDiamondPattern 2038	Diagonal cross-hatch pattern of dotted lines
vcDottedGridPattern 2037	Cross-hatch pattern of dotted lines
vcFDiagonalPattern 4	Diagonal lines slanting from top left to bottom right
vcHorizontalBrickPattern 2033	Horizontal brick pattern
vcHorizontalGradientPattern 52	Horizontal color gradient
vcHorizontalPattern 3	Horizontal lines
vcLargeCheckerboardPattern 2044	Checkerboard pattern showing squares of twice the size of vcSmallChecker-BoardPattern
vcLargeConfettiPattern 2029	Confetti pattern, large
vcLightDownwardDiagonalPattern 2012	Diagonal lines slanting to from top left to bottom right; spaced 50% closer than vcBDiagonalPattern
vcLightHorizontalPattern 2019	Horizontal lines spaced 50% closer than vcHorizontalPattern
vcLightUpwardDiagonalPattern 2013	Diagonal lines slanting from bottom left to top right, spaced 50% closer than vcBDiagonalPattern
vcLightVerticalPattern 2018	Vertical lines spaced 50% closer than vcVerticalPattern
vcNarrowHorizontalPattern 2021	Horizontal lines spaced 75 % closer than vcHorizontalPattern
vcNarrowVerticalPattern 2020	Vertical lines spaced 75% closer than vcVerticalPattern
vcNoPattern 1276	No fill pattern
vcOutlinedDiamondPattern 2045	Diagonal cross-hatch pattern, large
vcPlaidPattern 2035	Plaid pattern
vcShinglePattern 2039	Diagonal shingle pattern
vcSmallCheckerBoardPattern 2043	Checkerboard pattern

vcSmallConfettiPattern 2028	Confetti pattern
vcSmallGridPattern 2042	Cross-hatch pattern spaced 50% closer than vcCrossPattern
vcSolidDiamondPattern 2046	Checkerboard pattern showing diagonal squares
vcSpherePattern 2041	Checkerboard of spheres
vcTrellisPattern 2040	Trellis pattern
vcVerticalBottomLightedConvexPattern 43	Vertical color gradient from dark to bright
vcVerticalConcavePattern 40	Vertical color gradient from dark to bright to dark
vcVerticalConvexPattern 41	Vertical color gradient from bright to dark to bright
vcVerticalGradientPattern 62	Vertical color gradient
vcVerticalPattern 2	Vertical lines
vcVerticalTopLightedConvexPattern 42	Vertical color gradient from bright to dark
vcWavePattern 2031	Horizontal wave pattern
vcWeavePattern 2034	Interwoven stripe pattern
vcWideDownwardDiagonalPattern 2016	Diagonal lines slanting from top left to bottom right, showing the same spacing but three times the line width of vcF-DiagonalPattern
vcWideUpwardDiagonalPattern 2017	Diagonal lines slanting from bottom left to top right right, showing the same spacing but three times the line width of vcBDiagonalPattern
vcZigZagPattern 2030	Horizontal zig-zag lines

TrackingSpacePatternColorAsARGB

Property of VcGantt

This property lets you set or retrieve the pattern color of the tracking space. Color values have a transparency or alpha value, followed by a value for a red, a blue and a green partition (ARGB). The values range between 0..255. An alpha value of 0 equals complete transparency, whereas 255 represents a completely solid color.

	Data Type	Explanation
Property value	Integer	ARGB color values ({0...255},{0...255},{0...255},{0...255})
	Possible Values:	Data field index

UpdateBehaviorCollection

Read Only Property of VcGantt

This property gives access to the update behavior collection object that contains all update behaviors available.

	Data Type	Explanation
Property value	VcUpdateBehaviorCollection	UpdateBehaviorCollection object

Example Code

```
Dim updBehCltn As VcUpdateBehaviorCollection
Dim updBeh As VcUpdateBehavior

Set updBehCltn = VcGantt1.updBehCollection
For Each updBeh In updBehCltn
    List1.AddItem (updBeh.Name)
Next
```

UseHigherDiagramHistogramHeightRatioPrecision

Property of VcGantt

Set this property to "True" to enable the usage of the more accurate method **DiagramHistogramHeightRatioEx** or the event **OnHistogramHeight-ModifyEx** that return a value of the type "Double" to calculate the height ratio between diagram and histogram.

If this property is set to the default value "False", the method **DiagramHistogramHeightRatio** or the event **OnHistogramHeight** are used.

This property can also be set on the **General** property page.

	Data Type	Explanation
Property value	Boolean	More accurate methods to calculate the diagram/histogram height ratio are (True)/are not (False) used **Default value:** False
	Possible Values:	Group invisible/visible group nodes are/are not visible

UseHigherTableDiagramWidthRatioPrecision

Property of VcGantt

Set this property to "True" to enable the usage of the more accurate methods **LeftTableDiagramWidthRatioEx** and **RightTableDiagramWidthRatioEx** or the event **OnTableWidthModifyEx** that all return a value of the type "Double" to calculate the width ratio between table(s) and diagram.

If this property is set to the default value "False" then the methods **LeftTableDiagramWidthRatio** and **RightTableDiagramWidthRatio** or the event **OnTableWidth** are used.

This property can also be set on the **General** property page.

	Data Type	Explanation
Property value	Boolean	More accurate methods to calculate the table(s)/diagram width ratio are (True)/are not (False) used **Default value:** False
	Possible Values:	Group invisible/visible group nodes are/are not visible

UseSnapTargetsInInteractions

Property of VcGantt

This property lets you set or retrieve whether the snap targets are used on node/layer interactions.

	Data Type	Explanation
Property value	Boolean	Snap targets are used/not used on node/layer interactions
	Possible Values:	Group invisible/visible group nodes are/are not visible

UseTwinLineSashPhantom

Property of VcGantt

This property returns/sets whether a single or a double phantom line appears when interactively moving the sash with **standard** update behavior switched on.

	Data Type	Explanation
Property value	Boolean	Double phantom line while moving sash switched on/off **Default value:** True
	Possible Values:	Group invisible/visible group nodes are/are not visible

ViewComponentsBackColor

Property of VcGantt

This property lets you set or retrieve the diagram background color. If you combine this property with the property **DiagramAlternatingRowBack-Color** you can generate a color pattern that alternates linewise. This property also can be set on the **Layout** property page.

	Data Type	Explanation
Property value	Color	RGB color values ({0...255},{0...255},{0...255}) **Default value:** Color.White

Example Code

```
VcGantt1.ViewComponentsBackColor = System.Drawing.Color.Blue
```

ViewComponentsBorderColor

Read Only Property of VcGantt

This property lets you set or retrieve the border color of all frames at one time. This property also can be set on the **Layout** property page.

	Data Type	Explanation
Property value	Color	RGB color values ({0...255},{0...255},{0...255}) **Default value:** Color.White

Example Code

```
VcGantt1.ViewBorderColor = Color.Blue
```

WaitCursorEnabled

Property of VcGantt

This property lets you set or returns whether a wait cursor appears on time critical operations (like SheduleProject).

The property can also be set on the **General** property page.

	Data Type	Explanation
Property value	Boolean	Wait cursor is set/is not set **Default value:** False
	Possible Values:	Group invisible/visible group nodes are/are not visible

WorldView

Read Only Property of VcGantt

This property gives access to the VcWorldView object that defines the world view (complete view) of the diagram.

	Data Type	Explanation
Property value	VcWorldView	World View object

Example Code

```
Dim worldview As VcWorldView

Set worldview = VcGantt1.WorldView
worldview.Visible = True
```

ZoomFactor

Property of VcGantt

This property lets you set or retrieve the absolute zoom factor in percent (zoom factor = 100: original size, zoom factor > 100: enlargement, zoom factor < 100: reduction).

The absolute zoom factor is a rounded value and thus may display some inaccuracy.

Please also see the VcGantt methods **FitChartIntoView()** and **Zoom()**.

	Data Type	Explanation
Property value	Integer	absolute zoom factor (%)
	Possible Values:	Data field index

Example Code

```
VcGantt1.ZoomFactor = 150
```

ZoomingPerMouseWheelAllowed

Property of VcGantt

This property lets you set or retrieve whether zooming by the mouse wheel should be allowed to the user. This property also can be set on the **General** property page.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	Zooming allowed (true) / not allowed (False) **Default value:** False Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.ZoomingPerMouseWheelAllowed = True
```

Methods

AboutBox

Method of VcGantt

This method lets you open the **About** box. It contains an overview of the program and the library files currently used with the absolute path and version numbers. This feature makes the hotline support more comfortable. The overview can be selected by a mouse click, copied by the <Ctrl>+<C> keys and inserted by the <Ctrl>+<V> keys into a mail.

	Data Type	Explanation
Return value	Void	

Example Code

```
VcGantt1.AboutBox
```

Clear

Method of VcGantt

This method lets you delete all API objects created so far and restore the settings of the property pages carried out at design time.

	Data Type	Explanation
Return value	Boolean	The objects in the diagram were deleted successfully. {True}

Example Code

```
VcGantt1.Clear
```

ClearAll

Method of VcGantt

This method lets you delete all objects created so far and restore the settings of the property pages carried out at design time except for the calendars.

	Data Type	Explanation
Return value	Boolean	The Objects in the diagram were deleted successfully. {True}

Example Code

```
VcGantt1.ClearAll
```

ConvertDistance

Method of VcGantt

By this method you can convert distances from the unit of 1/100 mm into the unit of pixels, or vice versa. You can choose between x- and y-direction of the distance. The conversion takes into account the zoom factor set at a time (also see property **VcGantt.ZoomFactor**).

	Data Type	Explanation
Parameter:		
⇨ conversionType	DistanceConversionTypeEnum	Conversion type
	Possible Values:	
	vcXCentiMillimetersToPixels 1	Conversion of a distance in x-direction, from 1/100 millimeters to pixels.
	vcXPixelsToCentiMillimeters 3	Conversion of a distance in x-direction, from pixels to 1/100 millimeters.
	vcYCentiMillimetersToPixels 2	Conversion of a distance in y-direction, from 1/100 millimeters to pixels.
	vcYPixelsToCentiMillimeters 4	Conversion of a distance in y-direction, from pixels to 1/100 millimeters.
⇨ value	Long	Number of source units (that are to be converted)
Return value	Long	Number of target units (into which was converted)

DeleteLinkRecord

Method of VcGantt

This method lets you delete a link between two nodes. The link record will be identified by the primary keys set in the **Administrate Data Tables** dialog.

	Data Type	Explanation
Parameter: ⇨ linkRecordContent	Object	Content of the link record
Return value	Boolean	Link record was (True) / was not (False) deleted successfully.

Example Code

```
VcGantt1.DeleteLinkRecord ("A100;A105;;")
```

DeleteNodeRecord

Method of VcGantt

This method lets you delete a node. The node will be identified by the primary key in the node record. The data field that is used for the identification of nodes is set in the **Administrate Data Tables** dialog.

	Data Type	Explanation
Parameter: ⇨ nodeRecordContent	Object	Content of the node record
Return value	Boolean	Node record was (True) / was not (False) deleted successfully.

Example Code

```
VcGantt1.DeleteNodeRecord "A100;;;;;;"
```

DetectDataTableFieldName

Method of VcGantt

This method lets you retrieve the name of a data table field by its index.

	Data Type	Explanation
Parameter: ⇨ fieldIndex	Long	Index of the data table field of which the name is to be retrieved
Return value	String	Name of the data table field returned

Example Code

```
'Find the name of a DataTableField
Dim fieldName As String

fieldName = VcGantt1.DetectDataTableFieldName(0)
```

DetectDataTableName

Method of VcGantt

This method lets you retrieve the name of a data table by its index.

	Data Type	Explanation
Parameter:		
⇨ fieldIndex	Long	Index of the data table of which the name is to be retrieved
Return value	String	Name of the data table

Example Code

```
'Find the name of a DataTable
Dim tableName As String

tableName = VcGantt1.DetectDataTableName(0)
```

DetectFieldIndex

Method of VcGantt

This method lets you retrieve the index of a data table field by its name and the name of the data table.

	Data Type	Explanation
Parameter:		
⇨ dataTableName	String	Name of the data table that holds the field of which the index is to be retrieved
	Possible Values:	Name of the color map
⇨ dataTableFieldName	String	Name of the data table field of which the index is to be retrieved
	Possible Values:	Name of the color map
Return value	String	Index of the data table field returned

Example Code

```
'Find the index of a DataTableField
Dim fieldIndex As Integer
```

```
fieldIndex = VcGantt1.DetectFieldIndex("Maindata", "Name")
```

DumpConfiguration

Method of VcGantt

This method lets you save the configuration that consist of the .INI and the .IFD file.

This method should only be used for diagnosis purposes.

	Data Type	Explanation
Parameter:		
⇨ FileName	String	File name (including a path, if necessary)
	Possible Values:	Name of the color map
⇨ encoding	EncodingEnum	Mode of encoding
	Possible Values:	
	vcANSIEncoding 1	If a file was saved in ANSI encoding, it depends on the local settings of the Windows operating system. The file then contains characters which can be read correctly only if the language settings are the same as the ones that it was stored by.
	vcUnicodeEncoding 2	Saving a file in Unicode encoding makes it independent of whatever settings and hence should be the preferred mode if possible. If a file that was saved in Unicode encoding is to be loaded in Visual Basic 6 independently of the VARCHART component, it has to be treated in a special way.
Return value	Boolean	File was (True)/was not (False) stored successfully.

EditGroup

Method of VcGantt

This method invokes the **Edit Group data** dialog box for the group passed.

	Data Type	Explanation
Parameter:		
⇨ group	VcGroup	group whose data are to be edited
Return value	Boolean	group data were edited/editing was cancelled.

EditLink

Method of VcGantt

This method invokes the **Edit Link** dialog box for the link passed.

	Data Type	Explanation
Parameter:		
⇨ link	VcLink	Link the data of which are to be edited
Return value	Boolean	Link data were edited/edition was cancelled.

Example Code

```
Private Sub VcGantt1_OnLinkLClickCltn(ByVal linkCltn As _
                                VcGanttLib.VcLinkCollection, _
                                ByVal x As Long, ByVal y As Long, _
                                returnStatus As Variant)

Dim link As VcLink

    If linkCltn.Count > 0 Then
        For Each link In linkCltn
            VcGantt1.EditLink link
        Next link
    End If

End Sub
```

EditNode

Method of VcGantt

This method invokes the **Edit Data** dialog box for the node passed.

	Data Type	Explanation
Parameter:		
⇨ node	VcNode	Node whose data are to be edited
Return value	Boolean	Node data were edited./Editing was cancelled.

Example Code

```
Private Sub VcGantt1_OnNodeLClick(ByVal node As VcGanttLib.VcNode, _
                                ByVal location As VcGanttLib.LocationEnum, _
                                ByVal x As Long, ByVal y As Long, _
                                returnStatus As Variant)

    VcGantt1.EditNode node

End Sub
```

EndLoading

Method of VcGantt

This method indicates the finish of the loading procedure on the methods **InsertNodeRecord** and **InsertLinkRecord**, simultaneously triggering an update of the chart.

	Data Type	Explanation
Return value	Boolean	Loading was (True) / was not (False) finished.

Example Code

```
VcGantt1.EndLoading
```

ExportGraphicsToFile

Method of VcGantt

This method lets you store a Gantt diagram to a file without generating an **Save as** dialog box. You can store the files to the formats:

- *.BMP (Microsoft Windows Bitmap)
- *.EMF (Enhanced Metafile or Enhanced Metafile Plus)
- *.GIF (Graphics Interchange Format)
- *.JPG (Joint Photographic Experts Group)
- *.PNG (Portable Network Graphics)
- *.TIF (Tagged Image File Format)
- *.VMF (Viewer Metafile)
- *.WMF (Microsoft Windows Metafile, probably with EMF included)

EMF, EMF+, VMF and WMF are vector formats that allow to store a file independent of pixel resolution. All other formats are pixel-oriented and confined to a limited resolution.

The VMF format basically has been deprecated, but it will still be supported for some time to maintain compatibility with existing applications.

Further details on the different formats please find in the chapter **Important Concepts: Graphics Formats**.

When exporting to bitmap formats, setting 0 to the desired number of pixels of both, the x and the y direction, will keep the aspect ratio. If both pixel numbers equal 0, the size (in pixels) of the exported chart is calculated by VARCHART XGantt as listed below:

- PNG: a resolution of 100 dpi and a zoom factor of 100% are assumed. If alternatively a value of <= -50 is specified in the parameter SizeX, the absolute number will be used as DPI input. The number of DPIs will be stored to the PNG file, so with a given zoom factor display software can find the correct size for display.

- GIF, TIFF, BMP, JPEG: a resolution of 100 dpi and a zoom factor of 100% are assumed. If alternatively a value of <= -50 is specified in the parameter SizeX, the absolute number will be used as DPI input. In addition, an internal limit of 50 MBs of memory size is required for the uncompressed source bit map in the memory; so larger diagrams may have a smaller resolution than expected.

To formats of vector graphics, no pixel number can be set, but the below coodinate spaces:

- WMF: A fixed resolution is assumed where the longer side uses coordinates between 0 and 10,000 while the shorter side uses correspondingly smaller values to keep the aspect ratio.

- EMF/EMF+: The total resolution is adopted, using coordinates scaled by 1/100 mm in both, the x and y direction.

For further details on the different formats please read the chapter "Important Concepts: Graphics Formats".

	Data Type	**Explanation**
Parameter:		
⇨ FileName	String	File name (including a path, if necessary)
	Possible Values:	Name of the color map
⇨ PrintOutputFormat	PrintOutputFormat	Format of the file to be stored.
	Possible Values:	
	vcBMP 2	File will be written in the format BMP.
	vcEMF 9	File will be written in the format EMF.

	vcEMFPlus 12 vcEMFWithEMFPlusIncluded 11 vcEPS 3 vcGIF 4 vcJPG 5 vcPCX 6 vcPNG 7 vcTIF 8 vcVMF 0 vcWMF 1 vcWMFWithEMFIncluded 10	File will be written in the format EMF+, the standard extension is EMF. File will be written in the format EMF, additionally including the format EMF+. The standard extension is EMF. Deprecated File will be written in the format GIF. File will be written in the format JPG. Deprecated File will be written in the format PNG. File will be written in the format TIF. File will be written in the format VMF. File will be written in the format WMF. File will be written in the format WMF additionally including the format EMF. The standard extension is WMF.
⇨ SizeX	Integer **Possible Values:**	Width of the exported diagram in pixels. Available with pixel formats only. If this value is set to 0, its true size will be calculated from the aspect ratio. Data field index
⇨ SizeY	Integer **Possible Values:**	Height of the exported diagram in pixels. Available with pixel formats only. If this value is set to 0, its true size will be calculated from the aspect ratio. Data field index
Return value	Boolean	File was (True) / was not (False) stored successfully.

Example Code

```
VcGantt1.ExportGraphicsToFile"C:\temp\export", vcVMF, 0, 0
```

FitChartIntoView

Method of VcGantt

This method allows you to adjust the diagram to the control size while keeping the width-to-height-ratio so that either the height or the width of the diagram is completely visible. The method returns the relative enlargement or reduction in percent * 1000.

Please see also the property **ZoomFactor** and the method **Zoom()** of VcGantt.

	Data Type	**Explanation**
Parameter:		
⇨ fitMode	FitModeEnum **Possible Values:**	Selection of zoom factor

	vcFitHeight 23	The diagram is adjusted height-wise to the window size.
	vcFitMaximumOfWidthAnd Height 1051	The largest dimension of the diagram is adjusted to the window size.
	vcFitMinimumOfWidthAnd Height 1052	The smallest dimension of the diagram is adjusted to the window size.
	vcFitWidth 24	The diagram is adjusted width-wise to the window size.
	vcUseLargerZoomFactor 1053	The larger of the zoom factors is used. The corresponding dimension of the diagram does not fit into the window.
	vcUseSmallerZoomFactor 1054	The smaller of the zoom factors is used and the corresponding dimension of the diagram fits completely into the window.
Return value	Long	Relative zoom factor

Example Code

```
Dim myZoomfactor As Integer
VcGantt1.(FitChartIntoView(VcFitMode.vcFitWidth) / 1000)
```

FitHistogramsIntoView

Method of VcGantt

This method matches the visible histograms of the Gantt object into a view. For this, the histograms are re-scaled proportionally, so that their size ratio is maintained.

	Data Type	Explanation
Return value	Boolean	The histograms had to (True) / did not have to (False) be re-scaled.

Example Code

```
VcGantt1.FitHistogramsIntoView = True
```

FitRangeIntoView

Method of VcGantt

This method lets you match an arbitrary section of the time scale into a window to make the section visible.The size of the time units displayed will change in accordance with the window size and the size of the section defined. The beginning and the end are set by the **startVAlue** and **endValue** parameter, respectively. The parameter **gapAsNoOfTimeUnits** lets you set the number of time units, by which the visible section is to differ from the date at the beginning of the section displayed and by which the true end of the time scale is to differ from the end of the section displayed. The time unit itself you can set on the **General** property page.

	Data Type	Explanation
Parameter:		
⇨ startDate	Date/Time	Start date of the area to be matched
⇨ endDate	Date/Time	End date of the area to be matched
⇨ gapAsNoOfTimeUnits	Long	Number of time units to form the "gap" between startDate/endDate and the beginning of the visible section of the time scale start/end
Return value	Boolean	Area could/could not be matched.

Example Code

```
VcGantt1.FitRangeIntoView "14.09.14", "21.09.14", 1
```

GetAValueFromARGB

Method of VcGantt

A color value is composed by four parts: A (alpha), R (red), G (green) and B (blue). A value of 0 in the alpha position will result in complete transparency whereas 255 represents a completely solid color. Ascending values of R, G and B show increasingly lightening colors, the ultimate values 0,0,0 and 255,255,255 representing black and white, respectively. This method retrieves the alpha value of an ARGB value.

	Data Type	Explanation
Parameter:		
⇨ argb	Long	ARGB value, from which the alpha value is to be identified
Return value	Integer	Alpha value returned

Example Code

```
Dim alpha As Integer
Dim red As Integer
Dim green As Integer
Dim blue As Integer
Dim argb As Long
alpha = alpha + 11
red = red + 11
green = green + 11
blue = blue + 11
argb = VcGantt1.MakeARGB(alpha,red,green,blue)
alpha = VcGantt1.GetAValueFromARGB(argb)
```

GetBValueFromARGB

Method of VcGantt

A color value is composed by four parts: A (alpha), R (red), G (green) and B (blue). A value of 0 in the alpha position will result in complete transparency whereas 255 represents a completely solid color. Ascending values of R, G and B show increasingly lightening colors, the ultimate values 0,0,0 and 255,255,255 representing black and white, respectively. This method retrieves the "blue" value of an ARGB value.

	Data Type	Explanation
Parameter:		
⇨ argb	Long	ARGB value, from which the "blue" value is to be identified
Return value	Integer	"Blue" value returned

Example Code

```
Dim alpha As Integer
Dim red As Integer
Dim green As Integer
Dim blue As Integer
Dim argb As Long
alpha = alpha + 11
red = red + 11
green = green + 11
blue = blue + 11
argb = VcGantt1.MakeARGB(alpha,red,green,blue)
blue = VcGantt1.GetBValueFromARGB(argb)
```

GetCurrentComponentStart

Method of VcGantt

This method lets you retrieve the scroll value in 1/100 mm of a graphical element of the VARCHART XGantt control (time scale, diagram, histogram, table, table caption etc.) in any direction.

	Data Type	Explanation
Parameter:		
⇦ component	ComponentTypeEnum	Type of graphical element
	Possible Values:	
	vcAdditionalListComponent 1	additional table
	vcBottomListTitleComponent 14	bottom title bar
	vcBottomRightListTitleComponent 17	bottom right table
	vcBottomTimeScaleComponent 15	bottom time scale
	vcDiagramComponent 4	diagram
	vcHistogramComponent 8	histogram
	vcHistogramVerScaleComponent 7	numeric scale (vertical histogram scale)
	vcLegendComponent 10	legend (currently functionless; return values 00)

	vcListComponent 0 vcListTitleComponent 2 vcRightListComponent 5 vcRightListTitleComponent 16 vcTimeScaleComponent 3 vcTopTitleComponent 11	table table title table table title of the right table upper time scale upper title bar
⇦ scrollOrientation	ScrollOrientationEnum **Possible Values:** vcHorizontal 1 vcVertical 2	Direction of scrolling horizontal scrolling vertical scrolling
Return value	Long	Scroll value in 1/100 mm

GetCurrentViewDates

Method of VcGantt

This method lets you enquire the start and end dates of the visible section of the time scale.

Note: If you use VBScript, you can only use the analogue method **GetCurrentViewDatesAsVariant** because of the parameters by Reference.

	Data Type	Explanation
Parameter:		
⇦ leftDate	Date	Start date of the visible section of the time scale
⇦ rightDate	Date	End date of the visible section of the time scale
Return value	Boolean	Start/end dates of the visible section of the time scale are returned/not returned.

Example Code

```
Dim bGetCurrentViewDates As Boolean
Dim leftdate As Date
Dim rightdate As Date

GetCurrentViewDates = VcGantt1.GetCurrentViewDates(leftdate, rightdate)
```

GetCurrentViewDatesAsString

Method of VcGantt

This method lets you enquire the start and end dates of the visible section of the time scale. This method is identical with the method **GetCurrentViewDates** except the parameter format (string).

The date string has the fix format "DD.MM.YYYY;hh:mm:ss;".

	Data Type	Explanation
Parameter:		
⇦ leftDate	String **Possible Values:**	Start date of the visible section of the time scale Name of the color map
⇦ rightDate	String **Possible Values:**	End date of the visible section of the time scale Name of the color map
Return value	Boolean	Start/end dates of the visible section of the time scale are returned/not returned.

Example Code

```
Dim bGetCurrentViewDates As Boolean
Dim leftdate As String
Dim rightdate As String

bGetCurrentViewDates = VcGantt1.GetCurrentViewDates(leftdate, rightdate)
```

GetCurrentViewDatesAsVariant

Method of VcGantt

This method is identical with the method **GetCurrentViewDates** except for the parameters. It was necessary to implement this method because some languages (e.g. VBScript) can use parameters by Reference (indicated by ⇦) only if the type of these parameters is Variant.

Example Code

```
Dim bGetCurrentViewDates As Boolean
Dim leftdate As Variant
Dim rightdate As Variant

bGetCurrentViewDates = VcGantt1.GetCurrentViewDates(leftdate, rightdate)
```

GetDate

Method of VcGantt

This method lets you retrieve the date that corresponds to a x coordinate in the diagram section.

	Data Type	Explanation
Parameter:		
⇨ x	Long	X coordinate in the Gantt diagram, the corresponding date of which is to be retrieved
Return value	Date/Time	Date retrieved

Example Code

```
Private Sub VcGantt1_OnDiagramLClick(ByVal x As Long, ByVal y As Long, _
                                     returnStatus As Variant)

    Label1.Caption = VcGantt1.GetDate(x)

End Sub
```

GetDateAsString

Method of VcGantt

This method lets you retrieve the date that corresponds to a x coordinate in the diagram section.

	Data Type	Explanation
Parameter:		
⇨ x	Long	X coordinate in the Gantt diagram, the corresponding date of which is to be retrieved
Return value	String	Date retrieved

Example Code

```
Private Sub VcGantt1_OnDiagramLClick(ByVal x As Long, ByVal y As Long,
returnStatus As Variant)
    MsgBox VcGantt1.GetDateAsString(x)
End Sub
```

GetGValueFromARGB

Method of VcGantt

A color value is composed by four parts: A (alpha), R (red), G (green) and B (blue). A value of 0 in the alpha position will result in complete transparency whereas 255 represents a completely solid color. Ascending values of R, G and B show increasingly lightening colors, the ultimate values 0,0,0 and 255,255,255 representing black and white, respectively. This method retrieves the "green" value of an ARGB value.

	Data Type	Explanation
Parameter:		
⇨ argb	Long	ARGB value, from which the "green" value is to be identified
Return value	Integer	"Green" value returned

Example Code

```
Dim alpha As Integer
```

```
Dim red As Integer
Dim green As Integer
Dim blue As Integer
Dim argb As Long
alpha = alpha + 11
red = red + 11
green = green + 11
blue = blue + 11
argb = VcGantt1.MakeARGB(alpha,red,green,blue)
green = VcGantt1.GetGValueFromARGB(argb)
```

GetLinkByID

Method of VcGantt

This method gives access to a link by its identification which was specified on the **Administrate Data Tables** dialog. If the identification consists of more than one field (composite primary key), the multipart ID has to be noted as shown below:

ID=ID1|ID2|ID3

	Data Type	Explanation
Parameter:		
⇨ linkID	Variant	Link identification
Return value	VcLink	Link

Example Code

```
Dim link1 As VcLink
Dim successor As Integer

Set link1 = VcGantt1.GetLinkByID(" 1")
successor = link1.datafield(2)
```

GetLinkByIDs

Method of VcGantt

This method gives access to a link by the IDs of its predecessor node and its successor node. If the identification consists of more than one field (composite primary key), the multipart ID has to be noted as shown below:

ID=ID1|ID2|ID3

	Data Type	Explanation
Parameter:		
⇨ predecessorID	String **Possible Values:**	Identification of the predecessor node

		Name of the color map
⇨ successorID	String **Possible Values:**	Identification of the successor node Name of the color map
Return value	VcLink	Link

Example Code

```
Dim link As VcLink

Set link = VcGantt1.GetLinkByIDs(" 2", " 3")
```

GetNodeByID

Method of VcGantt

This method gives access to a node by its identification, which was specified on the **Administrate Data Tables** dialog. If the identification consists of several fields (composite primary key), this multipart ID has to be specified as follows:

ID=ID1|ID2|ID3

	Data Type	**Explanation**
Parameter: ⇨ nodeID	Variant	Node identification
Return value	VcNode	Node

Example Code

```
Dim node As VcNode

Set node = VcGantt1.GetNodeByID("10")
```

GetRValueFromARGB

Method of VcGantt

A color value is composed by four parts: A (alpha), R (red), G (green) and B (blue). A value of 0 in the alpha position will result in complete transparency whereas 255 represents a completely solid color. Ascending values of R, G and B show increasingly lightening colors, the ultimate values 0,0,0 and 255,255,255 representing black and white, respectively. This method retrieves the "red" value of an ARGB value.

	Data Type	Explanation
Parameter:		
⇨ argb	Long	ARGB value, from which the "red" value is to be identified
Return value	Integer	"Red" value returned

Example Code

```
Dim alpha As Integer
Dim red As Integer
Dim green As Integer
Dim blue As Integer
Dim argb As Long
alpha = alpha + 11
red = red + 11
green = green + 11
blue = blue + 11
argb = VcGantt1.MakeARGB(alpha,red,green,blue)
red = VcGantt1.GetRValueFromARGB(argb)
```

GetViewComponentSize

Method of VcGantt

This method lets you require at run time the size and position of a graphical element of the VARCHART ActiveX control (time scale, diagram, histogram, table, table caption etc.) (see event **OnViewComponentsSize-ModifyComplete**).

Note:

1. The position refers to the 0-origin of the graphical element of the VARCHART ActiveX control.

2. The values returned are pixel values.

3. If you use VBScript, due to the by-reference parameters you can only use the analogous method **GetViewComponentSizeAsVariant**.

	Data Type	Explanation
Parameter:		
⇨ viewComponent	ComponentTypeEnum	Component type
	Possible Values:	
	vcAdditionalListComponent 1	additional table
	vcBottomListTitleComponent 14	bottom title bar
	vcBottomRightListTitleComponent 17	bottom right table
	vcBottomTimeScaleComponent 15	bottom time scale
	vcDiagramComponent 4	diagram
	vcHistogramComponent 8	histogram
	vcHistogramVerScaleComponent 7	numeric scale (vertical histogram scale)

	vcLegendComponent 10 vcListComponent 0 vcListTitleComponent 2 vcRightListComponent 5 vcRightListTitleComponent 16 vcTimeScaleComponent 3 vcTopTitleComponent 11	legend (currently functionless; return values 00) table table title table table title of the right table upper time scale upper title bar
⇦ x	Long	X coordinate of the component
⇦ y	Long	Y coordinate of the component
⇦ width	Long	Component width
⇦ height	Long	Component height
Return value	Void	

Example Code

```
Private Sub handleHideHistogram()
   Dim x As Long
   Dim y As Long
   Dim width As Long
   Dim height As Long
   Dim scMod As Long
   scMod = ScaleMode
   ScaleMode = vbPixels

   VcGantt1.GetViewComponentSize vcHistogramVerScaleComponent, x, y, _
                         width, height

   ' plus 6 because of the sash
   Text1.Top = VcGantt1.Top + y + 6
   Text1.Left = VcGantt1.Left + x
   ' minus 25 because of the numeric scale
   Text1.width = width - 25
   ' minus 6 because of the sash
   Text1.height = height - 6
   ScaleMode = scMod
End Sub
```

GetViewComponentSizeAsVariant

Method of VcGantt

This method is identical with the method **GetViewComponentSize** except for the parameters. It was necessary to implement this event because some languages (e.g. VBScript) can use parameters by Reference (indicated by ⇦) only if the type of these parameters is VARIANT.

GroupNodes

Method of VcGantt

This methods lets you activate/deactivate the grouping. If you have set a grouping field by the **GroupingField** property or if you have set the grouping

order by the **GroupingOrderField** property, you need to activate the grouping by **GroupNodes**.

	Data Type	Explanation
Parameter:		
⇨ onOff	Boolean	Grouping on/off
	Possible Values:	Group invisible/visible group nodes are/are not visible
Return value	Boolean	Nodes were (True) / were not (False) grouped successfully.

Example Code

```
VcGantt1.GroupingField = 11
VcGantt1.GroupingOrderField = 12

VcGantt1.GroupNodes (True)
```

HistogramSetMaxYValue

Method of VcGantt

This method lets you specify the maximum value of the numeric scale of the (first) histogram. This value also can be set in the **Administrate Histograms** dialog (**End value**).

	Data Type	Explanation
Parameter:		
⇨ yValue	Long	Maximum y value
Return value	Long	Maximum y value set (1)/not set (0)

Example Code

```
VcGantt1.HistogramSetMaxYValue (40)
```

IdentifyField

Method of VcGantt

This method lets you identify the table field at a given cursor position.

	Data Type	Explanation
Parameter:		
⇨ x	Long	X coordinate of the cursor
⇨ y	Long	Y coordinate of the cursor

⇔ IdentifiedObjectType	Integer **Possible Values:**	Identified object type (for future use) Data field index
Return value	VcField	Field identified

Example Code

```
Private Sub VcGantt1_OnNodeRClick(ByVal node As VcGanttLib.VcNode, _
                                  ByVal location As VcGanttLib.LocationEnum, _
                                  ByVal x As Long, ByVal y As Long, _
                                  returnStatus As Variant)
    Dim field As VcField
    Dim objtype As Long

    Set field = VcGantt1.IdentifyField(x, y, objtype)
    If Not field Is Nothing then
       Label1.Caption = node.dataField(field.DataFieldID)
    End If
End Sub
```

IdentifyLayerAt

Method of VcGantt

This method lets you identify a layer. When a node was identified by the method **IdentifyObjectAt**, you can use it as a reference object for identifying its layer at the same position by a call of **IdentifyLayerAt**.

Note: If you are coding in VBScript, you will have to use the analogous method **IdentifyLayerAtAsVariant** because of the by-reference parameters.

	Data Type	**Explanation**
Parameter:		
⇨ x	Long	X coordinate of the cursor
⇨ y	Long	Y coordinate of the cursor
⇨ referenceNode	VcNode	Reference node
⇦ identifiedLayer	VcLayer	Layer identified
Return value	Boolean	Object identified/no object identified

Example Code

```
Private Sub VcGantt1_DragDrop(Source As Control, X As Single, Y As Single)

    Dim identifiedObj As Object
    Dim identifiedObjType As VcObjectTypeEnum
    Dim identifiedLayer As VcLayer
    Dim xPix, yPix As Long

    xPix = X / Screen.TwipsPerPixelX
    yPix = Y / Screen.TwipsPerPixelY
```

```
    Call VcGantt1.IdentifyObjectAt(xPix, yPix, identifiedObj, identifiedObjType)

    Select Case identifiedObjType
        Case vcObjTypeNodeInDiagram
            Call VcGantt1.IdentifyLayerAt(xPix, yPix, identifiedObj, _
                                identifiedLayer)
            If Not identifiedLayer Is Nothing Then
                MsgBox ("The Node """ + identifiedObj.DataField(0) + _
                        """, Layer """ + identifiedLayer.Name + _
                        """, was identified in the diagram area.")
            Else
                MsgBox ("The Node """ + identifiedObj.DataField(0) + _
                        """ was identified in diagram area; " + _
                        "no layer was identified.")
            End If

        Case vcObjTypeNodeInTable
            MsgBox ("The Node """ + identifiedObj.DataField(0) + _
                    """ was identified via the table.")

        Case Else
            MsgBox ("No node was identified.")
    End Select

End Sub
```

IdentifyLayerAtAsVariant

Method of VcGantt

This method is identical to the method **IdentifyLayerAt** except for the parameters. It was necessary to implement a separate because some languages (e.g. VBScript) can use by-reference parameters (indicated by ⇦) only if the type of these parameters is VARIANT.

IdentifyObject

Method of VcGantt

This method lets you identify an object that is located in the table or diagram section. The object type will be returned. When a node was identified by this method, you can use it as a reference object for identifying its layer at the same position by a second call of **IdentifyObject**.

If you use a development environment that always requires a reference to an object please use the method **IdentifyObjectAt** because in this method the parameter **reference object** is not needed.

	Data Type	Explanation
Parameter:		
⇨ x	Long	X coordinate of the cursor

⇨ y	Long	Y coordinate of the cursor
⇨ referenceObject	Object	Reference object that the ID refers to
⇦ identifiedObject	Object	Object identified
⇦ identifiedObjectType	VcObjectTypeEnum	Type of the object identified
	Possible Values:	
	vcObjTypeBox 15	object type **box**
	vcObjTypeCalendarGrid 18	object type **calendar grid**
	vcObjTypeCurve 12	object type **curve**
	vcObjTypeDateLine 9	object type **date line**
	vcObjTypeGroup 7	object type **group**
	vcObjTypeGroupInDiagram 11	object type **group in diagram area**
	vcObjTypeGroupInTable 7	object type **group in table area**
	vcObjTypeHistogram 13	object type **histogram**
	vcObjTypeLayer 8	object type **layer**
	vcObjTypeLinkCollection 3	object type **link collection**
	vcObjTypeNodeInDiagram 2	object type **node in diagram area**
	vcObjTypeNodeInLegend 17	object type **node in legend area**
	vcObjTypeNodeInTable 1	object type **node in table area**
	vcObjTypeNone 0	no object
	vcObjTypeNumericScale 10	object type **numeric scale**
	vcObjTypeSummaryNode 14	object type **summary bar**
	vcObjTypeTable 4	object type **table**
	vcObjTypeTableCaption 5	object type **table caption**
	vcObjTypeTimeScale 6	object type **time scale**
Return value	Boolean	Object identified/no object identified

Example Code

```
Private Sub VcGantt1 _DragDrop(Source As Control, X As Single, Y As Single)
    Dim label As Label
    Dim identifiedObj As Object
    Dim referenceObj As Object
    Dim identifiedObjType As Long
    Dim xPix, yPix As Long
    Dim colorValue As String

    xPix = X / Screen.TwipsPerPixelX
    yPix = Y / Screen.TwipsPerPixelY

    Set referenceObj = Nothing

    Call VcGantt1.IdentifyObject(xPix, yPix, referenceObj, identifiedObj, _
                                 identifiedObjType)

    Select Case identifiedObjType
        Case vcObjTypeNodeInDiagram
            Dim identifiedLayer As Object
            Dim identifiedSubObjType As Long
            Call VcGantt1.IdentifyObject(xPix, yPix, identifiedObj, _
                                         identifiedLayer, identifiedSubObjType)
            If identifiedSubObjType = VcGanttLib.VcObjectTypeEnum. _
                                         vcObjTypeLayer Then
                Dim node As VcNode
                Dim layer As VcLayer
                Set node = identifiedObj

                Set layer = identifiedLayer
                MsgBox ("The Node " + node.DataField(0) + "; Layer " + _
                                         layer.Name + " was identified in diagram
area")
```

```
            Else
                MsgBox ("The Node " + identifiedObj.DataField(0) + " was _
                                        identified in diagram area; no layer
identified")
            End If
        Case vcObjTypeNodeInTable
            MsgBox ("The Node " + identifiedObj.DataField(0) + " was _
                                        identified via the table")
        Case vcObjTypeGroup

            Dim barGroup As VcGroup
            Set barGroup = identifiedObj
            MsgBox ("The Group " + barGroup.Name + " was identified")
        Case vcObjTypeLinkCollection
            Dim linkCltn As VcLinkCollection
            Dim link As VcLink
            Set linkCltn = identifiedObj
            Set link = linkCltn.FirstLink
            While Not link Is Nothing
                MsgBox ("The Link " + link.AllData + " was identified")
                Set link = linkCltn.NextLink
            Wend
        Case vcObjTypeTable
            MsgBox ("The Table was hit")
        Case vcObjTypeTableCaption
            MsgBox ("The TableCaption was hit")
        Case vcObjTypeTimeScale
            MsgBox ("The Timescale " + identifiedObj.Name + " was identified")
        Case Else
            MsgBox ("No object identified.")
    End Select
End Sub
```

IdentifyObjectAt

Method of VcGantt

This method lets you identify any object in VARCHART XGantt. The object type will be returned. When a node was identified by this method, you can use it as a reference object for identifying its layer at the same position by a call of **IdentifyLayerAt**. If you want to identify a curve in a histogram you have to use the method **IdentifyObject**.

Note: If you use VBScript, you can only use the analogous method **IdentifyObjectAtAsVariant** because of the parameters by Reference.

	Data Type	Explanation
Parameter:		
⇨ x	Long	X coordinate of the cursor
⇨ y	Long	Y coordinate of the cursor
⇦ identifiedObject	Object	Object identified
⇦ identifiedObjectType	VcObjectTypeEnum	Type of the object identified
	Possible Values: vcObjTypeBox 15	object type **box**

	vcObjTypeCalendarGrid 18 vcObjTypeCurve 12 vcObjTypeDateLine 9 vcObjTypeGroup 7 vcObjTypeGroupInDiagram 11 vcObjTypeGroupInTable 7 vcObjTypeHistogram 13 vcObjTypeLayer 8 vcObjTypeLinkCollection 3 vcObjTypeNodeInDiagram 2 vcObjTypeNodeInLegend 17 vcObjTypeNodeInTable 1 vcObjTypeNone 0 vcObjTypeNumericScale 10 vcObjTypeSummaryNode 14 vcObjTypeTable 4 vcObjTypeTableCaption 5 vcObjTypeTimeScale 6	object type **calendar grid** object type **curve** object type **date line** object type **group** object type **group in diagram area** object type **group in table area** object type **histogram** object type **layer** object type **link collection** object type **node in diagram area** object type **node in legend area** object type **node in table area** no object object type **numeric scale** object type **summary bar** object type **table** object type **table caption** object type **time scale**
Return value	Boolean	Object identified/no object identified

Example Code

```
Private Sub VcGantt1_OnMouseMove(ByVal button As Integer, ByVal Shift As
Integer, ByVal x As Long, ByVal y As Long)

    Dim identifiedObject As Object
    Dim identifiedObjectType As VcObjectTypeEnum
    Dim node As VcNode

    Call VcGantt1.IdentifyObjectAt(x, y, identifiedObject, identifiedObjectType)

    Select Case identifiedObjectType
        Case VcObjectTypeEnum.vcObjTypeNodeInDiagram,
VcObjectTypeEnum.vcObjTypeNodeInTable
            Set node = identifiedObject
            Label1.Caption = node.DataField(1)
        Case Else
            Label1.Caption = ""
    End Select

End Sub
```

IdentifyObjectAtAsVariant

Method of VcGantt

This method is identical to the method **IdentifyObjectAt** except for the parameters. It was necessary to implement this event because some languages (e.g. VBScript) can use parameters by Reference (indicated by ⇦) only if the type of these parameters is VARIANT.

InsertLinkRecord

Method of VcGantt

This method lets you load the data of a link that connects two nodes. The data will be passed as a CSV string or as a data field in accordance with the structure defined in the **Administrate Data Tables** dialog in the **Relations** table. The method **EndLoading** should be invoked when the process of loading (links and nodes) is completed.

	Data Type	Explanation
Parameter:		
⇨ linkRecordContent	Object	Content of the link record
Return value	VcLink	Link

Example Code

```
VcGantt1.InsertNodeRecord ("A100;Activity 1;12.09.14;17.09.14;5;Planning")
VcGantt1.InsertNodeRecord ("A105;Activity 5;13.09.14;18.09.14;7;Testing")

VcGantt1.InsertLinkRecord ("A100;A105;FS;0")

VcGantt1.EndLoading

' or:

Dim linkRecord As Variant

linkRecord = Array("A100","A105","FS",0)

VcGantt1.InsertNodeRecord ("A100;Activity 1;12.09.14;17.09.14;5;Planning")
VcGantt1.InsertNodeRecord ("A105;Activity 5;13.09.14;18.09.14;7;Testing")

VcGantt1.InsertLinkRecord (linkRecord)

VcGantt1.EndLoading
```

InsertNodeRecord

Method of VcGantt

The data will be passed as a CSV string or as a data field in accordance with the structure defined in the **Administrate Data Tables** dialog in the **Maindata** table. The method **EndLoading** should be invoked when the process of loading (links and nodes) is completed.

	Data Type	Explanation
Parameter:		
⇨ nodeRecordContent	Data field	Content of the node record
Return value	VcNode	Node

Example Code

```
Dim nodeRecord As String
' data format: "Number;Name;Start date;Finish date;Group code;Group name"
nodeRecord = "A100;Activity 1;12.09.14;17.09.14;5;Planning"
VcGantt1.InsertNodeRecord (nodeRecord)

VcGantt1.EndLoading

' or

Dim nodeRecord As Variant

nodeRecord = Array("A100","Activity 1","12.09.14","17.09.14",5,"Planning")

VcGantt1.InsertNodeRecord (nodeRecord)

VcGantt1.EndLoading
```

MakeARGB

Method of VcGantt

This method lets you compose an ARGB value from the four single values of a color.

	Data Type	Explanation
Parameter:		
⇨ alpha	Integer **Possible Values:**	Alpha value Data field index
⇨ red	Integer **Possible Values:**	"Red" value Data field index
⇨ green	Integer **Possible Values:**	"Green" value Data field index
⇨ blue	Integer **Possible Values:**	"Blue" value Data field index
Return value	Long	ARGB value returned

Example Code

```
Dim alpha As Integer
Dim red As Integer
Dim green As Integer
Dim blue As Integer
Dim argb As Long
alpha = FF
red = A0
green = 34
blue = AB
argb = VcGantt1.MakeARGB(alpha,red,green,blue)
```

Open

Method of VcGantt

This method lets you load the records of the data tables of the selected file which had been saved earlier with the method **SaveAsEx(...)** in CSV format. CSV-Files may be retrieved and written in ANSI as well as in Unicode coding which is automatically recognized when read. The records are allocated to the corresponding data tables by using an appropriate identification line.

```
**** table name ****
```

Example:

```
**** Maindata ****
1;Node 1;07.05.2007;;5
2;Node 2;14.05.2007;;5
3;Node 3;21.05.2007;;5
**** Relations ****
1;1;2
2;2;3
```

Records of non existing tables are ignored when read. The contents of the data tables is replaced completely.

	Data Type	Explanation
Parameter:		
⇨ fileName	String **Possible Values:**	Name of the file to be opened Name of the color map
Return value	Boolean	File was (True) / was not (False) opened successfully.

Example Code

```
VcGantt1.Open "C:\Data\project1.csv"
```

OptimizeTimeScaleStartEnd

Method of VcGantt

This method lets you define the start and the end date of the time scale so that all nodes are completely visible. The start and end date are set in dependency on the displayed nodes. The parameter **NoOfUnits** lets you specify by how many time units the scale is to start on the left before the earliest start and by

how many time units it is to end on the right after latest finish of all activities. This property also can be set on the **General** property page.

	Data Type	Explanation
Parameter: ⇨ noOfUnits	Integer **Possible Values:**	Number of time units Data field index
Return value	Boolean	Timescale was (True) / was not (False) optimized successfully. The return value is **false** if both **TimeScaleStart** and **TimeScaleEnd** have not been modified. If no acitvities exist, the return value is always **false** because there are no date modifications.The specified number of time units is meaningless in such cases.

Example Code

```
VcGantt1.OptimizeTimeScaleStartEnd (5)
```

PageLayout

Method of VcGantt

This method lets you invoke the **Page Setup** dialog.

	Data Type	Explanation
Return value	Boolean	Dialog box was (True) / was not (False) opened successfully.

Example Code

```
VcGantt1.PageLayout
```

PrintDirectEx

Method of VcGantt

This method lets you print the diagram directly. A dialog box will not be displayed. If the printing was not successful the return value indicates the reason. This could be e.g. an entry in a log file.

	Data Type	Explanation
Return value	PrintResultStatusEnum	**Possible values:** vcPrintingSucceeded 0: Printing was performed successfully. vcNoPrinterInstalled 1: No printer was found neither the one specified by the call **VcPrinter.PrinterName** nor the one labeled as default printer by the Windows operating system. vcPrintingAbortedByUser 2: Printing was aborted by the user. vcPrintingAbortedByDriver 3: Printing was aborted by the Windows printer driver. vcUnprintablePageLayout 4. Printing could not be performed since the page layout did not match the printer properties such as paper size or margins.

Example Code

```
PrintStatusResultEnum status = VcGantt1.PrintDirectEx()
If status <> vcPrintingSucceeded Then
   Debug.Print "Printing failed: " & status & vbCrLf
End If
```

PrinterSetup

Method of VcGantt

This method lets you invoke the Windows **Print Setup** dialog box.

	Data Type	Explanation
Return value	Boolean	Dialog box was (True) / was not (False) opened successfully.

Example Code

```
VcGantt1.PrinterSetup
```

PrintIt

Method of VcGantt

This method triggers the printing of the diagram. The Windows **Print** dialog will open, using the parameters defined in the **PageLayout**.

	Data Type	Explanation
Return value	Boolean	Chart was (True) / was not (False) printed successfully.

Example Code

```
VcGantt1.PrintIt
```

PrintPreview

Method of VcGantt

This method invokes the print preview.

	Data Type	Explanation
Return value	Boolean	Dialog box was (True) / was not (False) opened successfully.

Example Code

```
VcGantt1.PrintPreview
```

PrintToFile

Method of VcGantt

This method lets you print the diagram directly to a file. Whether the printing is successful, depends on the printer driver since many PDF printer drivers do not accept file names.

	Data Type	Explanation
Parameter:		
⇨ fileName	String	File name
	Possible Values:	Name of the color map
Return value	Void	

Example Code

```
VcGantt1.PrintToFile
```

RecalculateAllStructureCodes

Method of VcGantt

By this method you can recalculate the structure code of the node hierarchy. The code is recalculated automatically after any modification. To avoid the recalculation for a set of actions, you can put them between the methods VcGantt.SuspendUpdate(true) and VcGantt.SuspendUpdate(false).

	Data Type	Explanation
Return value	Void	

Reset

Method of VcGantt

This methods lets you either delete objects (nodes, links, calendars etc.) from the diagram, the extent depending on the selected value of resetAction, or restore the settings of the property pages carried out at design time

	Data Type	Explanation
Parameter:		
⇨ resetAction	ResetActionEnum	Objects to be initialized or deleted
	Possible Values:	
	vcEmptyAllDataTables 4	The contents of all data tables are deleted but the data tables are kept.
	vcReloadConfiguration 2	Complete reinitialization. All settings and created objects are discarded.
	vcRemoveGroups 0	All groups and dependent objects, and thus all nodes and links are deleted.
	vcRemoveNodes 1	All nodes and dependent objects and thus also existing links are deleted.
Return value	Boolean	The objects in the diagram were deleted successfully. (True)

Example Code

```
VcGantt1.Reset(vcRemoveNodes) = True
```

SaveAsEx

Method of VcGantt

This method lets you save the records of all data tables to a file of CSV format, using the structure defined on the property page **Data Tables** invoked

by the property page **Objects**. Data tables that do not contain records will not be saved. If no file name was specified, the file most recently used by the **Open** method will be overwritten (correponding to the common **Save** function).

	Data Type	Explanation
Parameter:		
⇨ fileName	String	Name of the file to be saved
	Possible Values:	Name of the color map
⇨ encoding	EncodingEnum	Mode of encoding
	Possible Values:	
	vcANSIEncoding 1	If a file was saved in ANSI encoding, it depends on the local settings of the Windows operating system. The file then contains characters which can be read correctly only if the language settings are the same as the ones that it was stored by.
	vcUnicodeEncoding 2	Saving a file in Unicode encoding makes it independent of whatever settings and hence should be the preferred mode if possible. If a file that was saved in Unicode encoding is to be loaded in Visual Basic 6 independently of the VARCHART component, it has to be treated in a special way.
Return value	Boolean	File was (True)/was not (False) stored successfully.

Example Code

```
VcGantt1.SaveAsEx "C:\Data\project1.csv" , vcANSIEncoding
```

Schedule

Method of VcGantt

This method triggers a forward and a backward calculation of the current project. If you pass the start date, first a forward calculation will be performed, followed by a backward calculation. If you pass the final date, first a backward calculation will be performed, followed by a forward calculation. You can pass both dates, which will add the corresponding float to the activities. At least one date must be passed, otherwise an error message will occur. If a cycle amongst the nodes and links is identified, the ones affected will be marked.

	Data Type	Explanation
Parameter:		
⇨ startDate	Date/Time	Start date
Return value	Boolean	Forward scheduling was (True) / was not (False) successful

Example Code

```
VcGantt1.ScheduleProject("21.06.14", "")
```

ScrollComponentStartTo

Method of VcGantt

This method lets you scroll a graphical element of the VARCHART XGantt control (time scale, diagram, histogram, table, table caption etc.) in any direction to the indicated scroll value (the start coordinate) in 1/100 mm.

	Data Type	Explanation
Parameter:		
⇦ component	ComponentTypeEnum	Type of graphical element
	Possible Values:	
	vcAdditionalListComponent 1	additional table
	vcBottomListTitleComponent 14	bottom title bar
	vcBottomRightListTitleComponent 17	bottom right table
	vcBottomTimeScaleComponent 15	bottom time scale
	vcDiagramComponent 4	diagram
	vcHistogramComponent 8	histogram
	vcHistogramVerScaleComponent 7	numeric scale (vertical histogram scale)
	vcLegendComponent 10	legend (currently functionless; return values 00)
	vcListComponent 0	table
	vcListTitleComponent 2	table title
	vcRightListComponent 5	table
	vcRightListTitleComponent 16	table title of the right table
	vcTimeScaleComponent 3	upper time scale
	vcTopTitleComponent 11	upper title bar
⇦ scrollOrientation	ScrollOrientationEnum	Direction of scrolling
	Possible Values:	
	vcHorizontal 1	horizontal scrolling
	vcVertical 2	vertical scrolling
Return value	Boolean	Desired scroll value is/is not returned

ScrollToDate

Method of VcGantt

This method allows you to scroll to a particular date in the time scale. The **gapAsNoOfTimeUnits** parameter sets the number of time units that the gap between the specified date and the left or right edge of the time scale consists of (**vcLeftAligned** or **vcRightAligned**). By the parameter **horAlignment** you can specify if the date is to occur on the left or on the right side of the visible section of the time scale.

The time unit can be set on the **General** property page.

N.B: In case workfree times were collapsed, the collapsed times will be included in time calculations correctly, but they will not be displayed, which may lead to a seeming deviation from the values set.

	Data Type	Explanation
Parameter:		
⇨ date	Date/Time	Date
⇨ horAlignment	HorizontalAlignmentEnum	Horizontal alignment
	Possible Values: vcHorCenterAligned -1 vcLeftAligned -3 vcRightAligned -2	 horizontally centered left aligned right aligned
⇨ gapAsNoOfTimeUnits	Long	Number of time units
Return value	Boolean	Scrolling was (True) / was not (False) performed successfully.

Example Code

```
Call VcGantt1.ScrollToDate("20.10.14", vcLeftAligned, 2)
```

ScrollToGroupLine

Method of VcGantt

This method allows to scroll to the row containing a particular group and to specify whether that group should be displayed at the top, in the center or at the bottom of the screen.

	Data Type	Explanation
Parameter:		
⇨ group	VcGroup	Group to be scrolled to
⇨ verAlignment	VerticalAlignmentEnum	Vertical alignment
	Possible Values: vcBottomAligned 2 vcTopAligned 1 vcVerCenterAligned -1	 bottom aligned top aligned vertically centered
Return value	Boolean	Scrolling was (True) / was not (False) performed successfully.

ScrollToNode

Method of VcGantt

This method allows to scroll to a particular node and to specify whether that node should be displayed at the top, in the center or at the bottom of the screen.

	Data Type	Explanation
Parameter:		
⇨ node	VcNode	Node
⇨ verAlignment	VerticalAlignmentEnum	Vertical alignment
	Possible Values: vcBottomAligned 2 vcTopAligned 1 vcVerCenterAligned -1	 bottom aligned top aligned vertically centered
Return value	Boolean	Scrolling was (True) / was not (False) performed successfully.

Example Code

```
Dim node As VcNode

Set node = VcGantt1.GetNodeByID(" 2")
VcGantt1.ScrollToNode node, vcVerCenterAligned
```

ScrollToNodeLine

Method of VcGantt

This method allows to scroll to the row containing a particular node and to specify whether that node should be displayed at the top, in the center or at the bottom of the screen.

Note: If you choose the option **In one line**, all activities in a group will be displayed in one line. If the activities in the group coincide, they will be automatically displayed underneath one another in expanded mode to prevent overlapping. In this case using the **ScrollToNodeLine** method scrolls to the appropriate group row containing the selected node. Then it may happen that the selected node is not displayed in the center of the screen and is not visible.

	Data Type	Explanation
Parameter:		
⇨ node	VcNode	Node
⇨ verAlignment	VerticalAlignmentEnum	Vertical alignment

	Possible Values: vcBottomAligned 2 vcTopAligned 1 vcVerCenterAligned -1	 bottom aligned top aligned vertically centered
Return value	Boolean	Scrolling was (True) / was not (False) performed successfully.

Example Code

```
Imports NETRONIC.XGantt
 ...
Dim i As Integer
Dim objDataRecord(2) As Object
Dim dataRecordCol As VcDataRecordCollection

VcGantt1.ExtendedDataTablesEnabled = True
dataRecordCol =
VcGantt1.DataTableCollection.DataTableByName("Maindata").DataRecordCollection
    For i = 1 To 100
            objDataRecord(0) = i
            objDataRecord(1) = "Node " + i.ToString()
            dataRecordCol.Add(objDataRecord)
     Next
     VcGantt1.EndLoading()
     VcGantt1.ScrollToNodeLine(VcGantt1.GetNodeByID("50"),
VcVerticalAlignment.vcTopAligned)
```

ShowExportGraphicsDialog

Method of VcGantt

This method lets you invoke the **Save As** dialog for saving the diagram. Possible formats for saving:

- *.BMP (Microsoft Windows Bitmap)
- *.EMF (Enhanced Metafile or Enhanced Metafile Plus)
- *.GIF (Graphics Interchange Format)
- *.JPG (Joint Photographic Experts Group)
- *.PNG (Portable Network Graphics)
- *.TIF (Tagged Image File Format)
- *.VMF (Viewer Metafile)
- *.WMF (Microsoft Windows Metafile, probably EMF included)

EMF, EMF+, VMF and WMF are vector formats that allow to store a file independent of pixel resolution. All other formats are pixel-oriented and confined to a limited resolution.

The VMF format basically has been deprecated, but will be supported for some time to maintain compatibility with existing applications.

Further details on the different formats please find in the chapter **Important Concepts: Graphics Formats**.

When exporting, the size of the exported diagram will be calculated this way:

- PNG: a resolution of 100 dpi and a zoom factor of 100% are assumed. If alternatively a value of <= -50 is specified in the parameter SizeX, the absolute number will be used as DPI input.

- GIF, TIFF, BMP, JPEG: a resolution of 100 dpi and a zoom factor of 100% are assumed. If alternatively a value of <= -50 is specified in the parameter SizeX, the absolute number will be used as DPI input. In addition, an internal limit of 50 MBs of memory size is required for the uncompressed source bit map in the memory; so larger diagrams may have a smaller resolution than expected.

- WMF: A fixed resolution is assumed where the longer side uses coordinates between 0 and 10,000 while the shorter side uses correspondingly smaller values to keep the aspect ratio.

- EMF/EMF+: The total resolution is adopted, using coordinates scaled by 1/100 mm.

	Data Type	Explanation
Return value	Boolean	Chart was successfully (True) / was not successfully (False) exported

Example Code

```
VcGantt1.ShowExportGraphicsDialog
```

SortGroups

Method of VcGantt

This method lets you start the sorting of groups in a grouped diagram in accordance with the defined sorting parameter **GroupingOrderField** (**GroupingLevel**).

	Data Type	Explanation
Return value	Boolean	Groups were/were not sorted successfully.

Example Code

```
VcGantt1.GroupingOrderField(0) = 12
VcGantt1.SortGroups
```

SortNodes

Method of VcGantt

This method lets you start the sorting of the activities in accordance with the defined sorting parameters (**SortField** (**sortLevel**) and **SortOrder** (**sortLevel**)). If a grouping is activated, the sorting will be done separately for each group.

	Data Type	Explanation
Return value	Boolean	Nodes were/were not sorted successfully.

Example Code

```
VcGantt1.SortField (0) = 3
VcGantt1.SortOrder (0) = vcAscending

VcGantt1.SortNodes
```

SuspendUpdate

Method of VcGantt

For projects comprising many nodes, updating procedures may be very time consuming if actions are repeated for each node. You can accelerate the updating procedure by using the **SuspendUpdate** method. Bracket the code that describes the repeated action between **SuspendUpdate (True)** and **SuspendUpdate (False)** as in the below code example. This will get the nodes to be updated all at once and improve the performance.

	Data Type	Explanation
Parameter:		
⇨ suspendFlag	Boolean	SuspendUpdate(True): Start of the SuspendUpdate method/ SuspendUpdate(False): end of the SuspendUpdate method
	Possible Values:	Group invisible/visible group nodes are/are not visible
Return value	Void	

Example Code

```
VcGantt1.SuspendUpdate (True)

   If updateFlag Then
      For Each node In nodeCltn
         If node.DataField(2) < "07.09.14" Then
            node.DataField(13) = "X"
            node.UpdateNode
            counter = counter + 1
         End If
      Next node
   Else
      For Each node In nodeCltn
         If node.DataField(2) < "07.09.14" Then
            node.DataField(13) = ""
            node.UpdateNode
            counter = counter + 1
         End If
      Next node
   End If

VcGantt1.SuspendUpdate (False)
```

UpdateLinkRecord

Method of VcGantt

This method lets you modify the data of an existing link record. The link record will be identified by the primary key set in the **Administrate Data Tables** dialog. This method is used when external modifications of link data have to be carried out by the diagram. If the link updated does not exist, it will be generated.

	Data Type	Explanation
Parameter:		
⇨ linkRecordContent	Object	Content of the link record
Return value	VcLink	Link updated

Example Code

```
VcGantt1.UpdateLinkRecord ("A100;A105;FS;0")
```

UpdateNodeRecord

Method of VcGantt

This method lets you modify the data of an existing node record. The node record will be identified by the primary key defined in the **Administrate Data Tables** dialog. This method is used when external modifications of the data have to be carried out by the diagram.

	Data Type	Explanation
Parameter:		
⇨ nodeRecordContent	Object	Content of the node record
Return value	VcNode	Node record was (True) / was not (False) updated successfully.

Example Code

```
VcGantt1.UpdateNodeRecord ("A100;Activity 1;12.09.14;18.09.14;6;Planning")
```

UpdateRowNumberFields

Method of VcGantt

This method updates the field that stores the row number of the node. This field you can select on the **Nodes** property page from the **Row number field** combo box. Using this method is useful only if neither a hierarchical arrangement nor grouping are applied.

	Data Type	Explanation
Return value	Void	

Example Code

```
VcGantt1.UpdateRowNumberFields
VcGantt1.SaveAs ("C:\tmp\data.bar")
```

Zoom

Method of VcGantt

This method lets you enlarge/reduce the diagram on the display by the specified percentage factor (enlarging the diagram: zoom factor > 100, reducing the diagram: zoom factor < 100).

Please see also the VcGantt method **FitChartIntoView()** and the property **ZoomFactor**.

	Data Type	Explanation
Parameter:		
⇨ zoomFactor	Integer	Relative zoom factor
		{11...999}, other values will remain unconsidered
	Possible Values:	Data field index
Return value	Boolean	Zooming was (True) / was not (False) performed successfully.

Example Code

```
VcGantt1.Zoom 120
```

Events

Error

Event of VcGantt

This event occurs when an unforeseen error is found in the code of VARCHART XGantt. NETRONIC tries hard to avoid each error. This event helps to take down the errors that occur at the customers comfortably, e.g. in a file. The parameter profile is specified by the ActiveX default. Therefore some of the parameters that are passed are constant. The number always should be checked in the event, in order to prevent to suppress all error types in the future program development.

	Data Type	Explanation
Parameter:		
⇨ Description	String	Error description
	Possible Values:	Name of the color map
⇨ Scode	Long	&h800a402f (constant)
⇨ Source	String	Name of the control (constant)
	Possible Values:	Name of the color map
⇨ HelpFile	String	Help file: "" (constant)
	Possible Values:	Name of the color map
⇨ HelpContext	Long	Help context: 0 (constant)
⇦ CancelDisplay	Boolean	If True, then no normal error with number 71 (which could be catched via On Error GoTo) will be output.

	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
Private Sub VcGantt1_Error(Number As Integer, Description As String, _
          Scode As Long, Source As String, HelpFile As String, HelpContext _
          As Long, CancelDisplay As Boolean)

   Debug.Print CStr(Number) + " " + Description

End Sub
```

ErrorAsVariant

Event of VcGantt

This method is identical with the method **Error** except for the parameters. It was necessary to implement this event because some languages (e.g. VBScript) can use parameters by Reference (indicated by ⇦) only if the type of these parameters is VARIANT.

KeyDown

Event of VcGantt

This event occurs when the user presses a key while VARCHART XGantt has the focus. Key events allow to trigger VARCHART ActiveX functions by the keyboard. (For the interpretation of ANSI symbols please use the **KeyPress** event.)

	Data Type	Explanation
Parameter:		
⇨ KeyCode	Integer	Key code, e.g. vbKeyF1 (F1 key) or vbKeyHome (POS1 key)
	Possible Values:	Data field index
⇨ Shift	Integer	Number that indicates which one of the **Shift**, **Ctrl**, and **Alt** keys was pressed. **1** corresponds to the Shift key, **2** to the Ctrl key and **4** to the Alt key. Some, all, or none of the numbers may have been set, indicating that some, all, or none of the keys are depressed, respectively. When more than one key is in depressed state, their values add up. For example, if both the Ctrl and Alt keys are depressed, the value of **shift** would be "6".
	Possible Values:	Data field index

Example Code

```
Private Sub VcGantt1_KeyDown(KeyCode As Integer, Shift As Integer)
    MsgBox "key pressed"
End Sub
```

KeyPress

Event of VcGantt

This event occurs when the user presses and releases an ANSI key while VARCHART XGantt has the focus. Key events allow to trigger VARCHART ActiveX functions by the keyboard.

	Data Type	Explanation
Parameter:		
⇨ KeyAscii	Integer	An integer that returns the numerical key code of an default ANSI key. KeyAscii is returned as reference. If the parameter will be changed, another symbol will be returned to the object. If KeyAscii is set to 0, pressing a key will have no effect, i.e. no symbol will be passed to the object.
	Possible Values:	Data field index

Example Code

```
Private Sub VcGantt1_KeyPress(KeyAscii As Integer)
    MsgBox "Key pressed and released."
End Sub
```

KeyUp

Event of VcGantt

This event occurs when the user releases a key while VARCHART XGantt has the focus. Key events allow to trigger VARCHART ActiveX functions by using the keyboard. (To interprete ANSI symbols please use the **KeyPress** event.)

	Data Type	Explanation
Parameter:		
⇨ KeyCode	Integer	Key code, e.g. vbKeyF1 (F1 key) or vbKeyHome (POS1 key)
	Possible Values:	Data field index

⇨ Shift	Integer	Number that indicates which one of the **Shift**, **Ctrl**, and **Alt** keys was pressed. **1** corresponds to the Shift key, **2** to the Ctrl key and **4** to the Alt key. Some, all, or none of the numbers may have been set, indicating that some, all, or none of the keys are depressed, respectively. When more than one key is in depressed state, their values add up. For example, if both the Ctrl and Alt keys are depressed, the value of **shift** would be "6".
	Possible Values:	Data field index

Example Code

```
Private Sub VcGantt1_KeyUp(KeyCode As Integer, Shift As Integer)
    MsgBox "key released"
End Sub
```

OLECompleteDrag

Event of VcGantt

This event occurs when a source component is dropped onto a target component, informing the source component that a drag&drop operation was either performed or canceled.

	Data Type	Explanation
Parameter:		
⇨ effect	Long	Operation performed in the target component
	Possible Values:	
	vcDropEffectCopy 1	Drop results in a copy of data from the source to the target. The original data is unaltered by the drag operation.
	vcDropEffectMove 2	Drop results in data being moved from the source to the target. The source should remove the data from itself after the move.
	vcDropEffectNone 0	Target cannot accept the data.

Example Code

```
Private Sub VcGantt1_OLECompleteDrag(ByVal effect As Long)

    MsgBox effect

End Sub
```

OLEDragDrop

Event of VcGantt

Occurs when during OLE Drag & Dropping a source component is dropped onto a target component and if the **OLEDropMode** property of the target component is set to **vcOLEDropManual**and source and target component

are not identical. If they are identical you will receive either the event **OnNodeModifyEx** or **OnNodeCreate**.

	Data Type	Explanation
Parameter:		
⇨ data	DataObject	Object data passed
⇨ effect	Long	Operation to be performed
	Possible Values: vcDropEffectCopy 1	Drop results in a copy of data from the source to the target. The original data is unaltered by the drag operation.
	vcDropEffectMove 2	Drop results in data being moved from the source to the target. The source should remove the data from itself after the move.
	vcDropEffectNone 0	Target cannot accept the data.
⇨ button	Integer	Indicates the mouse button(s) pressed: **1** represents the left button, **2** is the right button, and the middle button is represented by **4**.
	Possible Values:	Data field index
⇨ shift	Integer	Number that indicates which one of the **Shift**, **Ctrl**, and **Alt** keys was pressed. **1** corresponds to the Shift key, **2** to the Ctrl key and **4** to the Alt key. Some, all, or none of the numbers may have been set, indicating that some, all, or none of the keys are depressed, respectively. When more than one key is in depressed state, their values add up. For example, if both the Ctrl and Alt keys are depressed, the value of **shift** would be "6".
	Possible Values:	Data field index
⇨ x	Long	X coordinate of the mouse cursor
⇨ y	Long	Y coordinate of the mouse cursor

OLEDragOver

Event of VcGantt

This event occurs when data are dragged over a target and the **OLEDropMode** property of the drop target was set to **vcOLEDropManual**.

	Data Type	Explanation
Parameter:		
⇨ data	DataObject	Object data passed
⇔ effect	Long	Value specifying the action to be performed when the user drops the selection on it. This allows the source to take the appropriate action (such as giving visual feedback).

	Possible Values: vcDropEffectCopy 1 vcDropEffectMove 2 vcDropEffectNone 0	 Drop results in a copy of data from the source to the target. The original data is unaltered by the drag operation. Drop results in data being moved from the source to the target. The source should remove the data from itself after the move. Target cannot accept the data.
⇨ button	Integer	Indicates the mouse button(s) pressed: **1** represents the left button, **2** is the right button, and the middle button is represented by **4**.
	Possible Values:	Data field index
⇨ shift	Integer	Number that indicates which one of the **Shift**, **Ctrl**, and **Alt** keys was pressed. **1** corresponds to the Shift key, **2** to the Ctrl key and **4** to the Alt key. Some, all, or none of the numbers may have been set, indicating that some, all, or none of the keys are depressed, respectively. When more than one key is in depressed state, their values add up. For example, if both the Ctrl and Alt keys are depressed, the value of **shift** would be "6".
	Possible Values:	Data field index
⇨ x	Long	X coordinate of the mouse cursor
⇨ y	Long	Y coordinate of the mouse cursor
⇦ state	OLEDragStateEnum	Drag state
	Possible Values: vcEnter 0 vcLeave 1 vcOver 2	 Object of the source control reaches the target. Object of source control is dragged out of the target. Object of the source control has moved from one position in the target to another.

OLEGiveFeedback

Event of VcGantt

This event occurs after every OLEDragOver event on the target component. OLEGiveFeedback allows the source component to provide visual feedback to the user, such as changing the mouse cursor to indicate what will happen if the user drops the object, or provide visual feedback on the selection (in the source component) to indicate what will happen.

	Data Type	**Explanation**
Parameter:		
⇨ effect	Long	Set by the target component in the OLEDragOver event specifying the action to be performed if the user drops the selection on it. This allows the source to take the appropriate action (such as giving visual feedback).

	Possible Values: vcDropEffectCopy 1 vcDropEffectMove 2 vcDropEffectNone 0	 Drop results in a copy of data from the source to the target. The original data is unaltered by the drag operation. Drop results in data being moved from the source to the target. The source should remove the data from itself after the move. Target cannot accept the data.
⇦ defaultCursors	Boolean **Possible Values:**	Determines whether the control uses the default mouse cursor provided by the component (true), or uses a user-defined mouse cursor (false). Group invisible/visible group nodes are/are not visible

OLESetData

Event of VcGantt

This event occurs on a source component when a target component performs the **GetData** method on the source's DataObject object, but a format for data has not been defined.

	Data Type	Explanation
Parameter:		
⇨ data	DataObject	Object data passed
⇨ dataFormat	Integer **Possible Values:**	Specifies the format of the data that the target component is requesting. The source component uses this value to determine what to fill the DataObject object. Data field index

OLEStartDrag

Event of VcGantt

This event occurs at the source when the VARCHART XGantt control initiates an OLE Drag & Drop operation when the **OLEDragMode** property is set to **vcOLEAutomatic**.

This event specifies the data formats and drop effects that the source component supports. It can also be used to insert data into the DataObject object.

The source component should use the logical **Or** operator against the supported values and place the result in the **allowedEffect** parameter. The target component can use this value to determine the appropriate action (and what the appropriate user feedback should be).

You should defer putting data into the **DataObject** until the target component requests it. This allows the source component to save time by not loading multiple data formats. When the target performs the **GetData** method on the DataObject, the source's **OLESetData** event will occur if the requested data are not contained in the **DataObject**. At this point, the data can be loaded into the **DataObject**, which will in turn provide the data to the target.

If the user does not load any formats into the **DataObject**, then the drag&drop operation is canceled.

	Data Type	Explanation
Parameter:		
⇨ data	DataObject	Object data passed
⇨ effect	Long	Set by the target component in the OLEDragOver event specifying the action to be performed if the user drops the selection on it. This allows the source to take the appropriate action (such as giving visual feedback).
	Possible Values:	
	vcDropEffectCopy 1	Drop results in a copy of data from the source to the target. The original data is unaltered by the drag operation.
	vcDropEffectMove 2	Drop results in data being moved from the source to the target. The source should remove the data from itself after the move.
	vcDropEffectNone 0	Target cannot accept the data.

OnBoxCreate

Event of VcGantt

This event occurs when the user creates a box interactively.

The data passed by this event can be read, but must not be modified. For modifying them please use the event **OnBoxCreateComplete**.

By setting the return status the create operation can be inhibited.

	Data Type	Explanation
Parameter:		
⇨ XOffset	Long	X position of the box
⇨ YOffset	Long	Y position of the box
⇨ Width	Long	Width of the box
⇨ Height	Long	Height of the box
⇔ returnStatus	Variant	Return status
	Possible Values: vcRetStatFalse 0 vcRetStatOK 1	 The box will be created. The box will not be created.

Example Code

```
Private Sub VcGantt1_OnBoxCreate(ByVal XOffset As Long, ByVal YOffset As Long,
ByVal Width As Long, ByVal Height As Long, returnStatus As Variant)
   MsgBox "XOffset " & XOffset & " - " & "YOffset " & YOffset
End Sub
```

OnBoxCreateComplete

Event of VcGantt

This event occurs when the interactive creation of a box is completed. The box object is returned.

	Data Type	Explanation
Parameter:		
⇨ box	VcBox	Box created

Example Code

```
Private Sub VcGantt1_OnBoxCreateComplete(ByVal box As VcGanttLib.VcBox)
    box.LineColor = RGB(255, 0, 0)
End Sub
```

OnBoxLClick

Event of VcGantt

This event occurs when the user clicks the left mouse button on a box. The box object hit and the position of the mouse (x,y-coordinates) are handed over as parameters.

	Data Type	Explanation
Parameter:		
⇨ box	VcBox	Box hit

⇨ x	Long	X coordinate of the mouse cursor
⇨ y	Long	Y coordinate of the mouse cursor
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcGantt1_OnBoxLClick(ByVal box As VcGanttLib.VcBox, _
                  ByVal x As Long, ByVal y As Long, returnStatus As Variant)

    Text1.Text = box.FieldText(1)

End Sub
```

OnBoxLDblClick

Event of VcGantt

This event occurs when the user double-clicks the left mouse button on a box. The VcBox object hit and the mouse position (x,y-coordinates) are returned.

	Data Type	Explanation
Parameter:		
⇨ box	VcBox	Box hit
⇨ x	Long	X coordinate of the mouse cursor
⇨ y	Long	Y coordinate of the mouse cursor
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcGantt1_OnBoxLDblClick(ByVal box As VcGanttLib.VcBox, _
                  ByVal x As Long, ByVal y As Long, returnStatus As Variant)
    box.FieldText(0) = Text1.Text
End Sub
```

OnBoxModify

Event of VcGantt

This event occurs when the user has modified a box interactively. The modified VcBox object and the modification type are passed as parameters.

The data passed by this event can be read, but must not be modified. For modifying them please use **OnBoxModifyComplete**.

By setting the return status the modification can be inhibited.

	Data Type	Explanation
Parameter:		
⇨ box	VcBox	Box modified
⇨ modificationType	BoxModificationTypeEnum	Modification type
	Possible Values:	
	vcBMTAnchoringModified 16	Anchoring of the box modified
	vcBMTAnything 1	any modification
	vcBMTNothing 0	no modification
	vcBMTSizeModified 8	Size of the box modified
	vcBMTTextModified 4	text modified
	vcBMTXYOffsetModified 2	Offset modified
⇔ returnStatus	Variant	Return status
	Possible Values:	
	vcRetStatFalse 0	The modification will be revoked.
	vcRetStatOK 1	The modification will be accepted.

Example Code

```
Private Sub VcGantt1_OnBoxModify(ByVal box As VcGanttLib.VcBox, _
                  ByVal modificationType As _
                  VcGanttLib.BoxModificationTypeEnum, _
                  returnStatus As Variant)

Select Case modificationType
    Case vcBMTAnything: MsgBox "Box modification"
    Case vcBMTXYOffsetModified: MsgBox "Offset changed"
    Case vcBMTTextModified: MsgBox "Box field text changed"
End Select

End Sub
```

OnBoxModifyCompleteEx

Event of VcGantt

This event occurs when the modification of the box is finished. The modified VcBox object and the modification type are passed as parameters.

	Data Type	Explanation
Parameter:		
⇨ modificationType	BoxModificationTypeEnum	Modification type
	Possible Values:	
	vcBMTAnchoringModified 16	Anchoring of the box modified
	vcBMTAnything 1	any modification
	vcBMTNothing 0	no modification
	vcBMTSizeModified 8	Size of the box modified
	vcBMTTextModified 4	text modified
	vcBMTXYOffsetModified 2	Offset modified

Example Code

```
Private Sub VcGantt1_OnBoxModifyCompleteEx(ByVal box As _
                  VcGanttLib.VcBox)
```

```
    MsgBox "The box has been modified."

End Sub
```

OnBoxRClick

Event of VcGantt

This event occurs when the user clicks the right mouse button on a box. The box object and the position of the mouse (x,y-coordinates) are returned. By setting the return status you can inhibit the integrated context menu to pop up and replace it by a context menu of your own at the coordinates delivered.

	Data Type	Explanation
Parameter:		
⇨ box	VcBox	Box hit
⇨ x	Long	X coordinate of the mouse cursor
⇨ y	Long	Y coordinate of the mouse cursor
⇔ returnStatus	Variant	Return status
	Possible Values: vcRetStatNoPopup 4 vcRetStatOK 1	 The context menu will be inhibited. The context menu will appear.

Example Code

```
Private Sub VcGantt1_OnBoxRClick(ByVal box As VcGanttLib.VcBox, _
                  ByVal x As Long, ByVal y As Long, returnStatus As Variant)

    ' Start own popup menu at the current mouse cursor position
    PopupMenu mnuBoxPopup

End Sub
```

OnCalendarGridRClick

Event of VcGantt

This event occurs when the user clicks the right mouse button on a calendar grid. Thecalendar grid object and the mouse position (x,y-coordinates) are returned. By setting the return status you can inhibit the integrated context menu to pop up and replace it by a context menu of your own at the coordinates delivered.

This event will only be triggered if the calendar grid could be identified, i.e. if the calendar grid property **Identifiable** had been set to **True**.

	Data Type	Explanation
Parameter:		
⇨ dateLine	VcDateLine	Calendar grid
⇨ x	Long	X coordinate of the mouse cursor
⇨ y	Long	Y coordinate of the mouse cursor
⇔ returnStatus	Variant	Return status
	Possible Values: vcRetStatNoPopup 4 vcRetStatOK 1	 The context menu will be inhibited. The context menu will appear.

Example Code

```
Private Sub VcGantt1_OnCalendarGridRClick(ByVal group As
VcGanttLib.VcCalendarGrid, _
                                ByVal x As Long, ByVal y As Long, _
                                returnStatus As Variant)

   ' Start own popup menu at the current mouse cursor position
   PopupMenu mnuCalendarGridPopup

   ' Suppress built-in context menu
   returnStatus = vcRetStatNoPopup

End Sub
```

OnCurveLClick

Event of VcGantt

This event occurs when the user clicks the left mouse button on a histogram curve, and before a curve is marked. By setting the VcReturnStatus to **vcRetStatFalse** marking of the curve can be prohibited. In spite of this, the curve values can be modified. At the moment, there is no option to suppress the option of modifying. The curve object hit and the position of the mouse (x,y-coordinates) are returned.

	Data Type	Explanation
Parameter:		
⇨ curve	VcCurve	Curve hit in histogram
⇨ x	Long	X coordinate of the mouse cursor
⇨ y	Long	Y coordinate of the mouse cursor
⇔ returnStatus	Variant	Return status
	Possible Values: vcRetStatFalse 0 vcRetStatOK 1	 The curve will not be marked. The curve will be marked.

Example Code

```
Private Sub VcGantt1_OnCurveLClick(ByVal curve As _
                                   VcGanttLib.VcCurve, ByVal x As Long, _
                                   ByVal y As Long, returnStatus As Variant)
    curve.LineColor = vbRed
End Sub
```

OnCurveLDblClick

Event of VcGantt

This event occurs when the user double-clicks the left mouse button on a histogram curve. The VcCurve object hit and the mouse position (x,y-coordinates) are returned.

	Data Type	Explanation
Parameter:		
⇨ curve	VcCurve	Curve hit in histogram
⇨ x	Long	X coordinate of the mouse cursor
⇨ y	Long	Y coordinate of the mouse cursor
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcGantt1_OnCurveLDblClick(ByVal curve As _
                                      VcGanttLib.VcCurve, ByVal x As Long, _
                                      ByVal y As Long, returnStatus As Variant)
    Call MsgBox("x: " & x & vbCrLf & "y: " & y)
End Sub
```

OnCurveModifyComplete

Event of VcGantt

This event occurs when the modification of the curve is finished.

	Data Type	Explanation
Parameter:		
⇨ curve	VcCurve	Curve modified

OnCurveModifyEx

Event of VcGantt

This event occurs when the user has modified a histogram curve interactively. This is valid for histogram curves generated by API commands

and for histogram curves generated by layers. The modified curve object, the beginning and the end of the section changed, as well as the value that the curve was changed by in y direction are returned. The curve type can be retrieved by the VcCurve property **CurveSource**.

Note: For each modified layer that contributes to the modification of a histogram curve the event **OnCurveModifyEx** will occur twice (once for the start date and once for the end date of the modified curve section).

If you set the return status to **vcRetStatFalse**, the modification will be revoked.

	Data Type	Explanation
Parameter:		
⇨ curve	VcCurve	Curve modified
⇨ date1	Date/Time	Beginning of the curve section changed
⇨ date2	Date/Time	End of the curve section changed
⇨ increment	Long	Value that the curve was changed by in y direction
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcGantt1_OnCurveModifyEx(ByVal curve As _
                                VcGanttLib.VcCurve, ByVal date1 As Date, _
                                ByVal date2 As Date, ByVal increment As Long, _
                                returnStatus As Variant)
Select Case curve.CurveSource
    Case 1
        MsgBox "The curve is calculated from layers." & vbCrLf _
                    & "Increment: " & increment & vbCrLf  _
                    & "Changed start date: " & date1 & vbCrLf _
                    & "Changed end date: " & date2
    Case 3
        MsgBox "Curve set via API." & vbCrLf _
                    & "Increment: " & increment & vbCrLf _
                    & "Changed start date: " & date1 & vbCrLf _
                    & "Changed end date: " & date2
End Select
End Sub
```

OnCurveModifyEx2

Event of VcGantt

This event occurs when the user has modified a histogram curve interactively. This is valid for histogram curves generated by API commands and for histogram curves generated by layers. The modified curve object, the beginning and the end of the section changed, as well as the value that the

curve was changed by in y direction are returned. The curve type can be retrieved by the VcCurve property **CurveSource**.

If you set the return status to **vcRetStatFalse**, the modification will be revoked.

Please note: For each modified layer that contributes to the modification of a histogram curve the event **OnCurveModifyEx** will occur twice (once for the start date and once for the end date of the modified curve section).

Please note: Compared to the event **OnCurveModifyEx**, this event allows the parameter **increment** to be a floating point number.

	Data Type	Explanation

Example Code

```
Private Sub VcGantt1_OnCurveModifyEx(ByVal curve As _
                                VcGanttLib.VcCurve, ByVal date1 As Date, _
                                ByVal date2 As Date, ByVal increment As Double,
_
                                returnStatus As Variant)
Select Case curve.CurveSource
    Case 1
        MsgBox "The curve is calculated from layers." & vbCrLf _
                    & "Increment: " & increment & vbCrLf  _
                    & "Changed start date: " & date1 & vbCrLf _
                    & "Changed end date: " & date2
    Case 3
        MsgBox "Curve set via API." & vbCrLf _
                    & "Increment: " & increment & vbCrLf _
                    & "Changed start date: " & date1 & vbCrLf _
                    & "Changed end date: " & date2
End Select
End Sub
```

OnCurveModifyExAsString

Event of VcGantt

This event occurs after a histogram curve was modified interactively. The event applies to histogram curves generated by API calls as well as to histogram curves generated by layers. The modified curve object, the beginning and the end of the section modified, and the value by which the curve was modified in y direction are returned. The curve type can be retrieved by the VcCurve property **CurveSource**.

The date string has the invariable format "DD.MM.YYYY;hh:mm:ss;".

Note: For a modified layer that contributes to the modification of a curve, the event **OnCurveModifyExAsString** will occur twice: once for the start date and once for the end date of the modified curve section.

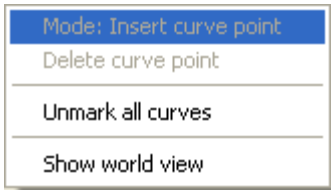
If you set the return status to **vcRetStatFalse**, the modification will be revoked.

	Data Type	Explanation
Parameter:		
⇨ curve	VcCurve	Curve modified
⇨ date1	String **Possible Values:**	Beginning of the curve section changed Name of the color map
⇨ date2	String **Possible Values:**	End of the curve section changed Name of the color map
⇨ increment	Long	Value that the curve was changed by in y direction
⇔ returnStatus	Variant	Return status

OnCurveRClick

Event of VcGantt

This event occurs when the user clicks the right mouse button on a curve. The curve object and the position of the mouse (x,y-coordinates) are returned. By setting the return status you can inhibit the integrated context menu to pop up and replace it by a context menu of your own at the coordinates delivered.



Above: integrated context menu

	Data Type	Explanation
Parameter:		
⇨ curve	VcCurve	Curve hit
⇨ x	Long	X coordinate of the mouse cursor

⇨ y	Long	Y coordinate of the mouse cursor
⇔ returnStatus	Variant	Return status
	Possible Values: vcRetStatNoPopup 4 vcRetStatOK 1	 The context menu will be inhibited. The context menu will appear.

Example Code

```
Private Sub VcGantt1_OnCurveRClick(ByVal curve As _
                                   VcGanttLib.VcCurve, ByVal x As Long, _
                                   ByVal y As Long, returnStatus As Variant)

    'Start own popup menu at the current mouse cursor position
    PopupMenu mnuCurvePopup

    ' Switch off the built-in context menu
    returnStatus = vcRetStatNoPopup

End Sub
```

OnDataRecordCreate

Event of VcGantt

This event occurs when the user creates an object that generates a data record. The generated data record object is returned, so that the data can be validated.

The data passed by this event can be read, but must not be modified. For modifying them please use the event **VcDataRecordCreateComplete**.

By setting the return status the create operation can be inhibited.

If a link or a node was created, you can in addition react to the analogous link or node event and check additional graphical data (see **OnNodeCreate** and **OnLinkCreate**).

	Data Type	Explanation
Parameter:		
⇨ node	VcNode	Data record created
⇔ returnStatus	Variant	Return status
	Possible Values: vcRetStatFalse 0 vcRetStatOK 1	 The data record will be created. The data record will not be created.

Example Code

```
Private Sub VcGantt1_OnDataRecordCreate(ByVal node As VcGanttLib.VcDataRecord, _
                                   returnStatus As Variant)
```

```
    'Show own "Edit" dialog for the new data record
    '(EditNewDataRecord attribute must be set to off!)
    On Error GoTo CancelError
    frmEditDialog.Show Modal, Me

    addDataRecord dataRecord.AllData

    Exit Sub

CancelError:
    returnStatus = vcRetStatFalse

End Sub
```

OnDataRecordCreateComplete

Event of VcGantt

This event occurs when the interactive creation of a data record is completed. The data record object, the creation type (**vcDataRecordCreated** and **vcDataRecordCreatedByResourceScheduling** only) and the information whether the data record created is the only one or the last one of a data record collection (momentarily always **True**) are returned, so that depending data can be validated.

If a link or a node was created, you can in addition react to the analogous link or node event and verify additional graphical data (s. events **OnNodeCreate-Complete** and **OnLinkCreateComplete**).

	Data Type	**Explanation**
Parameter:		
⇨ node	VcNode	Data record created
⇨ creationType	CreationTypeEnum	Creation type
	Possible Values:	
	vcDataRecordCreated 6	Data record created by interaction
	vcDataRecordCreatedByResourceScheduling 5	Data record automatically created by resource scheduling
	vcLinkCreated 2	Link created by linking two nodes
	vcNodeCreated 1	node created via mouse-click
⇨ isLastNodeInSeries	Boolean	**True**:The data record created is the only one or the last one of a data record collection.
		False:The data record created is not the only one or the last one of a data record collection.
	Possible Values:	
		Group invisible/visible

		group nodes are/are not visible

Example Code

```
Private Sub VcGantt1_OnDataRecordCreateComplete(ByVal dataRecord As _
                                VcGanttLib.VcDataRecord, ByVal creationType As
_
                                VcGanttLib.CreationTypeEnum, _
                                ByVal isLastDataRecordInSeries As Boolean)
   addDataRecord dataRecord.AllData
End Sub
```

OnDataRecordDelete

Event of VcGantt

This event occurs when a user deletes an object by the context menu if the object was based on a data record. The data record object to be deleted is returned, so that you can still verify its data and inhibit the deletion on a negative result by setting the return status.

	Data Type	Explanation
Parameter:		
⇨ node	VcNode	Data record deleted
⇔ returnStatus	Variant	Return status
	Possible Values: vcRetStatFalse 0 vcRetStatOK 1	 The data record will not be deleted. The data record will be deleted.

Example Code

```
Private Sub VcGantt1_OnDataRecordDelete(ByVal node As VcGanttLib.VcNode, _
                                returnStatus As Variant)
  'deny the deletion of the last data record in the chart
    If VcGantt1.DataRecordCollection.Count = 1 Then
        returnStatus = vcRetStatFalse
        MsgBox ("The last data record cannot be deleted.")
    End If
End Sub
```

OnDataRecordDeleteComplete

Event of VcGantt

This event occurs when the deletion of an object based on a data record is completed. The data record and the information whether the deleted data record is the only one or the last one of a data record collection are returned, so that depending data can be validated.

If a link or a node was deleted, you can in addition react to the analogous link or node event and verify additional graphical data (s. **OnNodeDelete-Complete** and **OnLinkDeleteComplete**).

	Data Type	Explanation
Parameter:		
⇨ node	VcNode	Data record deleted
⇨ isLastNodeInSeries	Boolean	**True**:The data record deleted is the only one or the last one of a data record collection. **False**:The data record deleted is not the only one or the last one of a data record collection.
	Possible Values:	Group invisible/visible group nodes are/are not visible

OnDataRecordModify

Event of VcGantt

This event occurs after an interactive modification of an object that is based on a data record. The modified VcDataRecord object and the modification type are returned.

The data passed by this event can be read, but must not be modified. For modifying them please use the event **OnDataRecordModifyComplete**.

By setting the return status the modification can be inhibited.

	Data Type	Explanation
Parameter:		
⇨ dataRecord	VcBox	Box modified
⇨ modificationType	ModificationTypeEnum	Modification type
	Possible Values: vcAnything 1 vcChangedGroup 16 vcEndModified 4 vcHierarchyModified 64 vcModifiedByResourceScheduling 128 vcModifiedBySchedule 32 vcMoved 8 vcNothing 0 vcStartModified 2	 modification type not determined group of the node changed The end date of the node has changed. Hierarchy of the nodes was changed Modification by resource scheduling (occurs with data records only) Modification by new date calculation Object was moved no modification The start date of the node changed
⇔ returnStatus	Variant	Return status
	Possible Values:	

vcRetStatFalse 0	The modification will be revoked.
vcRetStatOK 1	The modification will be accepted.

OnDataRecordModifyComplete

Event of VcGantt

This event occurs when the modification of the data record is finished.

	Data Type	Explanation
Parameter:		
⇨ dataRecord	VcDataRecord	Data record modified

Example Code

```
Private Sub VcGantt1_OnDataRecordModifyComplete(ByVal box As _
                  VcGanttLib.VcBox)

    MsgBox "The data record has been modified."

End Sub
```

OnDataRecordNotFound

Event of VcGantt

This event occurs if a depending data record was not found. The index of the field of the current data record, which holds the key to the depending data record, is returned and thus offers some information on the data record not found.

	Data Type	Explanation
Parameter:		
⇦ index	Long	Index of the field that contains the key of the depending data record

OnDateLineModify

Event of VcGantt

This event occurs when the user has modified a date line interactively. The modified VcDateLine object is passed as a parameter.

By setting the return status the modification can be inhibited.

	Data Type	Explanation
Parameter:		
⇨ dateLine	VcDateLine	Date line
⇔ returnStatus	Variant	Return status
	Possible Values: vcRetStatFalse 0 vcRetStatOK 1	 The modification will be revoked. The modification will be accepted.

Example Code

```
Private Sub VcGantt1_OnDateLineModify(ByVal dateLine As _
                                VcGanttLib.VcDateLine, _
                                returnStatus As Variant)
    MsgBox dateLine.Date
End Sub
```

OnDateLineRClick

Event of VcGantt

This event occurs when the user clicks the right mouse button on a date line. The date line object and the position of the mouse (x,y-coordinates) are captured and returned. By setting the return status you can inhibit the integrated context menu to pop up and replace it by a context menu of your own at the coordinates delivered.

	Data Type	Explanation
Parameter:		
⇨ dateLine	VcDateLine	Date line
⇨ x	Long	X coordinate of the mouse cursor
⇨ y	Long	Y coordinate of the mouse cursor
⇔ returnStatus	Variant	Return status
	Possible Values: vcRetStatNoPopup 4 vcRetStatOK 1	 The context menu will be inhibited. The context menu will appear.

Example Code

```
Private Sub VcGantt1_OnDateLineRClick(ByVal dateLine As _
                                VcGanttLib.VcDateLine, ByVal x As Long, _
                                ByVal y As Long, returnStatus As Variant)

    MsgBox dateLine.Name

End Sub
```

OnDeleteCurvePoint

Event of VcGantt

This event occurs when the user deletes a curve point of an histogram curve set by the API. It returns the histogram curve, the date and the y value of the deleted curve point. By setting the return status the deleting operation can be inhibited.

Please Note: The event **OnDeleteCurvePointEx** lets you pass floating point numbers as y values.

	Data Type	Explanation
Parameter:		
⇨ curve	VcCurve	Histogram curve hit
⇨ pointDate	Date	Date of the deleted curve point
⇨ value	Long	Y value of the deleted curve point
⇔ returnStatus	Variant	Return status
	Possible Values: vcRetStatFalse 0 vcRetStatOK 1	 The curve point will not be deleted. The curve point will be deleted.

Example Code

```
Private Sub VcGantt1_OnDeleteCurvePoint(ByVal curve As VcGanttLib.VcCurve, _
                                ByVal pointDate As Date, ByVal value As Long, _
                                returnStatus As Variant)
    If MsgBox("Do you want to delete this curve point (date:" & pointDate & _
                                ",  y value: " & value & ")?", vbYesNo, _
                                "Deleting curve point") = vbNo Then
    returnStatus = vcRetStatFalse
    End If
End Sub
```

OnDeleteCurvePointEx

Event of VcGantt

This event occurs when the user deletes a curve point of an histogram curve set by the API. It returns the histogram curve, the date and the y value of the deleted curve point. By setting the return status the deleting operation can be inhibited.

Please note: Compared to the event **OnDeleteCurvePoint**, this event allows to assign floating point numbers to the y value.

	Data Type	Explanation
Parameter:		
⇨ curve	VcCurve	Histogram curve hit
⇨ pointDate	Date	Date of the deleted curve point
⇨ value	Double	Y value of the deleted curve point
⇔ returnStatus	Variant	Return status
	Possible Values: vcRetStatFalse 0 vcRetStatOK 1	 The curve point will not be deleted. The curve point will be deleted.

Example Code

```
Private Sub VcGantt1_OnDeleteCurvePointEx(ByVal curve As VcGanttLib.VcCurve, _
                                ByVal pointDate As Date, ByVal value As Double,
_
                                returnStatus As Variant)
    If MsgBox("Do you want to delete this curve point (date:" & pointDate & _
                                ",  y value: " & value & ")?", vbYesNo, _
                                "Deleting curve point") = vbNo Then
    returnStatus = vcRetStatFalse
    End If
End Sub
```

OnDiagramLClick

Event of VcGantt

This event occurs when the user clicks the left mouse button on the diagram in an empty space. The position of the mouse (x,y-coordinates) is returned.

	Data Type	Explanation
Parameter:		
⇨ x	Long	X coordinate of the mouse cursor
⇨ y	Long	Y coordinate of the mouse cursor
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcGantt1_OnDiagramLClick(ByVal x As Long, _
                                ByVal y As Long, returnStatus As Variant)
    MsgBox "x: " & x & vbNewLine & "y: " & y
End Sub
```

OnDiagramLDblClick

Event of VcGantt

This event occurs when the user double-clicks the left mouse button on the diagram in an empty space. The position of the mouse (x,y-coordinates) is returned.

	Data Type	Explanation
Parameter:		
⇨ x	Long	X coordinate of the mouse cursor
⇨ y	Long	Y coordinate of the mouse cursor
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcGantt1_OnDiagramLDblClick(ByVal x As Long, _
                                ByVal y As Long, returnStatus As Variant)

    VcGantt1.Zoom (90)

End Sub
```

OnDiagramRClick

Event of VcGantt

This event occurs when the user clicks the right mouse button on a diagram in an empty space. The position of the mouse (x,y-coordinates) is returned. By setting the return status you can inhibit the integrated context menu to pop up and replace it by a context menu of your own at the coordinates delivered.

Above: integrated context menu

	Data Type	Explanation
Parameter:		
⇨ x	Long	x Coordinate of the mouse cursor
⇨ y	Long	Y coordinate of the mouse cursor
⇔ returnStatus	Variant	Return status
	Possible Values: vcRetStatNoPopup 4 vcRetStatOK 1	 The context menu will be inhibited. The context menu will appear.

Example Code

```
Private Sub VcGantt1_OnDiagramRClick(ByVal x As Long, ByVal y As Long, _
                                 returnStatus As Variant)

   ' Start own popup menu at the current mouse cursor position
    PopupMenu mnuDiagramPopup

   ' Switch off the built-in context menu
    returnStatus = vcRetStatNoPopup

End Sub
```

OnGroupDelete

Event of VcGantt

This event occurs when the user deletes a group. It returns the group object. By setting the return status the deleting operation can be inhibited. The user can delete only empty groups that do not contain any elements.

	Data Type	Explanation
Parameter:		
⇨ group	VcGroup	Group deleted
⇔ returnStatus	Variant	Return status
	Possible Values: vcRetStatFalse 0 vcRetStatOK 1	 The group will not be deleted. The group will be deleted.

Example Code

```
Private Sub VcGantt1_OnGroupDelete(ByVal group As VcGanttLib.VcGroup, _
                                 returnStatus As Variant)
    If group.Name = "A" Then
        returnStatus = vcRetStatFalse
        MsgBox ("Group A cannot be deleted.")
    End If
End Sub
```

OnGroupLClick

Event of VcGantt

This event occurs when the user clicks the left mouse button on a group heading in the table. The group object and the mouse position (x,y-coordinates) are returned.

	Data Type	Explanation
Parameter:		
⇨ group	VcGroup	Group hit
⇨ x	Long	X coordinate of the mouse cursor
⇨ y	Long	Y coordinate of the mouse cursor
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcGantt1_OnGroupLClick(ByVal group As VcGanttLib.VcGroup, _
                                   ByVal x As Long, ByVal y As Long, _
                                   returnStatus As Variant)
    MsgBox group.SubGroups.Count
End Sub
```

OnGroupLDblClick

Event of VcGantt

This event occurs when the user double-clicks the left mouse button on a group heading in the table. The group object and the mouse position (x,y-coordinates) are returned.

	Data Type	Explanation
Parameter:		
⇨ group	VcGroup	group hit
⇨ x	Long	X coordinate of the mouse cursor
⇨ y	Long	Y coordinate of the mouse cursor
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcGantt1_OnGroupLDblClick(ByVal group As VcGanttLib.VcGroup, _
                                      ByVal x As Long, ByVal y As Long, _
                                      returnStatus As Variant)

   MsgBox group.Name

End Sub
```

OnGroupModify

Event of VcGantt

This event occurs when a user interactively modifies a group. The group object, the type of modification and the return status are returned.

The data passed by this event can be read, but must not be modified. For modifying them please use **OnDataLineModifyComplete**.

By setting the return status the modification can be inhibited.

	Data Type	Explanation
Parameter:		
⇨ Group	VcGroup	Group modified
⇨ modificationType	GroupModificationTypeEnum	Type of modification
	Possible Values:	
	vcGMTAnything 1	Modification type **not determined**
	vcGMTEndModified 16	The end date was changed.
	vcGMTMinusPressed 2	Modification type **Minus symbol clicked on**
	vcGMTMoved 32	Object was moved
	vcGMTNothing 0	Modification type **nothing**
	vcGMTPlusPressed 4	Modification type **Plus symbol clicked on**
	vcGMTStartModified 8	The start date was changed
⇔ returnStatus	Variant	Return status
	Possible Values:	
	vcRetStatFalse 0	The modification will be revoked.
	vcRetStatOK 1	The modification will be accepted.

Example Code

```
Private Sub VcGantt1_OnGroupModify(ByVal group As VcGanttLib.VcGroup, _
                                   ByVal modificationType As _
                                   VcGanttLib.GroupModificationTypeEnum, _
                                   returnStatus As Variant)

    Select Case modificationType
       Case vcGMTNothing
            MsgBox "No modification"
       Case vcGMTAnything
            MsgBox "Any modification"
       Case vcGMTMinusPressed
            MsgBox "Collapsing group: " & group.Name
       Case vcGMTPlusPressed
            MsgBox "Expanding group: " & group.Name
     End Select

End Sub
```

OnGroupModifyComplete

Event of VcGantt

This event occurs when the modification of the group is finished.

	Data Type	Explanation
Parameter:		
⇨ Group	VcGroup	Group modified

Example Code

```
Private Sub VcGantt1_OnGroupModifyComplete(ByVal group As VcGanttLib.VcGroup)
    MsgBox "The group has been modified."
End Sub
```

OnGroupModifyEx

Event of VcGantt

This event occurs when the user modifies a group. The data of the node before and after the modification are passed. By the **modificationType** parameter you get further information of the type of modification. If you set the returnStatus to **vcRetStatFalse**, the modification will be revoked.

This event should be used only for reading data from the current group, but not for modifying one. For modifying data please use **OnGroupModify-Complete**.

	Data Type	Explanation
Parameter:		
⇨ oldgroup	VcGroup	Group before the modification
⇨ group	VcGroup	Group to be modified
⇨ modificationType	GroupModificationTypeEnum	Type of modification
	Possible Values: vcGMTAnything 1 vcGMTEndModified 16 vcGMTMinusPressed 2 vcGMTMoved 32 vcGMTNothing 0 vcGMTPlusPressed 4 vcGMTStartModified 8	 Modification type **not determined** The end date was changed. Modification type **Minus symbol clicked on** Object was moved Modification type **nothing** Modification type **Plus symbol clicked on** The start date was changed
⇦ returnStatus	Variant	Return status

OnGroupRClick

Event of VcGantt

This event occurs when the user clicks the right mouse button on a group title in the table. The group object and the mouse position (x,y-coordinates) are returned. By setting the return status you can inhibit the integrated context menu to pop up and replace it by a context menu of your own at the coordinates delivered.

	Data Type	Explanation
Parameter:		
⇨ group	VcGroup	Group hit
⇨ x	Long	X coordinate of the mouse cursor
⇨ y	Long	Y coordinate of the mouse cursor
⇔ returnStatus	Variant	Return status
	Possible Values: vcRetStatNoPopup 4 vcRetStatOK 1	 The context menu will be inhibited. The context menu will appear.

Example Code

```
Private Sub VcGantt1_OnGroupRClick(ByVal group As VcGanttLib.VcGroup, _
                                   ByVal x As Long, ByVal y As Long, _
                                   returnStatus As Variant)

   ' Start own popup menu at the current mouse cursor position
   PopupMenu mnuGroupPopup

   ' Suppress built-in context menu
   returnStatus = vcRetStatNoPopup

End Sub
```

OnGroupsMark

Event of VcGantt

This event occurs when the user selects groups for marking or when he unmarks marked groups by a click into the empty diagram. If the user marked groups, the GroupCollection contains the groups selected by the most recent marking action. If the user unmarked groups by a click into the empty diagram, the group collection will be empty.

If you set the return status to **vcRetStatFalse**, you have to mark or unmark groups yourself.

The data passed by this event can be read, but must not be modified. For modifying them please use **OnGroupsMarkComplete**.

	Data Type	Explanation
Parameter:		
⇨ groupCollection	VcGroupCollection	GroupCollection that contains the groups selected by the user. If the user clicked in the diagram, the GroupCollection is empty.
⇨ button	Integer	Indicates in which way the buttons were marked: **0**: by keyboard, **1**: left mouse button pressed, **2**: right mouse button pressed, **4**: mouse button pressed
	Possible Values:	Data field index
⇨ shift	Integer	Number that indicates which one of the **Shift**, **Ctrl**, and **Alt** keys was pressed. **1** corresponds to the Shift key, **2** to the Ctrl key and **4** to the Alt key. Some, all, or none of the numbers may have been set, indicating that some, all, or none of the keys are depressed, respectively. When more than one key is in depressed state, their values add up. For example, if both the Ctrl and Alt keys were depressed, the value of **shift** would equal "6".
	Possible Values:	Data field index
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcGantt1_OnNodesMarkEx(ByVal GroupCollection As _
                                   VcGanttLib.VcGroupCollection, ByVal button _
                                   As Integer, ByVal shift As Integer, _
                                   returnStatus As Variant)

If MsgBox("Mark this group?", vbYesNo, "Marking groups") = _
    vbNo Then returnStatus = vcRetStatFalse

End Sub
```

OnGroupsMarkComplete

Event of VcGantt

This event occurs when the operation of marking or unmarking groups is finished.

	Data Type	Explanation
Parameter:		
⇦ (no parameter)		No parameter

Example Code

```
Private Sub VcGantt1_OnGroupsMarkComplete()
```

```
    MsgBox "Groups have been marked successfully."
End Sub
```

OnHelpRequested

Event of VcGantt

This event occurs if the user presses the F1 key on a dialog at run time. The application can invoke its own help system, to offer information specific to the dialog and to the application.

	Data Type	Explanation
Parameter:		
⇦ dialogType	DialogTypeEnum	Dialog for which help was requested
	Possible Values:	
	vcEditDataRecordDialog 5400	Help was requested for the **Edit Data Record** dialog.
	vcEditTimeScaleDialog 5409	Help was requested for the **Edit Time Scale** dialog.
	vcPageSetupDialog 4097	Help was requested for the **Page Set Up** dialog.
	vcPrintPreviewDialog 4096	Help was requested for the **Print Preview** dialog.

OnHistogramLClick

Event of VcGantt

This event occurs when the user clicks the left mouse button on a histogram. The histogram object and the mouse position (x,y-coordinates) are returned.

	Data Type	Explanation
Parameter:		
⇨ histogram	VcHistogram	Histogram hit
⇨ x	Long	X coordinate of the mouse cursor
⇨ y	Long	Y coordinate of the mouse cursor
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcGantt1_OnHistogramLClick(ByVal Histogram As _
                                VcGanttLib.VcHistogram, ByVal x As Long, _
                                ByVal y As Long, returnStatus As Variant)

    Call MsgBox("Histogram: " & Histogram.Name & vbCrLf & _
                      "x: " & x & vbCrLf & "y: " & y)

End Sub
```

OnHistogramLDblClick

Event of VcGantt

This event occurs when the user double-clicks the left mouse button on a histogram. The histogram object and the mouse position (x,y-coordinates) are returned.

	Data Type	Explanation
Parameter:		
⇨ histogram	VcHistogram	Histogram hit
⇨ x	Long	X coordinate of the mouse cursor
⇨ y	Long	Y coordinate of the mouse cursor
⇔ returnStatus	Variant	Return status

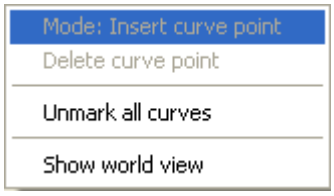
Example Code

```
Private Sub VcGantt1_OnHistogramLDblClick(ByVal Histogram As _
                                VcGanttLib.VcHistogram, ByVal x As Long, _
                                ByVal y As Long, returnStatus As Variant)

MsgBox Histogram.Name

End Sub
```

OnHistogramRClick

Event of VcGantt

This event occurs when the user clicks the right mouse button on a histogram. The histogram object and the mouse position (x,y-coordinates) are returned. By setting the return status you can inhibit the integrated context menu to pop up and replace it by a context menu of your own at the coordinates delivered.



Above: integrated context menu

	Data Type	Explanation
Parameter:		
⇨ histogram	VcHistogram	Histogram hit
⇨ x	Long	X coordinate of the mouse cursor

⇨ y	Long	Y coordinate of the mouse cursor
⇔ returnStatus	Variant	Return status
	Possible Values: vcRetStatNoPopup 4 vcRetStatOK 1	 The context menu will be inhibited. The context menu will appear.

Example Code

```
Private Sub VcGantt1_OnHistogramRClick(ByVal Histogram As _
                                VcGanttLib.VcHistogram, ByVal x As Long, _
                                ByVal y As Long, returnStatus As Variant)

    ' Start own popup menu at the current mouse cursor position
    PopupMenu mnuHistogramPopup

    ' Suppress built-in context menu
    returnStatus = vcRetStatNoPopup

End Sub
```

OnHistogramsHeight

Event of VcGantt

This event occurs, when the user modifies the ratio of the diagram height to the histogram height. The collection of the histograms and the diagram / histogram height ratio are returned. If you set the return status to vcRetStat-False, the modification will be revoked.

	Data Type	Explanation
Parameter:		
⇨ histogramCollection	VcHistogramCollection	Histogram collection
⇨ histogramsHeightRatio	Long	Ratio of the height of the histograms to the height of the diagram
⇔ returnStatus	Variant	Return status
	Possible Values: vcRetStatFalse 0 vcRetStatOK 1	 The height will not change. The height will change.

Example Code

```
Private Sub VcGantt1_OnHistogramsHeight(ByVal HistogramCollection As
VcGanttLib.VcHistogramCollection, ByVal histogramsHeightRatio As Long,
returnStatus As Variant)

    If histogramsHeightRatio > 30 Then
        returnStatus = vcRetStatFalse
        VcGantt1.DiagramHistogramHeightRatio = 30
    End If

End Sub
```

OnHistogramsHeightChanged

Event of VcGantt

This event occurs, after the ratio of the diagram height to the histogram height which was modified by the user was changed. The collection of the histograms and the diagram / histogram height ratio are returned.

	Data Type	Explanation
Parameter:		
⇨ histogramCollection	VcHistogramCollection	Histogram collection
⇨ histogramsHeightRatio	Long	Ratio of the height of the histograms to the height of the diagram

Example Code

```
Private Sub VcGantt1_OnHistogramsHeightChanged(ByVal HistogramCollection As
VcGanttLib.VcHistogramCollection, ByVal histogramsHeightRatio As Long)

    VcGantt1.FitHistogramsIntoView

End Sub
```

OnHistogramsHeightModifyEx

Event of VcGantt

This event occurs when the user interactively modifies the height of the histogram. The histogram and the modified diagram/histogram aspect ratio are returned. By setting the return status you can inhibit the modification.

In contrast to the **OnHistogramHeigth** event this event returns the parameter **histogramHeightRatio** as a "Double" value, thus achieving a higher level of accuracy. The use of this event has to be enabled by the **UseHigher-DiagramHistogramHeightRatioPrecision** property or by activating the corresponding option on the **General** property page.

	Data Type	Explanation
Parameter:		
⇨ histogramCollection	VcHistogramCollection	Histogram collection containing all histograms of this Gantt instance
⇨ histogramHeightRatio	Double	Ratio of the total height of the diagram (including histogram) to the height of the histogram
⇔ returnStatus	Variant	Return status

OnInsertCurvePoint

Event of VcGantt

This event occurs when the user has selected the histogram context menu item **Mode: Insert curve point** and then inserts a curve point to a histogram curve generated by API commands. It returns the histogram curve, the date and the y value of the inserted curve point. If you set the returnStatus to **vcRetStatFalse**, the inserting operation will be revoked.

	Data Type	Explanation
Parameter:		
⇨ curve	VcCurve	Curve hit in histogram
⇨ pointDate	Date	Date of the inserted curve point
⇨ value	Long	Y value of the inserted curve point
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcGantt1_OnInsertCurvePoint(ByVal curve As VcGanttLib.VcCurve, _
                                ByVal pointDate As Date, ByVal value As Long, _
                                returnStatus As Variant)
    If MsgBox("Do you want to insert this curve point (date:" & pointDate & ", _
                                y value: " & value & ")?", vbYesNo, _
                                "Deleting curve point") = vbNo Then
    returnStatus = vcRetStatFalse
    End If
End Sub
```

OnInsertCurvePointEx

Event of VcGantt

This event occurs when the user has selected the histogram context menu item **Mode: Insert curve point** and then inserts a curve point to an histogram curve set by API commands. It returns the histogram curve, the date and the y value of the inserted curve point. If you set the returnStatus to **vcRetStatFalse**, the inserting operation will be revoked.

Please note: Compared to the event **OnInsertCurvePointEx**, this event allows the parameter **value** to be a floating point number.

	Data Type	Explanation
Parameter:		
⇨ curve	VcCurve	Curve hit in histogram
⇨ pointDate	Date	Date of the inserted curve point
⇨ value	Double	Y value of the inserted curve point

⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcGantt1_OnInsertCurvePoint(ByVal curve As VcGanttLib.VcCurve, _
                                ByVal pointDate As Date, ByVal value As Long, _
                                returnStatus As Variant)
    If MsgBox("Do you want to insert this curve point (date:" & pointDate & ", _
                                y value: " & value & ")?", vbYesNo, _
                                "Deleting curve point") = vbNo Then
    returnStatus = vcRetStatFalse
    End If
End Sub
```

OnInteractionEndComplete

Event of VcGantt

This event occurs on ending an interaction (LiveUpdate switched on).

	Data Type	Explanation
Parameter:		
⇨ InInteractionMode	InInteractionModeEnum	Mode of interaction
	Possible Values:	
	vcIIMCopyMoveNode 1014	Move copied node
	vcIIMCopyNode 1007	Copy node
	vcIIMCreateLinkChangeSuccessor 1101	Change successor
	vcIIMCreateNodeResizeRightX 1012	Modify start date of layer
	vcIIMCreateResizeObjectContainerWidthHight 1072	Modify size of textbox
	vcIIMDragDropNode 1018	Drag and drop node
	vcIIMDragDropNodeInTable 1019	Move node in table by drag and drop
	vcIIMModifySectionStartDate 1061	Modifiy start date of time scale section
	vcIIMMoveCurvePointX 1052	Move curve point x
	vcIIMMoveCurvePointXandY 1051	Move curve points x and y
	vcIIMMoveCurvePointY 1053	Move curve point y
	vcIIMMoveGroupInDiagram 1100	Group in diagram is moved
	vcIIMMoveGroupInTable 1009	Move group in table
	vcIIMMoveHorValueLine 1031	Move date line horizontally
	vcIIMMoveLayer 1004	Move layer
	vcIIMMoveNode 1001	Move node
	vcIIMMoveNodeInRow 1002	Move node in row
	vcIIMMoveNodeInTable 1008	Move node in table
	vcIIMMoveNodeVertical 1003	Move node vertically
	vcIIMMoveObjectContainer 1073	Move textbox
	vcIIMMoveSash 1026	Move sash
	vcIIMResizeBasicUnitWidth 1062	Modify basic unit width
	vcIIMResizeLeftX 1005	Modify start date of layer
	vcIIMResizeNumericBasicUnitWidth 1063	Modify numeric basic unit width
	vcIIMResizeObjectContainerHeight 1075	Modify height of textbox
	vcIIMResizeObjectContainerWidth 1074	Modify width of text box
	vcIIMResizeObjectContainerWidthHeight 1076	Modify width and height of textbox
	vcIIMResizeRightX 1006	Modify end date of layer

	vcIIMUnKnown -1	Usuallly not returned by eventargs, but can be used e.g. for inidcating a variable as not having been set
	vcIIMvcIIMResizeLeftTableColumnWidth 1041	Modify column width of left table
	vcIIMvcIIMResizeRightTableColumnWidth 1042	Modify column width of right table
⇨ InteractionObject	InteractionObject	Object affected by the interaction
⇨ ObjectType	InteractionType	type of object affected by the interaction

OnInteractionModeChange

Event of VcGantt

This event occurs after having changed the interaction mode in the contextmenu.

By setting the return status to **vcRetStatFalse** the modification will not be applied and the event **OnInteractionModeChangeComplete** will not be triggered.

	Data Type	Explanation
Parameter:		
⇨ NewInteractionMode	InteractionModeEnum	Selected interaction mode
	Possible Values:	
	vcCreateBox 36	Box creating mode
	vcCreateLink 4	Link creating mode
	vcCreateNode 2	Node creating mode
	vcDeleteLink 5	Link deleting mode
	vcDeleteNode 3	Node deleting mode
	vcPanning 6	Panning mode
	vcPointer 0	Select mode
⇨ returnstatus	returnstatus	Return status

OnInteractionModeChangeComplete

Event of VcGantt

This event occurs after having changed the interaction mode in the contextmenu and when the change not having been prevented by setting the return status of the event **OnInteractionModeChange** to **vcRetStatFalse**.

	Data Type	Explanation

OnInteractionObjectChangingComplete

Event of VcGantt

This event occurs when there is no object yet at the beginning of an interaction (LiveUpdate switched on; such as creating nodes or boxes) and as soon as the object has been created internally.

	Data Type	Explanation
Parameter:		
⇨ InInteractionMode	InInteractionModeEnum	Mode of interaction
	Possible Values:	
	vcIIMCopyMoveNode 1014	Move copied node
	vcIIMCopyNode 1007	Copy node
	vcIIMCreateLinkChangeSuccessor 1101	Change successor
	vcIIMCreateNodeResizeRightX 1012	Modify start date of layer
	vcIIMCreateResizeObjectContainerWidthHight 1072	Modify size of textbox
	vcIIMDragDropNode 1018	Drag and drop node
	vcIIMDragDropNodeInTable 1019	Move node in table by drag and drop
	vcIIMModifySectionStartDate 1061	Modifiy start date of time scale section
	vcIIMMoveCurvePointX 1052	Move curve point x
	vcIIMMoveCurvePointXandY 1051	Move curve points x and y
	vcIIMMoveCurvePointY 1053	Move curve point y
	vcIIMMoveGroupInDiagram 1100	Group in diagram is moved
	vcIIMMoveGroupInTable 1009	Move group in table
	vcIIMMoveHorValueLine 1031	Move date line horizontally
	vcIIMMoveLayer 1004	Move layer
	vcIIMMoveNode 1001	Move node
	vcIIMMoveNodeInRow 1002	Move node in row
	vcIIMMoveNodeInTable 1008	Move node in table
	vcIIMMoveNodeVertical 1003	Move node vertically
	vcIIMMoveObjectContainer 1073	Move textbox
	vcIIMMoveSash 1026	Move sash
	vcIIMResizeBasicUnitWidth 1062	Modify basic unit width
	vcIIMResizeLeftX 1005	Modify start date of layer
	vcIIMResizeNumericBasicUnitWidth 1063	Modify numeric basic unit width
	vcIIMResizeObjectContainerHeight 1075	Modify height of textbox
	vcIIMResizeObjectContainerWidth 1074	Modify width of text box
	vcIIMResizeObjectContainerWidthHeight 1076	Modify width and height of textbox
	vcIIMResizeRightX 1006	Modify end date of layer
	vcIIMUnKnown -1	Usuallly not returned by eventargs, but can be used e.g. for inidcating a variable as not having been set
	vcIIMvcIIMResizeLeftTableColumnWidth 1041	Modify column width of left table

	vcIIMvcIIMResizeRightTableColumnWidth 1042	Modify column width of right table
⇨ InteractionObject	InteractionObject	Object affected by the interaction
⇨ ObjectType	InteractionType	type of object affected by the interaction

OnInteractionStartComplete

Event of VcGantt

This event occurs when an interaction is started by pressing the left mouse key (LiveUpdate switched on) and it returns information about the object the mouse has hit (object and object type).

	Data Type	Explanation
Parameter:		
⇨ InInteractionMode	InInteractionModeEnum	Mode of interaction
	Possible Values:	
	vcIIMCopyMoveNode 1014	Move copied node
	vcIIMCopyNode 1007	Copy node
	vcIIMCreateLinkChangeSuccessor 1101	Change successor
	vcIIMCreateNodeResizeRightX 1012	Modify start date of layer
	vcIIMCreateResizeObjectContainerWidthHight 1072	Modify size of textbox
	vcIIMDragDropNode 1018	Drag and drop node
	vcIIMDragDropNodeInTable 1019	Move node in table by drag and drop
	vcIIMModifySectionStartDate 1061	Modifiy start date of time scale section
	vcIIMMoveCurvePointX 1052	Move curve point x
	vcIIMMoveCurvePointXandY 1051	Move curve points x and y
	vcIIMMoveCurvePointY 1053	Move curve point y
	vcIIMMoveGroupInDiagram 1100	Group in diagram is moved
	vcIIMMoveGroupInTable 1009	Move group in table
	vcIIMMoveHorValueLine 1031	Move date line horizontally
	vcIIMMoveLayer 1004	Move layer
	vcIIMMoveNode 1001	Move node
	vcIIMMoveNodeInRow 1002	Move node in row
	vcIIMMoveNodeInTable 1008	Move node in table
	vcIIMMoveNodeVertical 1003	Move node vertically
	vcIIMMoveObjectContainer 1073	Move textbox
	vcIIMMoveSash 1026	Move sash
	vcIIMResizeBasicUnitWidth 1062	Modify basic unit width
	vcIIMResizeLeftX 1005	Modify start date of layer
	vcIIMResizeNumericBasicUnitWidth 1063	Modify numeric basic unit width
	vcIIMResizeObjectContainerHeight 1075	Modify height of textbox
	vcIIMResizeObjectContainerWidth 1074	Modify width of text box
	vcIIMResizeObjectContainerWidthHeight 1076	Modify width and height of textbox
	vcIIMResizeRightX 1006	Modify end date of layer
	vcIIMUnKnown -1	Usuallly not returned by eventargs, but can be used e.g. for inidcating a variable as not having been set

	vcIIMvcIIMResizeLeftTableColumnWidth 1041 vcIIMvcIIMResizeRightTableColumnWidth 1042	Modify column width of left table Modify column width of right table
⇨ InteractionObject	InteractionObject	Object affected by the interaction
⇨ ObjectType	InteractionType	type of object affected by the interaction

OnLegendViewClosed

Event of VcGantt

This event occurs when the legend view popup window is closed.

	Data Type	Explanation
Parameter: ⇦ (no parameter)		

Example Code

```
Private Sub VcGantt1_OnLegendViewClosed()
    MsgBox "Do you want to close the legend view window?", vbOKCancel
End Sub
```

OnLinkCreate

Event of VcGantt

This event occurs when the user creates a link between two nodes. The generated link object is returned, so that a validation and if necessary a data base entry can be made.

The data passed by this event can be read, but must not be modified. For modifying them please use the event **OnLinkCreateComplete**.

By setting the return status the create operation can be inhibited.

	Data Type	Explanation
Parameter:		
⇨ link	VcLink	Link created
⇔ returnStatus	Variant	Return status
	Possible Values: vcRetStatFalse 0 vcRetStatOK 1	 The link will not be created. The link will be created.

Example Code

```
Private Sub VcGantt1_OnLinkCreate(ByVal link As VcGanttLib.VcLink, _
                                  returnStatus As Variant)

   ' Show own "Edit" dialog for the new link
    On Error GoTo CancelError
    frmEditDialog.Show Modal, Me

    ' create a record in the underlying database of the application
    addDataRecord link.AllData

    Exit Sub

CancelError:
    returnStatus = vcRetStatFalse

End Sub
```

OnLinkCreateComplete

Event of VcGantt

This event occurs when the interactive creation of a link is completed. The link object, the creation type (always **VcLinkCreated**) and the information whether the created link is the only link or the last link of a link collection (always **True**) are returned, so that a validation can be made.

	Data Type	Explanation
Parameter:		
⇨ link	VcLink	Link created
⇨ creationType	CreationTypeEnum	Creation type of nodes/links
	Possible Values: vcLinkCreated 2	 Link created by linking two nodes
⇨ isLastLinkInSeries	Boolean	The created link is/is not the only link or the last link of a link collection.
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
Private Sub VcGantt1_OnLinkCreateComplete(ByVal link As VcGanttLib.VcLink, _
                                  ByVal creationType As _
                                  VcGanttLib.CreationTypeEnum, _
                                  ByVal isLastLinkInSeries As Boolean)

   'create a record in the underlying database of the application
   addDataRecord link.AllData

End Sub
```

OnLinkDelete

Event of VcGantt

This event occurs when a user deletes a link by the context menu. The link object to be deleted is returned, so that you can still check for - whatever - conditions and prohibit the deletion on a negative result by setting the return status.

	Data Type	Explanation
Parameter:		
⇨ link	VcLink	Link deleted
⇔ returnStatus	Variant	Return status
	Possible Values: vcRetStatFalse 0 vcRetStatOK 1	 The link will not be deleted. The link will be deleted.

Example Code

```
Private Sub VcGantt1_OnLinkDelete(ByVal link As VcGanttLib.VcLink, _
                                  returnStatus As Variant)
    ' deny deletion of link with a certain predecessor
    If link.PredecessorNode.DataField(0) = "1" Then
        returnStatus = vcRetStatFalse
    End If
End Sub
```

OnLinkDeleteComplete

Event of VcGantt

This event occurs when the deletion of a link is completed. The link object and the information whether the created link is the only link or the last link of a link collection are returned, so that a validation can be made.

	Data Type	Explanation
Parameter:		
⇨ link	VcLink	Link deleted
⇨ isLastLinkInSeries	Boolean	The deleted link is/is not the only link or the last link of a link collection.
	Possible Values:	 Group invisible/visible group nodes are/are not visible

Example Code

```
Private Sub VcGantt1_OnLinkDeleteComplete(ByVal link As VcGanttLib.VcLink, ByVal
isLastLinkInSeries As Boolean)

    MsgBox "The link " & link.AllData & " was deleted."
```

```
End Sub
```

OnLinkLClickCltn

Event of VcGantt

This event occurs when the user clicks the left mouse button on a link or on several overlapping links. The LinkCollection object and the mouse position (x,y-coordinates) are returned.

	Data Type	Explanation
Parameter:		
⇨ linkCltn	VcLinkCollection	LinkCollection object hit
⇨ x	Long	Y coordinate of the mouse cursor
⇨ y	Long	Y coordinate of the mouse cursor
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcGantt1_OnLinkLClickCltn(ByVal linkCltn _
                                As VcGanttLib.VcLinkCollection, _
                                ByVal x As Long, ByVal y As Long, _
                                returnStatus As Variant)
Dim link As VcLink

    ' set certain data field of all links
    For Each link In linkCltn
        link.DataField(2) = "A"
    Next
End Sub
```

OnLinkLDblClickCltn

Event of VcGantt

This event occurs when the user double-clicks the left mouse button on a link or on several overlapping links. The LinkCollection object and the mouse position (x,y-coordinates) are returned.

	Data Type	Explanation
Parameter:		
⇨ linkCltn	VcLinkCollection	LinkCollection object hit
⇨ x	Long	X coordinate of the mouse cursor
⇨ y	Long	Y coordinate of the mouse cursor
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcGantt1_OnLinkLDblClickCltn(ByVal linkCltn As _
                                VcGanttLib.VcLinkCollection, _
                                ByVal x As Long, ByVal y As Long, _
                                returnStatus As Variant)

   ' Edit link in own dialog
    If linkCltn.Count = 1 Then
        On Error GoTo CancelError
        frmEditLinkDialog.setLink linkCltn.FirstLink
        frmEditLinkDialog.Show Modal, Me
    End If

    ' Deny the "Edit Link Data" dialog
    CancelError:
    returnStatus = vcRetStatFalse

End Sub
```

OnLinkRClickCltn

Event of VcGantt

This event occurs when the user clicks the right mouse button on a link or on several overlapping links. The LinkCollection object and the mouse position (x,y-coordinates) are returned. By setting the return status you can inhibit the integrated context menu to pop up and replace it by a context menu of your own at the coordinates delivered.

	Data Type	Explanation
Parameter:		
⇨ linkCltn	VcLinkCollection	LinkCollection object hit
⇨ x	Long	X coordinate of the mouse cursor
⇨ y	Long	Y coordinate of the mouse cursor
⇔ returnStatus	Variant	Return status
	Possible Values: vcRetStatNoPopup 4 vcRetStatOK 1	 The context menu will be inhibited. The context menu will appear.

Example Code

```
Private Sub VcGantt1_OnLinkRClickCltn(ByVal linkCltn As _
                                VcGanttLib.VcLinkCollection, _
                                ByVal x As Long, ByVal y As Long, _
                                returnStatus As Variant)
    ' Start a popup menu at the current mouse cursor position
    PopupMenu mnuLinkPopup

    ' Suppress the built-in context menu
    returnStatus = vcRetStatNoPopup
End Sub
```

OnModifyComplete

Event of VcGantt

This event occurs when data have been modified interactively in the chart, that means it occurs after the following events:

- OnBoxModifyComplete
- OnCurveModifyEx
- OnDeleteCurvePoint
- OnGroupModifyComplete
- OnLinkCreateComplete
- OnLinkDeleteComplete
- OnNodeCreateCompleteEx
- OnNodeDelete
- OnNodeModifyComplete

This event allows you to set a mark in the application that reminds to save the data before closing the program.

	Data Type	Explanation
Parameter:		
⇦ (no parameter)		No parameter

OnMouseDblClk

Event of VcGantt

This event occurs when the user double-clicks a mouse button.

Please also regard the **MouseProcessingEnabled** property.

	Data Type	Explanation
Parameter:		
⇨ button	Integer	indicates the mouse button(s) pressed: **1** represents the left button, **2** is the right button, and the middle button is represented by **4**.
	Possible Values:	Data field index
⇨ Shift	Integer	Number that indicates which one of the **Shift**, **Ctrl**, and **Alt** keys was pressed. **1** corresponds to the Shift key, **2** to the Ctrl key and **4** to the Alt key. Some, all, or none of the numbers may have been set, indicating that some, all, or none of the keys are depressed, respectively. When more than one key is in depressed state, their values add up. For example, if both the Ctrl and Alt keys are depressed, the value of **shift** would be "6".
	Possible Values:	Data field index
⇨ x	Long	X coordinate of the mouse cursor
⇨ y	Long	Y coordinate of the mouse cursor

OnMouseDown

Event of VcGantt

This event occurs when the user clicks a mouse button.

Please also regard the **MouseProcessingEnabled** property.

	Data Type	Explanation
Parameter:		
⇨ button	Integer	Indicates the mouse button(s) pressed: **1** represents the left button, **2** is the right button, and the middle button is represented by **4**.
	Possible Values:	Data field index
⇨ Shift	Integer	Number that indicates which one of the **Shift**, **Ctrl**, and **Alt** keys was pressed. **1** corresponds to the Shift key, **2** to the Ctrl key and **4** to the Alt key. Some, all, or none of the numbers may have been set, indicating that some, all, or none of the keys are depressed, respectively. When more than one key is in depressed state, their values add up. For example, if both the Ctrl and Alt keys are depressed, the value of **shift** would be "6".
	Possible Values:	Data field index
⇨ x	Long	X coordinate of the mouse cursor

⇨ y	Long	Y coordinate of the mouse cursor

OnMouseMove

Event of VcGantt

This event occurs when the user moves the mouse.

Please also regard the **MouseProcessingEnabled** property.

	Data Type	Explanation
Parameter:		
⇨ button	Integer	indicates the mouse button(s) pressed: **1** represents the left button, **2** is the right button, and the middle button is represented by **4**.
	Possible Values:	Data field index
⇨ Shift	Integer	Number that indicates which one of the **Shift**, **Ctrl**, and **Alt** keys was pressed. **1** corresponds to the Shift key, **2** to the Ctrl key and **4** to the Alt key. Some, all, or none of the numbers may have been set, indicating that some, all, or none of the keys are depressed, respectively. When more than one key is in depressed state, their values add up. For example, if both the Ctrl and Alt keys are depressed, the value of **shift** would be "6".
	Possible Values:	Data field index
⇨ x	Long	X coordinate of the mouse cursor
⇨ y	Long	Y coordinate of the mouse cursor

OnMouseUp

Event of VcGantt

This event occurs when the user loosens the pressed left mouse button.

Please also regard the **MouseProcessingEnabled** property.

	Data Type	Explanation
Parameter:		
⇨ button	Integer	Indicates the mouse button(s) pressed: **1** represents the left button, **2** is the right button, and the middle button is represented by **4**.
	Possible Values:	

		Data field index
⇨ Shift	Integer	Number that indicates which one of the **Shift**, **Ctrl**, and **Alt** keys was pressed. **1** corresponds to the Shift key, **2** to the Ctrl key and **4** to the Alt key. Some, all, or none of the numbers may have been set, indicating that some, all, or none of the keys are depressed, respectively. When more than one key is in depressed state, their values add up. For example, if both the Ctrl and Alt keys are depressed, the value of **shift** would be "6".
	Possible Values:	Data field index
⇨ x	Long	X coordinate of the mouse cursor
⇨ y	Long	Y coordinate of the mouse cursor

OnNodeCreate

Event of VcGantt

This event occurs when the user creates a node. The node object is returned, so that a validation can be made.

The data passed by this event can be read, but must not be modified. For modifying them please use the event **OnNodeCreateCompleteEx**.

By setting the return status the create operation can be inhibited.

	Data Type	Explanation
Parameter:		
⇨ node	VcNode	Node to be created
⇔ returnStatus	Variant	Return status
	Possible Values: vcRetStatFalse 0 vcRetStatOK 1	 The node will not be created. The node will be created.

Example Code

```
Private Sub VcGantt1_OnNodeCreate(ByVal node As VcGanttLib.VcNode, _
                                  returnStatus As Variant)

    'Show own "Edit" dialog for the new node
    '(EditNewNodes attribute must be set to off!)
    On Error GoTo CancelError
    frmEditDialog.Show Modal, Me

    'create a record in the underlying database of the application
    addDataRecord node.AllData

    Exit Sub
```

```
CancelError:
    returnStatus = vcRetStatFalse

End Sub
```

OnNodeCreateCompleteEx

Event of VcGantt

This event occurs when the interactive creation of a node is completed. The node object, the creation type (here **vcNodeCreated**) and the information whether the created node is the only node or the last node of a node collection (always **True**) are returned, so that a validation can be made.

	Data Type	Explanation
Parameter:		
⇨ node	VcNode	Node created
⇨ creationType	CreationTypeEnum	Creation type of nodes/links
	Possible Values: vcDataRecordCreated 6	Data record created by interaction
	vcDataRecordCreatedByResourceScheduling 5	Data record automatically created by resource scheduling
	vcNodeCreated 1	node created via mouse-click
⇨ isLastNodeInSeries	Boolean	The created node is/is not the only node or the last node of a node collection.
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
Private Sub VcGantt1_OnNodeCreateCompleteEx(ByVal node As _
                                VcGanttLib.VcNode, ByVal creationType As _
                                VcGanttLib.CreationTypeEnum, _
                                ByVal isLastNodeInSeries As Boolean)
   'create a record in the underlying database of the application
   addDataRecord node.AllData
End Sub
```

OnNodeDelete

Event of VcGantt

This event occurs when the user deletes a node by the context menu. The node object to be deleted is returned, so that you can still check for - whatever - conditions and prohibit the deletion on a negative result by setting the return status.

	Data Type	Explanation
Parameter:		
⇨ node	VcNode	Node deleted
⇔ returnStatus	Variant	Return status
	Possible Values: vcRetStatFalse 0 vcRetStatOK 1	 The node will not be deleted. The node will be deleted.

Example Code

```
Private Sub VcGantt1_OnNodeDelete(ByVal node As VcGanttLib.VcNode, _
                                  returnStatus As Variant)
  'deny the deletion of the last node in the chart
    If VcGantt1.NodeCollection.Count = 1 Then
        returnStatus = vcRetStatFalse
        MsgBox ("The last node in the chart cannot be deleted.")
    End If
End Sub
```

OnNodeDeleteCompleteEx

Event of VcGantt

This event occurs when deleting a node interactively is completed. The node object and the information whether the deleted node was the last one of a batch are returned for data validation.

	Data Type	Explanation
Parameter:		
⇨ node	VcNode	Node deleted
⇨ isLastNodeInSeries	Boolean	The deleted node is (True) / is not (False) the last node of batch
	Possible Values:	 Group invisible/visible group nodes are/are not visible

Example Code

```
Private Sub VcGantt1_OnNodeDeleteCompleteEx(ByVal node As VcGanttLib.VcNode,
ByVal isLastNodeInSeries As Boolean)

    MsgBox "The node " & node.AllData & " was deleted."

End Sub
```

OnNodeLClick

Event of VcGantt

This event occurs when the user clicks the left mouse button on a node (location = vcInDiagram) or on a table entry related to an activity (location = vcInTable). The node object and the cursor position (x,y-coordinates) are captured and passed.

	Data Type	Explanation
Parameter:		
⇨ node	VcNode	Node hit
⇨ location	LocationEnum	Location in the diagram
	Possible Values: vcInDiagram 1 vcInTable 0	 Located in the node area Located in the table area
⇨ x	Long	X coordinate of the mouse cursor
⇨ y	Long	Y coordinate of the mouse cursor
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcGantt1_OnNodeLClick(ByVal node As VcGanttLib.VcNode, _
                                  ByVal location As VcGanttLib.LocationEnum, _
                                  ByVal x As Long, ByVal y As Long, _
                                  returnStatus As Variant)
    'change data field of the node
    node.DataField(8) = 1 - CInt(node.DataField(8))
End Sub
```

OnNodeLDblClick

Event of VcGantt

This event occurs when the user double-clicks the left mouse button on a node (location = vcInDiagram) or on a table entry related to an activity (location = vcInTable). The node object and the cursor position (x,y-coordinates) are captured and passed. By setting the returnStatus, the integrated **Edit Data** dialog can be inhibited to appear.

	Data Type	Explanation
Parameter:		
⇨ node	VcNode	Node hit
⇨ location	LocationEnum	Location in the diagram
	Possible Values: vcInDiagram 1 vcInTable 0	 Located in the node area Located in the table area

⇨ x	Long	X coordinate of the mouse cursor
⇨ y	Long	Y coordinate of the mouse cursor
⇔ returnStatus	Variant	Return status
	Possible Values: vcRetStatFalse 0 vcRetStatOK 1	 The **Edit data** dialog will not appear. The **Edit data** dialog will appear.

Example Code

```
Private Sub VcGantt1_OnNodeLDblClick(ByVal node As VcGanttLib.VcNode, _
                                ByVal location As VcGanttLib.LocationEnum, _
                                ByVal x As Long, ByVal y As Long, _
                                returnStatus As Variant)

' Show own "Edit Node" dialog
    On Error GoTo CancelError
    frmEditDialog.setNode node
    frmEditDialog.Show Modal, Me

    returnStatus = vcRetStatFalse

    Exit Sub

CancelError:
    returnStatus = vcRetStatFalse

End Sub
```

OnNodeModifyComplete

Event of VcGantt

This event occurs when the modification of the node specified is finished.

	Data Type	Explanation
Parameter:		
⇨ node	VcNode	Node modified
⇨ isLastNodeInSeries	Boolean	The modified node is/is not the only node or the last node of a node collection.
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
Private Sub VcGantt1_OnNodeModifyComplete(ByVal node As _
                                VcGanttLib.VcNode, ByVal _
                                isLastNodeInSeries As Boolean)

    ' Modify a record in the underlying database of the application
    modifyDataRecord node.AllData

End Sub
```

OnNodeModifyCompleteEx

Event of VcGantt

This event occurs after the user has modified the node.

	Data Type	Explanation
Parameter:		
⇨ node	VcNode	Node modified
⇨ isLastNodeInSeries	Boolean	The modified node is/is not the only node or the last node of a node collection.
	Possible Values:	Group invisible/visible group nodes are/are not visible
⇨ modificationType	ModificationTypeEnum	Type of modification
	Possible Values:	
	vcAnything 1	modification type not determined
	vcChangedGroup 16	group of the node changed
	vcEndModified 4	The end date of the node has changed.
	vcHierarchyModified 64	Hierarchy of the nodes was changed
	vcModifiedByResourceScheduling 128	Modification by resource scheduling (occurs with data records only)
	vcModifiedBySchedule 32	Modification by new date calculation
	vcMoved 8	Object was moved
	vcNothing 0	no modification
	vcStartModified 2	The start date of the node changed

Example Code

```
Private Sub VcGantt1_OnNodeModifyCompleteEx(ByVal node As VcGanttLib.VcNode,
ByVal modificationType As VcGanttLib.ModificationTypeEnum, ByVal
isLastNodeInSeries As Boolean)

    Select Case modificationType
        Case ModificationTypeEnum.vcMoved
            node.MarkNode = True
    End Select

End Sub
```

OnNodeModifyEx

Event of VcGantt

This event occurs when the user modifies a node. In the course of this, the length or the position of the node or a value in the **Edit Data** dialog may have been changed. The data of the node before and after the modification are passed. By the **modificationType** parameter you get further information of the kind of modification. By setting the returnStatus to **vcRetStatFalse**, the modification will be inhibited.

This event should be used only for reading data of the current node, but not for modifying them. For modifying data please use **OnNodeModify-Complete**.

	Data Type	Explanation
Parameter:		
⇨ oldNode	VcNode	Node before the modification
⇨ node	VcNode	Node to be modified
⇨ modificationType	ModificationTypeEnum	Type of modification (A combination of the values is also possible.)
	Possible Values: vcAnything 1 vcChangedGroup 16 vcEndModified 4 vcHierarchyModified 64 vcModifiedByResourceScheduling 128 vcModifiedBySchedule 32 vcMoved 8 vcNothing 0 vcStartModified 2	 modification type not determined group of the node changed The end date of the node has changed. Hierarchy of the nodes was changed Modification by resource scheduling (occurs with data records only) Modification by new date calculation Object was moved no modification The start date of the node changed
⇦ returnStatus	Variant	Return status

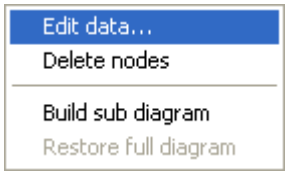
Example Code

```
Private Sub VcGantt1_OnNodeModifyEx(ByVal oldNode As _
                                VcGanttLib.VcNode, ByVal node As _
                                VcGanttLib.VcNode, ByVal modificationType As _
                                VcGanttLib.ModificationTypeEnum, returnStatus _
                                As Variant)

    ' Revoke the modification if the node would change the group
    If modificationType And vcChangedGroup Then
        MsgBox "The node cannot be moved into another group."
        returnStatus = vcRetStatFalse
    End If

End Sub
```

OnNodeRClick

Event of VcGantt

This event occurs when the user clicks the right mouse button on a node (location = vcInDiagram) or on a table entry related to an activity (location = vcInTable). The node object hit and the cursor position (x,y-coordinates) are returned. By setting the return status you can inhibit the integrated context menu to pop up and replace it by a context menu of your own at the coordinates delivered.



Above: integrated context menu

	Data Type	Explanation
Parameter:		
⇨ node	VcNode	Node hit
⇨ location	LocationEnum	Placed in the chart
	Possible Values: vcInDiagram 1 vcInTable 0	 Located in the node area Located in the table area
⇨ x	Long	X coordinate of the mouse cursor
⇨ y	Long	Y coordinate of the mouse cursor
⇔ returnStatus	Variant	Return status
	Possible Values: vcRetStatNoPopup 4 vcRetStatOK 1	 The context menu will be inhibited. The context menu will appear.

Example Code

```
Private Sub VcGantt1_OnNodeRClick(ByVal node As VcGanttLib.VcNode, _
                                  ByVal location As VcGanttLib.LocationEnum, _
                                  ByVal x As Long, ByVal y As Long, _
                                  returnStatus As Variant)

   ' Start own popup menu at the current mouse cursor position
    PopupMenu mnuNodePopup

    returnStatus = vcRetStatNoPopup

End Sub
```

OnNodeResizeStart

Event of VcGantt

This event occurs when the user starts to interactively stretch or shorten a node. It may serve to set smaller modifications to the XGantt, such as making step size depend on nodes (TimeUnitsPerStep).

	Data Type	Explanation
Parameter:		
⇨ layer	VcLayer	Layer that was dragged

OnNodesMarkComplete

Event of VcGantt

This event occurs when the operation of marking or unmarking nodes is finished.

	Data Type	Explanation
Parameter:		
⇦ (no parameter)		No parameter

Example Code

```
Private Sub VcGantt1_OnNodesMarkComplete()
    MsgBox "Nodes have been marked successfully."
End Sub
```

OnNodesMarkEx

Event of VcGantt

This event occurs when the user selects nodes for marking or when he unmarks marked nodes by a click into an empty section of the diagram. The NodeCollection contains the nodes selected by the most recent marking action of the user. If the user unmarked nodes by a click into an empty section, the NodeCollection will be empty.

The parameters **button** and **shift** return the control and mouse buttons that were pressed. If you set the return status to **vcRetStatFalse**, you have to mark or unmark nodes yourself.

The data passed by this event can be read, but must not be modified. For modifying them please use **OnNodesMarkComplete**.

	Data Type	Explanation
Parameter:		
⇨ nodeCollection	VcNodeCollection	NodeCollection that contains the nodes selected by the user. If the user clicked in the diagram, the NodeCollection is empty.
⇨ button	Integer	Indicates in which way the buttons were marked: **0**: by keyboard, **1**: left mouse button pressed, **2**: right mouse button pressed, **4**: mouse button pressed
	Possible Values:	Data field index

⇨ shift	Integer	Number that indicates which one of the **Shift**, **Ctrl**, and **Alt** keys was pressed. **1** corresponds to the Shift key, **2** to the Ctrl key and **4** to the Alt key. Some, all, or none of the numbers may have been set, indicating that some, all, or none of the keys are depressed, respectively. When more than one key is in depressed state, their values add up. For example, if both the Ctrl and Alt keys were depressed, the value of **shift** would equal "6".
	Possible Values:	Data field index
⇔ returnStatus	Variant	Return status
	Possible Values: vcRetStatFalse 0 vcRetStatOK 1	 Marking has to be done manually. Marking is done automatically.

Example Code

```
Private Sub VcGantt1_OnNodesMarkEx(ByVal NodeCollection As _
                                VcGanttLib.VcNodeCollection, ByVal button _
                                As Integer, ByVal shift As Integer, _
                                returnStatus As Variant)

If MsgBox("Mark this node?", vbYesNo, "Marking nodes") = _
    vbNo Then returnStatus = vcRetStatFalse

End Sub
```

OnNumericScaleLClick

Event of VcGantt

This event occurs when the user clicks the left mouse button on the numeric scale. The numeric scale object and the cursor position (x,y-coordinates) are returned.

	Data Type	Explanation
Parameter:		
⇨ numericScale	VcNumericScale	Numeric scale hit
⇨ x	Long	X coordinate of the mouse cursor
⇨ y	Long	Y coordinate of the mouse cursor
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcGantt1_OnNumericScaleLClick(ByVal numericScale As _
                                VcGanttLib.VcNumericScale, ByVal x As Long, _
                                ByVal y As Long, returnStatus As Variant)
    numericScale.BackgroundColor = RGB(222, 211, 33)
End Sub
```

OnNumericScaleLDblClick

Event of VcGantt

This event occurs when the user double-clicks the left mouse button on the numeric scale. The numeric scale object and the mouse position (x,y-coordinates) are returned.

	Data Type	Explanation
Parameter:		
⇨ numericScale	VcNumericScale	Numeric scale hit
⇨ x	Long	X coordinate of the mouse cursor
⇨ y	Long	Y coordinate of the mouse cursor
⇔ returnStatus	Variant	Return status

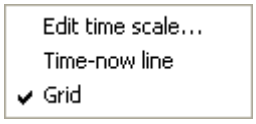
Example Code

```
Private Sub VcGantt1_OnNumericScaleLDblClick(ByVal numericScale As _
                                VcGanttLib.VcNumericScale, ByVal x As Long, _
                                ByVal y As Long, returnStatus As Variant)

    numericScale.MajorTicks = Text1.Text

End Sub
```

OnNumericScaleRClick

Event of VcGantt

This event occurs when the user clicks the right mouse button on a numeric scale. The numeric scale object and the mouse position (x,y-coordinates) are returned. By setting the return status you can inhibit the integrated context menu to pop up and replace it by a context menu of your own at the coordinates delivered.



	Data Type	Explanation
Parameter:		
⇨ numericScale	VcNumericScale	Numeric scale hit
⇨ x	Long	X coordinate of the mouse cursor
⇨ y	Long	Y coordinate of the mouse cursor
⇔ returnStatus	Variant	Return status
	Possible Values:	

vcRetStatNoPopup 4	The context menu will be inhibited.
vcRetStatOK 1	The context menu will appear.

Example Code

```
Private Sub VcGantt1_OnNumericScaleRClick(ByVal numericScale As _
                                VcGanttLib.VcNumericScale, ByVal x As Long, _
                                ByVal y As Long, returnStatus As Variant)

    If MsgBox("Change unit label of numeric scale?", vbYesNo) = vbYes Then
        numericScale.UnitLabel = Text1.Text
    End If

End Sub
```

OnNumericScaleRescale

Event of VcGantt

This event occurs when the user rescales the numeric scale. The NumericScale object and the new BasicUnitWidth are returned, so that you can check whether the scaling is allowed. By setting the return status you can inhibit the modification.

	Data Type	Explanation
Parameter:		
⇨ numericScale	VcNumericScale	Numeric scale hit
⇨ newBasicUnitWidth	Long	New width of the basic unit
⇔ returnStatus	Variant	Return status
	Possible Values:	
	vcRetStatFalse 0	The numeric scale will not be modified.
	vcRetStatOK 1	The numeric scale will be modified.

Example Code

```
Private Sub VcGantt1_OnNumericScaleRescale (ByVal numericScale As _
                                VcGanttLib.VcNumericScale, _
                                ByVal newBasicUnitWidth As Long, _
                                returnStatus As Variant)

    Select Case newBasicUnitWidth
    Case Is <= 1000
        MsgBox "New basic unit width: " & newBasicUnitWidth
    Case Is > 1000
        MsgBox "The maximum basic unit width is 1000."
        returnStatus = vcRetStatFalse
    End Select
End Sub
```

OnObjectDrawCompleteEx

Event of VcGantt

This events only occurs after an object was drawn. It lets you complete or modify the shape of objects drawn by VARCHART XGantt by programming code of your own.

ObjectDraw events are only triggered after the corresponding option was set to its special object type. The option is available for layers and user-defined annotation ribbons.

To draw a layer, you either have to set the property **ObjectDrawEvents-Enabled** of the object **VcLayer** to **True** at run time, or alternatively, at design time, you tick the check box **ObjectDraw Events** for the according layer in the **Specify Bar Appearance** dialog.

To draw a user-defined annotation ribbon, you have to tick the check box **ObjectDraw Events** for the according ribbon in the **Edit Histograms** dialog

If you wish to suppress default drawing of layers or annotation ribbons and to replace it by programming code of your own, please use the event **On-ObjectDrawEx**.

	Data Type	Explanation
Parameter:		
hDC	Long	Device context
object	Object	Object which is drawn
objectType	VcObjectTypeEnum	Type of object to be drawn
	Possible Values: vcObjTypeNodeInDiagram 2 vcObjTypeNodeInLegend 17 vcObjTypeNumericScale 10 vcObjTypeSummaryNode 14	 object type **node in diagram area** object type **node in legend area** object type **numeric scale** object type **summary bar**
subObject	Object	Sub-object that is passed context-dependent
subObjectType	VcObjectTypeEnum	Type of subobject
	Possible Values: vcObjTypeLayer 8	 object type **layer**
completeRect	VcRect	Rectangle in device coordinates into which the complete object is to be drawn
updateRect	VcRect	Rectangle in device coordinates which marks the update area. This area may be the same size as or smaller than the rectangle in completeRect.

lineWidth	Long	Width of a thin line. May be used in case of drawing commands in order to adapt the line width to the device context (monitor or printer).
xZoomFactor	Double	This parameter specifies the zoom factor in x-direction, which allows a conversion from distances specified as units of 1/100 mm into pixels, and vice versa. The zoom factor refers to the output device (screen, print preview or printer).
yZoomFactor	Double	This parameter specifies the zoom factor in y-direction, which allows a conversion from distances specified as units of 1/100 mm into pixels, and vice versa. The zoom factor refers to the output device (screen, print preview or printer).

Example Code

```
Private Declare Function GetStockObject Lib "gdi32" (ByVal nIndex As Long) As
Long
Private Const WHITE_BRUSH = 0
Private Const WHITE_PEN = 6
Private Declare Function FillRect Lib "user32" (ByVal hdc As Long, lpRect As
RECT, ByVal hBrush As Long) As Long
Private Type RECT
        Left As Long
        Top As Long
        Right As Long
        Bottom As Long
End Type

Private Sub VcGantt1_OnObjectDrawComplete(ByVal hdc As Long, ByVal Object As
Object, ByVal objectType As VcGanttLib.VcObjectTypeEnum, ByVal subObject As
Object, ByVal subObjectType As VcGanttLib.VcObjectTypeEnum, ByVal completeRect
As VcGanttLib.VcRect, ByVal updateRect As VcGanttLib.VcRect, ByVal xZoomFactor
As Double, ByVal yZoomFactor As Double)
   Dim smallRect As RECT
   Dim hBrush As Long

   ' drawing a white square into the layer
   hBrush = GetStockObject(WHITE_BRUSH)
   smallRect.Left = completeRect.Left + 2
   smallRect.Top = completeRect.Top + 2
   smallRect.Right = completeRect.Right - 2
   smallRect.Bottom = completeRect.Bottom - 2
   FillRect hdc, smallRect, hBrush
End Sub
```

OnObjectDrawEx

Event of VcGantt

This event is triggered before an object is drawn. It enables you to shape the object by adding your own programming code. By setting the return status to the drawing can be inhibited.

ObjectDraw events are only triggered after the corresponding option was set to its special object type. The option is available for layers and user-defined annotation ribbons.

To draw a layer, you either have to set the property **ObjectDrawEvents-Enabled** of the object **VcLayer** to **True** at run time, or alternatively, at design time, you tick the check box **ObjectDraw Events**for the according layer in the **Specify Bar Appearance** dialog.

To draw a user-defined annotation ribbon, you have to tick the check box **ObjectDraw Events** for the according ribbon in the **Edit Histograms** dialog

To add something to the layer or annotation ribbon drawn by VARCHART XGantt, please use the event **OnObjectDrawCompleteEx**.

	Data Type	**Explanation**
Parameter:		
hDC	Long	Device context
object	Object	Object which is drawn
objectType	VcObjectTypeEnum	Type of object to be drawn
	Possible Values: vcObjTypeNodeInDiagram 2 vcObjTypeNodeInLegend 17 vcObjTypeNumericScale 10 vcObjTypeSummaryNode 14	 object type **node in diagram area** object type **node in legend area** object type **numeric scale** object type **summary bar**
subObject	Object	Sub-object that is passed context-dependent
subObjectType	VcObjectTypeEnum	Type of subobject
	Possible Values: vcObjTypeLayer 8	 object type **layer**
completeRect	VcRect	Rectangle in device coordinates into which the complete object is to be drawn
updateRect	VcRect	Rectangle (in device coordinates) which marks the update area. This area may be the same size as or smaller than the rectangle in completeRect.
lineWidth	Long	Width of a thin line. May be used in case of drawing commands in order to adapt the line width to the device context (monitor or printer).
returnStatus	Variant	Return status
	Possible Values: vcRetStatFalse 0 vcRetStatOK 1	 The object will not be drawn. The object will be drawn.

xZoomFactor	Double	This parameter specifies the zoom factor in x-direction, which allows a conversion from distances specified as units of 1/100 mm into pixels, and vice versa. The zoom factor refers to the output device (screen, print preview or printer).
yZoomFactor	Double	This parameter specifies the zoom factor in y-direction, which allows a conversion from distances specified as units of 1/100 mm into pixels, and vice versa. The zoom factor refers to the output device (screen, print preview or printer).

OnOptimizeTableColumnWidth

Event of VcGantt

This event occurs after a double-click on the separation line between two table columns, provided that on the **General** property page the **Allow table column width optimization** check box was activated or the property **AllowTableColumnWidthOptimization** was set. Then the width of the column on the left will be adapted automatically to the length of the text which it contains. The table and the index of the modified column are returned. By setting the return status, you can inhibit the optimization.

	Data Type	Explanation
Parameter:		
⇨ table	VcTable	Table
⇨ index	Integer	Index of the column modified
	Possible Values:	
		Data field index
⇔ returnStatus	Variant	Return status
	Possible Values:	
	vcRetStatFalse 0	The width of the table column will not be optimized.
	vcRetStatOK 1	The width of the table column will be optimized.

Example Code

```
Private Sub VcGantt1_OnOptimizeTableColumnWidth(ByVal Table As
VcGanttLib.VcTable, ByVal Index As Integer, returnStatus As Variant)

    MsgBox "The index of the modified column is " & Index

End Sub
```

OnPreScrollComponent

Event of VcGantt

This event occurs when you have ordered a scroll action, but before the integrated scrolling process is performed. This events lets you acquire for each interactive scroll action:

1. the scrolled component (only vcDiagramComponent, vcHistogram-Component, vcListComponent and vcRightListComponent are considered as "Master scrollers“ because the other components depend on these and are scrolled together with them)

2. the scrolling direction (horizontal or vertical)

3. the type of user action.

If you set the returnStatus to **vcRetStatFalse**, the integrated scrolling process will be suppressed, and in your application, you can react to the event by using your own solution.

Note: The actual scroll action results from the combination of the parameters **orientation** and **scrollAction**, because in Windows programs the up/left- and down/right actions have got the same numbers, e. g.:

vcScrollActionSBPageLeft = vcScrollActionSBPageUp = 2

vcScrollActionThumbTrackLeft = vcScrollActionThumbTrackUp = 107

The below example shows the difference when using the parameter **orientation** with **VcScrollActionSBPageLeft** and with **vcScrollActionSBPageUp**, which both have the value 2.

	Data Type	Explanation
Parameter:		
⇨ component	ComponentTypeEnum	Component type
	Possible Values:	
	vcAdditionalListComponent 1	additional table
	vcBottomListTitleComponent 14	bottom title bar
	vcBottomRightListTitleComponent 17	bottom right table
	vcBottomTimeScaleComponent 15	bottom time scale
	vcDiagramComponent 4	diagram
	vcHistogramComponent 8	histogram
	vcHistogramVerScaleComponent 7	numeric scale (vertical histogram scale)
	vcLegendComponent 10	legend (currently functionless; return values 00)
	vcListComponent 0	table

	vcListTitleComponent 2	table title
	vcRightListComponent 5	table
	vcRightListTitleComponent 16	table title of the right table
	vcTimeScaleComponent 3	upper time scale
	vcTopTitleComponent 11	upper title bar
⇨ Orientation	ScrollOrientationEnum	Scrolling direction
	Possible Values:	
	vcHorizontal 1	horizontal scrolling
	vcVertical 2	vertical scrolling
⇨ scrollAction	ScrollActionEnum	Type of scrolling
	Possible Values:	
	vcScrollActionAutoscrollDown 102	The view was automatically scrolled downward.
	vcScrollActionAutoscrollLeft 101	The view was automatically scrolled towards the right.
	vcScrollActionAutoscrollRight 102	The view was automatically scrolled towards the left.
	vcScrollActionAutoscrollUp 101	The view was automatically scrolled upward.
	vcScrollActionMouseWheelDown 106	While the mouse wheel was pressed, the mouse was moved downward.
	vcScrollActionMouseWheelLeft 105	While the mouse wheel was pressed, the mouse was moved towards the left.
	vcScrollActionMouseWheelRight 106	While the mouse wheel was pressed, the mouse was moved towards the right.
	vcScrollActionMouseWheelUp 105	While the mouse wheel was pressed, the mouse was moved upward.
	vcScrollActionSBLineDown 1	The view was automatically scrolled to its bottom limit
	vcScrollActionSBLineLeft 0	The view was automatically scrolled to its left limit
	vcScrollActionSBLineRight 1	The view was automatically scrolled to its right limit
	vcScrollActionSBLineUp 0	The view was automatically scrolled to its top limit
	vcScrollActionSBNothing -1	The view was not scrolled
	vcScrollActionSBPageDown 3	The view was scrolled downward by a page
	vcScrollActionSBPageLeft 2	The view was scrolled towards the left by a page
	vcScrollActionSBPageRight 3	The view was scrolled towards the right by a page
	vcScrollActionSBPageUp 2	The view was scrolled upward by a page
	vcScrollActionSBThumbPosition 4	The scrolling by a step has been finished.
	vcScrollActionSBThumbTrack 5	The view was scrolled by a step
	vcScrollActionScrollEnd 104	Scrolling via the **End** button or the context menu to the diagram end (right down)
	vcScrollActionScrollHome 103	Scrolling via the **Pos 1** button or the context menu to the upper left corner of the diagram
	vcScrollActionThumbTrackDown 108	Thumb (bar of the scrollbar) moved down
	vcScrollActionThumbTrackLeft 107	Thumb (bar of the scrollbar) moved toward the left
	vcScrollActionThumbTrackRight 108	Thumb (bar of the scrollbar) moved toward the right
	vcScrollActionThumbTrackUp 107	Thumb (bar of the scrollbar) moved up
⇨ delta	Long	Scrolling length (in pixels)
⇔ returnStatus	Variant	Return status

Example Code

```
If orientation = vcHorizontal and scrollAction = vcScrollActionSBPageLeft _
                    Then MsgBox "Scolled left"
ElseIf orientation =  vcHorizontal _
                    and scrollAction = vcScrollActionSBPageRight _
                    Then MsgBox "Scrolled right"
ElseIf orientation =  vcVertical and scrollAction = vcScrollActionSBPageUp _
                    Then MsgBox "Scrolled up"
End If
```

OnPreScrollDiagramHor

Event of VcGantt

This event occurs when you have ordered a scroll action, but before the integrated scrolling process is performed. The old start and end date of the visible diagram area are returned. The **scrollAction** parameter provides information about the type of the performed scrolling process. If you set the returnStatus to **vcRetStatFalse**, the integrated scrolling process will be suppressed, and in your application, you can react to the event by using your own solution.

	Data Type	Explanation
Parameter:		
⇨ curStartDate	Date/Time	Current start date of the visible part of the diagram
⇨ curEndDate	Date/Time	Current end date of the visible part of the diagram
⇨ scrollAction	ScrollActionEnum	Scrolling type
	Possible Values:	
	vcScrollActionAutoscrollDown 102	The view was automatically scrolled downward.
	vcScrollActionAutoscrollLeft 101	The view was automatically scrolled towards the right.
	vcScrollActionAutoscrollRight 102	The view was automatically scrolled towards the left.
	vcScrollActionAutoscrollUp 101	The view was automatically scrolled upward.
	vcScrollActionMouseWheelDown 106	While the mouse wheel was pressed, the mouse was moved downward.
	vcScrollActionMouseWheelLeft 105	While the mouse wheel was pressed, the mouse was moved towards the left.
	vcScrollActionMouseWheelRight 106	While the mouse wheel was pressed, the mouse was moved towards the right.
	vcScrollActionMouseWheelUp 105	While the mouse wheel was pressed, the mouse was moved upward.
	vcScrollActionSBLineDown 1	The view was automatically scrolled to its bottom limit
	vcScrollActionSBLineLeft 0	The view was automatically scrolled to its left limit
	vcScrollActionSBLineRight 1	The view was automatically scrolled to its right limit
	vcScrollActionSBLineUp 0	The view was automatically scrolled to its top limit
	vcScrollActionSBNothing -1	The view was not scrolled

	vcScrollActionSBPageDown 3	The view was scrolled downward by a page
	vcScrollActionSBPageLeft 2	The view was scrolled towards the left by a page
	vcScrollActionSBPageRight 3	The view was scrolled towards the right by a page
	vcScrollActionSBPageUp 2	The view was scrolled upward by a page
	vcScrollActionSBThumbPosition 4	The scrolling by a step has been finished.
	vcScrollActionSBThumbTrack 5	The view was scrolled by a step
	vcScrollActionScrollEnd 104	Scrolling via the **End** button or the context menu to the diagram end (right down)
	vcScrollActionScrollHome 103	Scrolling via the **Pos 1** button or the context menu to the upper left corner of the diagram
	vcScrollActionThumbTrackDown 108	Thumb (bar of the scrollbar) moved down
	vcScrollActionThumbTrackLeft 107	Thumb (bar of the scrollbar) moved toward the left
	vcScrollActionThumbTrackRight 108	Thumb (bar of the scrollbar) moved toward the right
	vcScrollActionThumbTrackUp 107	Thumb (bar of the scrollbar) moved up
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcGantt1_OnPreScrollDiagramHor(ByVal curStartDate _
                                As Date, ByVal curEndDate As Date, _
                                ByVal scrollAction As _
                                VcGanttLib.ScrollActionEnum, _
                                returnStatus As Variant)

    If curStartDate > "01.01.2014" Then
        returnStatus = vcRetStatFalse
    End If

End Sub
```

OnResourceSchedulingProgress

Event of VcGantt

During the resource scheduling process, this event informs on the progress of the scheduling procedure. The number of jobs scheduled and the total number of jobs are reported. By setting the return status to **vcRetStatFalse**, the scheduling procedure will be abandoned.

	Data Type	Explanation
Parameter:		
⇨ ScheduledJobCount	Long	Number of scheduled jobs
⇨ TotalJobCount	Long	Total number of jobs
⇦ ReturnStatus	Object	Return status
		vcRetStatFalse: scheduling is abandoned
		vcRetStatDefault: scheduling is continued

Example Code

```
Private Sub VcGantt1_OnResourceSchedulingProgress(ByVal scheduledJobCount As
Long, ByVal totalJobCount As Long, returnStatus As Variant)
    MsgBox scheduledJobCount & " of " & totalJobCount
End Sub
```

OnResourceSchedulingWarning

Event of VcGantt

This event is triggered if the resource scheduling procedure finds inconsistencies in the data records (see method **process** in the object VcResource-Scheduler2). This event detects certain errors in the data definition. You can cancel the scheduling procedure by setting the return status.

	Data Type	Explanation
Parameter:		
⇦ warningType	ResSchedWarningTypeEnum	Warning type
	Possible Values:	
	vcResSchedAssignment-LoadPerItemIsZero 23	In the assignment data set specified the content of the data field **LoadOrConsumptionPerItem** is evaluated to be 0. This leads to the assignment being ignored during scheduling.
	vcResSchedAssignment-NoDataRecords 0	No assignment data records exist; the parameter **dataRecord** is **Nothing**.
	vcResSchedAssignment-NoOperationID 3	In the assignment data record also passed the data field of the ID of the operations data record is empty. Because of this, the assignment will be ignored in the ongoing procedure.
	vcResSchedAssignment-NoResourceID 1	In the assignment data record also passed all data fields of IDs of resource data records are empty. Because of this, the assignment will be ignored in the ongoing procedure.
	vcResSchedAssignment-OperationNotFound 4	In the assignment data record also passed the operations data record corresponding to the operations data record ID was not found. Because of this, the assignment will be ignored in the ongoing procedure.
	vcResSchedAssignment-ResourceNotFound 2	In the assignment data record also passed the resource data record corresponding to the resource data record ID was not found. Because of this, the assignment will be ignored in the ongoing procedure.
	vcResSchedAssignment-TimingResourceMultiple 5	The assignment data record also passed represents a prohibited second or other assignment of an operation to a resource of the type **vcResSchedTiming**. Because of this, the assignment will be ignored in the ongoing procedure.
	vcResSchedOperation-LoadPerItemIsZero 24	In the operation data set specified the content of the data field **LoadPerItem** is evaluated to be 0. This leads to the operation being ignored during scheduling.
	vcResSchedOperation-NoTaskID 6	In the operations data record also passed the data field of the ID of the task data record is empty. Because of this, the operation will be ignored in the ongoing procedure.

vcResSchedOperation-OverlapQuantityOutOf-Range 19	This warning occurs if the overlap quantity of an operation exceeds the quantity of the associated task. This warning will cause the task to be excepted from scheduling.
vcResSchedOperation-StartLockDateOutOf-Range 15	This warning occurs if the start lock date of an operation is not between the release date and teh due date of the task. This warning will cause the task to be excepted from scheduling.
vcResSchedOperation-TaskNotFound 7	In the operations data record also passed the task data record corresponding to the task data record ID was not found. Because of this, the operation will be ignored in the ongoing procedure.
vcResSchedOperation-WorkInProcessOutOf-Range 20	This warning occurs if the work already completed of the operation data set passed exceeds the quantity of the associated task. This warning will cause the task to be excepted from scheduling.
vcResSchedResource-CalendarNotFound 22	This warning occurs if the calendar object of the name stored in the data field denoted by the property ResourceCalendarNameFieldIndex does not exist.
vcResSchedResource-GroupResourceNot-Found 10	In the resources data record also passed the resource data record corresponding to the group resource data record ID was not found. Because of this, the recource cannot be allocated to a group.
vcResSchedResource-HistogramNotFound 21	This warning occurs if the histogram of a name equal to the resource does not exist.
vcResSchedResource-InputCurveIsCompletely-Zero 12	The values of the input curve of the resource data record also passed are all zero. Input curves for resources of the type **vcResSchedTiming** and **vcResSchedWork** are capacity curves; for resources of the resource type **vcResSchedMaterial** they are supply curves.
vcResSchedResource-InputCurveNotFound 11	The input curve of the resource data record also passed was not found. Input curves for resources of the type **vcResSchedTiming** and **vcRes-SchedWork** are capacity curves; for resources of the resource type **vcResSchedMaterial** they are supply curves.
vcResSchedResource-OutputCurveNotFound 13	The output curve of the resource data record also passed was not found. Output curves for resources of the type **vcResSchedTiming** and **vcResSchedWork** are workload curves; for resources of the resource type **vcResSched-Material** they are stock curves.
vcResSchedResource-OutputCurveOfFalse-Type 14	The output curve of the resource data record also passed cannot be used, since it is not of the type **vcSetCurve** (please see method **CurveType** of the object **VcCurve**). Output curves for resources of the type **vcResSchedTiming** and **vcRes-SchedWork** are workload curves; for resources of the resource type **vcResSchedMaterial** they are stock curves.
vcResSchedTaskCapacity-BeyondLimit 25	This warning occurs if there is at least one operation in the task whose capacity demand is above an internal limit. The capacity demand results from the task quantity in the task, the LoadPerItem in the operation and, if necessary, an efficiency factor in the resource. The current limit is 100000.
vcResSchedTaskDueDate-EarlierThanReleaseDate 9	In the task data record also passed the release date is earlier than the due date. Because of this, the task will be ignored in the ongoing procedure.

	vcResSchedTaskDueDate-EqualToReleaseDate 18	This warning occurs if the release date of a task equals the due date. This warning will cause the task to be excepted from scheduling.
	vcResSchedTaskDueDate-OutOfRange 17	This warning occurs if the due date of a task is not between the PlanningStartDate and the PlanningEndDate or between the dates in the visible section (default). If also the release date is outside the time span allowed, the task will be excepted from scheduling.
	vcResSchedTaskQuantity-IsZero 8	In the task data record also passed the task quantity is zero. Because of this, the task will be ignored in the ongoing procedure.
	vcResSchedTaskRelease-DateOutOfRange 16	This warning occurs if the release date of a task is not between the PlanningStartDate and the PlanningEndDate or between the dates set by the default. If also the due date is outside the time span allowed, the task will be excepted from scheduling.
⇦ dataRecord	VcDataRecord	Data record, to which the warning refers
⇦ returnStatus	Variant	Return status
		vcRetStatFalse: scheduling is abandoned
		vcRetStatDefault: scheduling is continued
	Possible Values:	
	vcRetStatFalse 0	Resource scheduling will be cancelled.
	vcRetStatOK 1	Resource scheduling will be continued.

Example Code

```
Private Sub VcGantt1_OnResourceSchedulingWarning(ByVal warningType As
VcGanttLib.ResourceSchedulingWarningTypeEnum, ByVal DataRecord As
VcGanttLib.VcDataRecord, returnStatus As Variant)

    Select Case warningType
        Case
ResourceSchedulingWarningTypeEnum.vcResSchedTaskDueDateEarlierThanReleaseDate
            MsgBox "Please change the due date " & DataRecord.DataField(11)
    End Select

End Sub
```

OnScrollComponent

Event of VcGantt

For each interactive scrolling action this event lets you identify the below listed values:

1. the scrolled component (only vcDiagramComponent, vcHistogramComponent, vcListComponent and vcRightListComponent are considered as "Master scrollers“ because the other components depend on these and are scrolled together with them)

2. the scrolling direction (horizontal or vertical)

3. the type of user action.

Note: The actual scroll action results from the combination of the parameters **orientation** and **scrollAction**, because in Windows programs the up/left- and down/right actions have got the same numbers, e. g.:

vcScrollActionSBPageLeft = vcScrollActionSBPageUp = 2

vcScrollActionThumbTrackLeft = vcScrollActionThumbTrackUp = 107

The following example shows the distinction by the usage of the parameter **orientation** for **VcScrollActionSBPageLeft** and **vcScrollActionSBPageUp** which have both the value 2.

	Data Type	**Explanation**
Parameter:		
⇨ component	ComponentTypeEnum	Component type
	Possible Values:	
	vcAdditionalListComponent 1	additional table
	vcBottomListTitleComponent 14	bottom title bar
	vcBottomRightListTitleComponent 17	bottom right table
	vcBottomTimeScaleComponent 15	bottom time scale
	vcDiagramComponent 4	diagram
	vcHistogramComponent 8	histogram
	vcHistogramVerScaleComponent 7	numeric scale (vertical histogram scale)
	vcLegendComponent 10	legend (currently functionless; return values 00)
	vcListComponent 0	table
	vcListTitleComponent 2	table title
	vcRightListComponent 5	table
	vcRightListTitleComponent 16	table title of the right table
	vcTimeScaleComponent 3	upper time scale
	vcTopTitleComponent 11	upper title bar
⇨ Orientation	ScrollOrientationEnum	Scrolling direction
	Possible Values:	
	vcHorizontal 1	horizontal scrolling
	vcVertical 2	vertical scrolling
⇨ scrollAction	ScrollActionEnum	Type of scrolling
	Possible Values:	
	vcScrollActionAutoscrollDown 102	The view was automatically scrolled downward.
	vcScrollActionAutoscrollLeft 101	The view was automatically scrolled towards the right.
	vcScrollActionAutoscrollRight 102	The view was automatically scrolled towards the left.
	vcScrollActionAutoscrollUp 101	The view was automatically scrolled upward.
	vcScrollActionMouseWheelDown 106	While the mouse wheel was pressed, the mouse was moved downward.
	vcScrollActionMouseWheelLeft 105	While the mouse wheel was pressed, the mouse was moved towards the left.
	vcScrollActionMouseWheelRight 106	While the mouse wheel was pressed, the mouse was moved towards the right.

vcScrollActionMouseWheelUp 105	While the mouse wheel was pressed, the mouse was moved upward.
vcScrollActionSBLineDown 1	The view was automatically scrolled to its bottom limit
vcScrollActionSBLineLeft 0	The view was automatically scrolled to its left limit
vcScrollActionSBLineRight 1	The view was automatically scrolled to its right limit
vcScrollActionSBLineUp 0	The view was automatically scrolled to its top limit
vcScrollActionSBNothing -1	The view was not scrolled
vcScrollActionSBPageDown 3	The view was scrolled downward by a page
vcScrollActionSBPageLeft 2	The view was scrolled towards the left by a page
vcScrollActionSBPageRight 3	The view was scrolled towards the right by a page
vcScrollActionSBPageUp 2	The view was scrolled upward by a page
vcScrollActionSBThumbPosition 4	The scrolling by a step has been finished.
vcScrollActionSBThumbTrack 5	The view was scrolled by a step
vcScrollActionScrollEnd 104	Scrolling via the **End** button or the context menu to the diagram end (right down)
vcScrollActionScrollHome 103	Scrolling via the **Pos 1** button or the context menu to the upper left corner of the diagram
vcScrollActionThumbTrackDown 108	Thumb (bar of the scrollbar) moved down
vcScrollActionThumbTrackLeft 107	Thumb (bar of the scrollbar) moved toward the left
vcScrollActionThumbTrackRight 108	Thumb (bar of the scrollbar) moved toward the right
vcScrollActionThumbTrackUp 107	Thumb (bar of the scrollbar) moved up

Example Code

```
If orientation = vcHorizontal and scrollAction = vcScrollActionSBPageLeft _
                    Then MsgBox "Scolled left"
ElseIf orientation =  vcHorizontal and _
                    scrollAction = vcScrollActionSBPageRight _
                    Then MsgBox "Scrolled right"
ElseIf orientation =  vcVertical and scrollAction = vcScrollActionSBPageUp _
                    Then MsgBox "Scrolled up"
End If
```

OnScrollDiagramHor

Event of VcGantt

This event occurs after a scroll action was performed. The new start and end date of the visible diagram area are captured and passed. The **scrollAction** parameter provides information about the type of the performed scrolling process.

	Data Type	Explanation
Parameter:		
⇨ newStartDate	Date/Time	New start date of the visible part of the diagram

⇨ newEndDate	Date/Time	New final date of the visible part of the diagram
⇨ scrollAction	ScrollActionEnum	Scrolling type
	Possible Values:	
	vcScrollActionAutoscrollDown 102	The view was automatically scrolled downward.
	vcScrollActionAutoscrollLeft 101	The view was automatically scrolled towards the right.
	vcScrollActionAutoscrollRight 102	The view was automatically scrolled towards the left.
	vcScrollActionAutoscrollUp 101	The view was automatically scrolled upward.
	vcScrollActionMouseWheelDown 106	While the mouse wheel was pressed, the mouse was moved downward.
	vcScrollActionMouseWheelLeft 105	While the mouse wheel was pressed, the mouse was moved towards the left.
	vcScrollActionMouseWheelRight 106	While the mouse wheel was pressed, the mouse was moved towards the right.
	vcScrollActionMouseWheelUp 105	While the mouse wheel was pressed, the mouse was moved upward.
	vcScrollActionSBLineDown 1	The view was automatically scrolled to its bottom limit
	vcScrollActionSBLineLeft 0	The view was automatically scrolled to its left limit
	vcScrollActionSBLineRight 1	The view was automatically scrolled to its right limit
	vcScrollActionSBLineUp 0	The view was automatically scrolled to its top limit
	vcScrollActionSBNothing -1	The view was not scrolled
	vcScrollActionSBPageDown 3	The view was scrolled downward by a page
	vcScrollActionSBPageLeft 2	The view was scrolled towards the left by a page
	vcScrollActionSBPageRight 3	The view was scrolled towards the right by a page
	vcScrollActionSBPageUp 2	The view was scrolled upward by a page
	vcScrollActionSBThumbPosition 4	The scrolling by a step has been finished.
	vcScrollActionSBThumbTrack 5	The view was scrolled by a step
	vcScrollActionScrollEnd 104	Scrolling via the **End** button or the context menu to the diagram end (right down)
	vcScrollActionScrollHome 103	Scrolling via the **Pos 1** button or the context menu to the upper left corner of the diagram
	vcScrollActionThumbTrackDown 108	Thumb (bar of the scrollbar) moved down
	vcScrollActionThumbTrackLeft 107	Thumb (bar of the scrollbar) moved toward the left
	vcScrollActionThumbTrackRight 108	Thumb (bar of the scrollbar) moved toward the right
	vcScrollActionThumbTrackUp 107	Thumb (bar of the scrollbar) moved up

Example Code

```
Private Sub VcGantt1_OnScrollDiagramHor(ByVal newStartDate _
                                As Date, ByVal newEndDate As Date, _
                                ByVal scrollAction As _
                                VcGanttLib.ScrollActionEnum)

If newEndDate > "01.01.2014" Then
    Select Case MsgBox("scrolling to: " & newEndDate, vbOKCancel)
        Case vbOK
```

```
            Call VcGantt1.ScrollToDate(newEndDate, vcHorCenterAligned, 0)
        Case vbCancel
            returnStatus = vcRetStatFalse
    End Select
End If

End Sub
```

OnSelectField

Event of VcGantt

This event occurs, if a cell in a table or a field in a box was selected. The selection can be inhibited by setting the return status.

	Data Type	Explanation
Parameter:		
editObject	Object editiertes Objekt	
editObjectType	VcObjectTypeEnum Objekttyp	
	Possible Values:	
	vcObjTypeBox 15	object type **box**
	vcObjTypeCalendarGrid 18	object type **calendar grid**
	vcObjTypeCurve 12	object type **curve**
	vcObjTypeDateLine 9	object type **date line**
	vcObjTypeGroup 7	object type **group**
	vcObjTypeGroupInDiagram 11	object type **group in diagram area**
	vcObjTypeGroupInTable 7	object type **group in table area**
	vcObjTypeHistogram 13	object type **histogram**
	vcObjTypeLayer 8	object type **layer**
	vcObjTypeLinkCollection 3	object type **link collection**
	vcObjTypeNodeInDiagram 2	object type **node in diagram area**
	vcObjTypeNodeInLegend 17	object type **node in legend area**
	vcObjTypeNodeInTable 1	object type **node in table area**
	vcObjTypeNone 0	no object
	vcObjTypeNumericScale 10	object type **numeric scale**
	vcObjTypeSummaryNode 14	object type **summary bar**
	vcObjTypeTable 4	object type **table**
	vcObjTypeTableCaption 5	object type **table caption**
	vcObjTypeTimeScale 6	object type **time scale**
fieldIndex	Long Feldindex	
objRectComplete	VcRect komplettes Rechteck des getroffenen Objekts	
objRectVisible	VcRect sichtbares Rechteck des getroffenen Objekts	
fldRectComplete	VcRect komplettes Rechteck des getroffenen Feldes	
fldRectVisible	VcRect sichtbares Rechteck des getroffenen Feldes	
returnStatus	Variant	
	Possible Values:	
	vcRetStatFalse 0	The field will not be selected.
	vcRetStatOK 1	The field will be selected.

OnShowCurveNameInMenu

Event of VcGantt

This event occurs when the names of histogram curves defined by API commands are displayed in a context menu. If you set the returnStatus to **vcRetStatFalse**, the names of the histogram curves are not displayed in a context menu.

	Data Type	Explanation
Parameter:		
⇨ Histogram	VcHistogram	Histogram hit
⇨ curveName	String	Name of the histogram curve
	Possible Values:	Name of the color map
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcGantt1_OnShowCurveNameInMenu(ByVal Histogram As _
                                VcGanttLib.VcHistogram,_
                                ByVal curveName As String, _
                                returnStatus As Variant)

    returnStatus = retStatFalse

End Sub
```

OnShowDate

Event of VcGantt

This event occurs when the user moves the mouse inside the diagram or the time scale area. The date of the mouse position is returned.

	Data Type	Explanation
Parameter:		
⇨ dateVal	Date/Time	Date

Example Code

```
Private Sub VcGantt1_OnShowDate(ByVal dateVal As Date)

    Text1.Text = dateVal

End Sub
```

OnShowInPlaceEditor

Event of VcGantt

This event occurs when the implemented editor is started.

The event will be activated only if the property **InPlaceEditingAllowed** is set to True.

By setting the return status to **False** this event can be inhibited so that your own editor can be started at the coordinates passed.

	Data Type	Explanation
Parameter:		
⇨ editObject	Object	Object edited
⇨ editObjectType	VcObjectTypeEnum	Object type
	Possible Values: vcObjTypeBox 15 vcObjTypeNodeInLegend 17 vcObjTypeNodeInTable 1 vcObjTypeNone 0	 object type **box** object type **node in legend area** object type **node in table area** no object
⇨ fieldIndex	Long	Field index
⇨ objRectComplete	VcRect	Complete rectangle of the object hit
⇨ objRectVisible	VcRect	Visible rectangle of the object hit
⇨ fldRectComplete	VcRect	Complete rectangle of the field hit
⇨ fldRectVisible	VcRect	Visible rectangle of the field hit
returnStatus	Variant	
	Possible Values: vcRetStatFalse 0 vcRetStatOK 1	 The integrated editor will not start. The integrated editor will start.

Example Code

```
Private Sub VcGantt1_OnShowInPlaceEditor(ByVal editObject As Object, _
                  ByVal editObjectType As VcGanttLib.VcObjectTypeEnum, _
                  ByVal fieldIndex As Long, ByVal objRectComplete As _
                  VcGanttLib.VcRect, ByVal objRectVisible As _
                  VcGanttLib.VcRect, ByVal fldRectComplete As _
                  VcGanttLib.VcRect, ByVal fldRectVisible As _
                  VcGanttLib.VcRect, returnStatus As Variant)

   Dim oldScaleMode As Long

   If editObjectType = vcObjTypeNodeInTable Then
      returnStatus = vcRetStatFalse

      Set myEditObject = editObject
      myEditObjectType = editObjectType
      myEditObjectFieldIndex = fieldIndex
      oldScaleMode = Me.ScaleMode
      Me.ScaleMode = vbPixels

      Select Case fieldIndex
```

```
        Case 1   'Name
           Text1.Left = fldRectVisible.Left + VcGantt1.Left
           Text1.Top = fldRectVisible.Top + VcGantt1.Top
           Text1.Width = fldRectVisible.Width
           Text1.Height = fldRectVisible.Height
           Text1.Text = editObject.DataField(fieldIndex)
           Text1.Visible = True
           Text1.SetFocus

        Case 2, 3     'Start or End
           MonthView1.Left = fldRectVisible.Left + VcGantt1.Left
           MonthView1.Top = fldRectVisible.Top + VcGantt1.Top
           MonthView1.Value = editObject.DataField(fieldIndex)
           MonthView1.Visible = True
           MonthView1.SetFocus

        Case 13   'Employee
           Combo1.Left = fldRectVisible.Left + VcGantt1.Left
           Combo1.Top = fldRectVisible.Top + VcGantt1.Top
           Combo1.Width = fldRectVisible.Width
           Combo1.Text = editObject.DataField(fieldIndex)
           Combo1.Visible = True
           Combo1.SetFocus

      End Select

      Me.ScaleMode = oldScaleMode

   End If

End Sub
```

OnStatusLineText

Event of VcGantt

This event occurs when a message of general interest is displayed in the status line, e.g. a functional note during loading, or some information on the node to which the cursor is pointing.

	Data Type	Explanation
Parameter:		
⇨ text	String	Information text
	Possible Values:	Name of the color map
⇨ paneNo	Integer	Index of pane
	Possible Values:	Data field index

Example Code

```
Private Sub VcGantt1_OnStatusLineText(ByVal Text As String, _
                                      ByVal paneNo As Integer)
    Text1.Text = Text
End Sub
```

OnSupplyTextEntry

Event of VcGantt

This event only occurs when the VcGantt property **EnableSupplyTextEntryEvent** is set to **True**. It occurs when a text is displayed. You can use this event for editing the texts of context menus, dialog boxes, info boxes, error messages and the names of days and months.

	Data Type	Explanation
Parameter:		
⇨ controlIndex	TextEntryIndexEnum	Text constant the contents of which is to be replaced
	Possible Values:	
	vcTXECtxmenArrowMode 2116	Text in context menu:**Pointer mode**
	vcTXECtxmenBarGroupSepLine 2111	Constant not longer in use but still visible in the API
	vcTXECtxmenCancelGrouping 2108	Constant not longer in use but still visible in the API
	vcTXECtxmenCreateBoxMode 2135	Text in context menu: **Mode: Create box**
	vcTXECtxmenCreateLinkMode 2118	Text in context menu: **Mode: Create link**
	vcTXECtxmenCreateNodeMode 2117	Text in context menu: **Mode: Create node**
	vcTXECtxmenDateLineGrid 2106	Text in context menu: **Grid**
	vcTXECtxmenDeleteCurvePoint 2131	Text in context menu: **Delete curve point**
	vcTXECtxmenDeleteLink 2102	Text in context menu: **Delete link**
	vcTXECtxmenDeleteNode 2101	Text in context menu: **Delete nodes**
	vcTXECtxmenEditGroup 2160	Text in context menu of the **group**: **Edit group data**
	vcTXECtxmenEditLink 2154	Text in context menu: **Edit Link**
	vcTXECtxmenEditNode 2100	Text in context menu: **Edit data**
	vcTXECtxmenFilePrint 2122	Text in context menu: **Print**
	vcTXECtxmenFilePrintPreview 2121	Text in context menu: **Print preview**
	vcTXECtxmenFilePrintSetup 2120	Text in context menu: **Print setup**
	vcTXECtxmenFullDiagram 2156	Text in context menu: **Restore full diagram**
	vcTXECtxmenGraphicExport 2123	Text in context menu: **Export graphics**
	vcTXECtxmenGroupCollapse 2114	Text in context menu: **Collapse group**
	vcTXECtxmenGroupCollapseRowsBelow 2129	Text in context menu: **Collapse Rows Below**
	vcTXECtxmenGroupDelete 2115	Text in context menu: **Delete group**
	vcTXECtxmenGrouped 2107	Constant not longer in use but still visible in the API
	vcTXECtxmenGroupExpand 2113	Text in context menu: **Expand Group**
	vcTXECtxmenGroupExpandRowsBelow 2128	Text in context menu: **Expand Rows Below**
	vcTXECtxmenGroupingOptions 2109	Constant not longer in use but still visible in the API
	vcTXECtxmenGroupNodesBelow 2126	Constant not longer in use but still visible in the API
	vcTXECtxmenGroupNodesInOneRow 2127	Text in **group** context menu: **All Nodes In One Row**
	vcTXECtxmenGroupNodesOptimized 2124	Text in **group** context menu: **Arrange Nodes Optimized**
	vcTXECtxmenGroupNodesOverlaid 2125	Text in **group** context menu: **Arrange Nodes Overlaid**
	vcTXECtxmenGroupOutlineIndent 2134	Text in the context menu: **Outline indent**
	vcTXECtxmenGroupOutlineOutdent 2133	Text in the context menu: **Outline outdent**

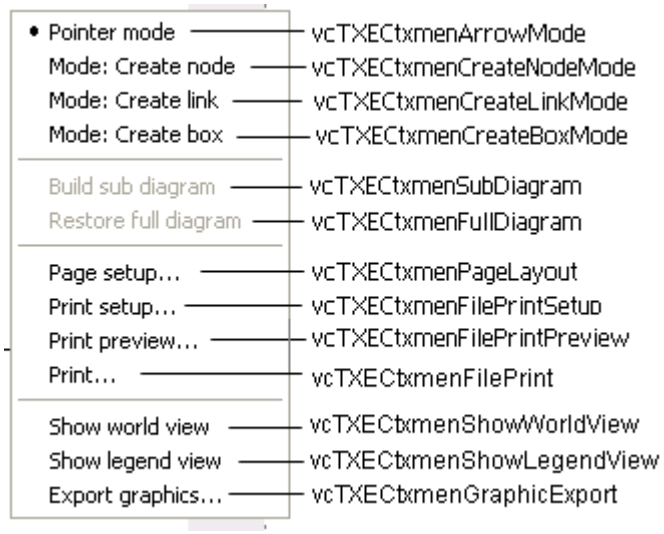
vcTXECtxmenGroupSortingOptions 2110	Text in context menu: **Sorting options for groups**
vcTXECtxmenInsertCurvePointMode 2130	Text in context menu: **Insert curve point**
vcTXECtxmenInvertSelection 2103	Constant not longer in use but still visible in the API
vcTXECtxmenPageLayout 2119	Text in context menu: **Page setup**
vcTXECtxmenReOptimizeNodesInGroup 2136	Text in context menu: **Re-optimize nodes**
vcTXECtxmenShowLegendView 2158	Text in context menu: **Show legend view**
vcTXECtxmenShowWorldView 2157	Text in context menu: **Show world view**
vcTXECtxmenSubDiagram 2155	Text in context menu: **Build sub diagram**
vcTXECtxmenTimeScaleEditor 2104	Text in context menu: **Edit time scale**
vcTXECtxmenToggleDateLine 2105	Text in context menu: **Time-now line**
vcTXECtxmenUnmarkAllCurves 2136	Text in context menu of the **histogram**: **Unmark all curves**
vcTXEDlgLegArrangement 2046	Text in the **Legend Attributes** dialog: **Arrangement**
vcTXEDlgLegBottomMargin 2052	Text in the **Legend Attributes** dialog: **Bottom margin:**
vcTXEDlgLegFixedToColumns 2048	Text in the **Legend Attributes** dialog: **Fixed to columns**
vcTXEDlgLegFixedToRows 2047	Text in the **Legend Attributes** dialog: **Fixed to rows**
vcTXEDlgLegFixedToRowsAndColumns 2049	Text in the **Legend Attributes** dialog: **Fixed to rows and columns**
vcTXEDlgLegIdcancel 2042	**Legend Attributes** dialog: **Cancel** button
vcTXEDlgLegIdd 2040	Dialog **Legend Attributes**: Text in Title Bar
vcTXEDlgLegIdok 2041	Button text in **Legend Attributes** dialog: **OK**
vcTXEDlgLegLegendElements 2045	Text in the **Legend Attributes** dialog: **Legendelements**
vcTXEDlgLegLegendFont 2053	**Legend Attributes** dialog: legend **Font...** button
vcTXEDlgLegLegendTitleFont 2044	**Legend Attributes** dialog: legend title **Font...** button
vcTXEDlgLegLegendTitleVisible 2043	Text in the **Legend Attributes** dialog: **Legend title visible**
vcTXEDlgLegMargins 2050	Text in the **Legend Attributes** dialog: **Margins**
vcTXEDlgLegTopMargin 2051	Text in the **Legend Attributes** dialog: **Top margin:**
vcTXEDlgNedCaptionPrefix 2024	**Edit data** dialog, text for text line: “Node”
vcTXEDlgNedIdapply 2027	**Edit data** dialog, **Apply** button
vcTXEDlgNedIdcancel 2016	Text in the **Edit data** dialog: **Cancel**
vcTXEDlgNedIdclose 2029	**Edit data** dialog: **Close** button
vcTXEDlgNedIdd 2014	caption of the **Edit data** dialog
vcTXEDlgNedIdhelp 2028	**Edit data** dialog: **Help** button
vcTXEDlgNedIdok 2015	Text in the **Edit data** dialog: **OK**
vcTXEDlgNedNamesColStr 2018	Text in the **Edit data** dialog: **Fields**
vcTXEDlgNedTTGotoFirst 2032	**Edit data** dialog: tooltip text **Show first selected activity**
vcTXEDlgNedTTGotoLast 2035	**Edit data** dialog, Tooltip “Show last selected activity
vcTXEDlgNedTTGotoNext 2034	**Edit data** dialog, tooltip text **Show next selected activity**
vcTXEDlgNedTTGotoPrev 2033	**Edit data** dialog: tooltip text **Show previous selected activity**
vcTXEDlgNedValuesColStr 2019	Text in the **Edit data** dialog: **Values**
vcTXEDlgTscEndDate 2012	Text in **Edit time scale** dialog: **End Date**
vcTXEDlgTscIdcancel 2010	**Edit time scale** dialog: button text **Cancel**
vcTXEDlgTscIdd 2008	**Edit time scale** dialog: text in title bar

vcTXEDlgTscIdok 2009	**Edit time scale** dialog: button text **OK**
vcTXEDlgTscScale 2013	Text in **Edit time scale** dialog: **Scale**
vcTXEDlgTscStartDate 2011	Text in dialog **Edit time scale**: **Start Date**
vcTXEErrTxtCannotMoveToEmptyRow 2735	Constant not longer in use but still visible in the API
vcTXEErrTxtEndNotEarlierThanNextSect 2734	Message text: "End date ""%s"" is not earlier than end date of next section.\n\nThe old date will be inserted again."
vcTXEErrTxtEndNotLaterThanStart 2732	Message text: "End date ""%s"" is not later than start date.\n\nThe old date will be inserted again."
vcTXEErrTxtEntryTooLong 2730	Message text: "Entry is too long, %s characters are possible."
vcTXEErrTxtSpinNoButton 2727	Constant not longer in use but still visible in the API
vcTXEErrTxtSpinNumberFormatFloat 2724	Constant not longer in use but still visible in the API
vcTXEErrTxtSpinNumberFormatInt 2723	Constant not longer in use but still visible in the API
vcTXEErrTxtSpinNumberMissing 2722	Constant not longer in use but still visible in the API
vcTXEErrTxtSpinNumberTooHigh 2725	Constant not longer in use but still visible in the API
vcTXEErrTxtSpinNumberTooLow 2726	Constant not longer in use but still visible in the API
vcTXEErrTxtSpinUnitInsert 2720	Constant not longer in use but still visible in the API
vcTXEErrTxtSpinUnitNotInsert 2721	Constant not longer in use but still visible in the API
vcTXEErrTxtSpinWrongFormatString 2728	Constant not longer in use but still visible in the API
vcTXEErrTxtSpinWrongUnitInserted 2718	Constant not longer in use but still visible in the API
vcTXEErrTxtSpinWrongUnitNotInserted 2719	Constant not longer in use but still visible in the API
vcTXEErrTxtStartNotEarlierThanEnd 2731	Message text: "Start date ""%s"" is not earlier than end date.\n\nThe old date will be inserted again."
vcTXEErrTxtStartNotLaterThanPrevSect 2733	Message text: "Start date ""%s"" is not later than start date of previous section.\n\nThe old date will be inserted again."
vcTXEErrTxtWrongLongInteger 2729	Message text: "Entry is not an integer or too big."
vcTXEInfWndChangeEndDate 2615	Tooltip text: **Change End Date**
vcTXEInfWndChangeSectionStartDate 2614	Tooltip text: **Modify section start date**
vcTXEInfWndChangeStartDate 2618	Tooltip text: **Change Start Date.**
vcTXEInfWndCopyActivity 2619	Tooltip text: **Copy Node**
vcTXEInfWndCreateActivity 2611	Tooltip text: **Create Node**
vcTXEInfWndDate 2620	Tooltip text: **Date**
vcTXEInfWndDateValue 12620	Tooltip text: **Date**
vcTXEInfWndDayPl 2604	Tooltip text: **days**
vcTXEInfWndDaySi 2603	Tooltip text: **day**
vcTXEInfWndDuration 2602	Tooltip text: **Duration**
vcTXEInfWndDurationValue 12602	Tooltip text: **Duration**
vcTXEInfWndEnd 2601	Tooltip text: **End**
vcTXEInfWndEndValue 12601	Tooltip text: **End date**
vcTXEInfWndHourPl 2606	Tooltip text: **hours**
vcTXEInfWndHourSi 2605	Tooltip text: **hour**
vcTXEInfWndMinPl 2608	Tooltip text: **minutes**
vcTXEInfWndMinSi 2607	Tooltip text: **minute**
vcTXEInfWndMoveActivity 2612	Tooltip text: **Move Node**
vcTXEInfWndMoveLayer 2613	Tooltip text: **Move Layer**
vcTXEInfWndResizeBUW 2616	Tooltip text: **Resize section width**

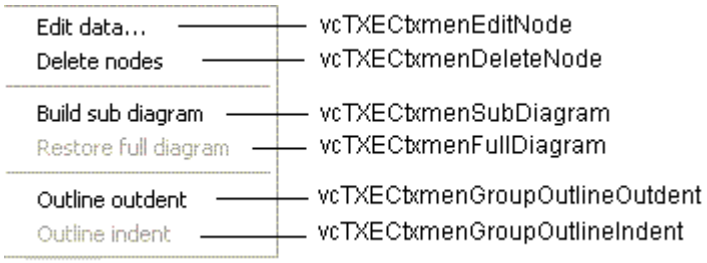
vcTXEInfWndResizeNumericBUW 2617	Tooltip text: **Modify numeric scale's width**
vcTXEInfWndSecPl 2610	Tooltip text: **seconds**
vcTXEInfWndSecSi 2609	Tooltip text: **second**
vcTXEInfWndStart 12600	Tooltip text: **Start date of date line**
vcTXEInfWndStart 2600	Tooltip text: **Start**
vcTXEPrctBtAll 2306	Button text in **Print Preview** dialog: **Overview**
vcTXEPrctBtApply 2318	Button text in **Page Setup** dialog: **Apply**
vcTXEPrctBtCancel 2302	Button text in **Print Busy** box: **Cancel**
vcTXEPrctBtClose 2303	Button text in **Print Preview** dialog: **Close**
vcTXEPrctBtFitToPage 2308	Button text in **Print Preview** dialog: **Fit To Page**
vcTXEPrctBtNext 2305	Button text in **Print Preview** dialog: **Next**
vcTXEPrctBtOk 2301	Button text in **Page Setup** dialog: **OK**
vcTXEPrctBtPageLayout 2311	Button text in **Print Preview** dialog: **Page Setup**
vcTXEPrctBtPreviewZoomFactorItems 2321	Entries in the combobox **Zoom factor** of the **Print Preview** dialog: **!Auto|75%|100%|150%|200%**
vcTXEPrctBtPrevious 2304	Button text in **Print Preview** dialog: **Previous**
vcTXEPrctBtPrint 2313	Button text in **Print Preview** dialog: **Print**
vcTXEPrctBtPrinterSetup 2312	Button text in **Print Preview** dialog: **Printer setup**
vcTXEPrctBtSingle 2307	Button text in **Print Preview** dialog: **Single**
vcTXEPrctBtZoomPrint 2319	Button text in **Print Preview** dialog: **Print Area…**
vcTXEPrctDtAddCuttingMarks 2514	Text in the **Page Setup** dialog: **Show crop marks**
vcTXEPrctDtAdjustTimescale 2560	Page Layout Text: **Adjust time scale to width of pages**
vcTXEPrctDtAdoptTableWidthOfView 2591	Text in **Page Setup** dialog: **Adopt appearance from view on screen**
vcTXEPrctDtAlignment 2526	Text in the **Page Setup** dialog: **Alignment**
vcTXEPrctDtAlignmentItems 2583	Text in the **Page Setup** dialog: **Top left|Top|Top right|Left|Centered|Right|Bottom left|Bottom|Bottom right**
vcTXEPrctDtBottom 2521	Text in the **Page Setup** dialog: **Bottom**
vcTXEPrctDtCm 2530	Text in the **Page Setup** dialog: **cm**
vcTXEPrctDtCombinedFitToPage 2574	Text in the **Page Setup** dialog: **Zoom with horizontal fitting**
vcTXEPrctDtCurrentValues 2581	Text in the **Page Setup** dialog: **Current**
vcTXEPrctDtEnableDiagram 2559	Text in **Page Setup** dialog: **Show diagram**
vcTXEPrctDtEnableTable 2558	Text in **Page Setup** dialog: **Show Table**
vcTXEPrctDtExportPage 2568	
vcTXEPrctDtFitToPage 2508	Text in the **Page Setup** dialog: **Fit to page counts**
vcTXEPrctDtFoldingMarksItems 2577	Text in the **Page Setup** dialog: **Form A|Form B|Form C**
vcTXEPrctDtFoldingMarksText 2576	Text in the **Page Setup** dialog: **Show &folding marks (DIN 824):**
vcTXEPrctDtFooterGroup 2584	Text in the **Page Setup** dialog: **Footer line**
vcTXEPrctDtFrameOutside 2515	Text in the **Page Setup** dialog: **Show frame outside**
vcTXEPrctDtInch 2588	Text in the **Page Setup** dialog: **in**
vcTXEPrctDtLeft 2520	Text in the **Page Setup** dialog: **Left**
vcTXEPrctDtMargins 2529	Text in the **Page Setup** dialog: **Minimum sizes for sheet margins**

vcTXEPrctDtMaxPages 2580	Text in the **Page Setup** dialog: **pages**
vcTXEPrctDtOff 2557	Text **Off** dialog
vcTXEPrctDtOptions 2528	Text in the **Page Setup** dialog: **Options**
vcTXEPrctDtPageDescription 2562	Text in **Page Setup** dialog: **Text**
vcTXEPrctDtPageLayout 2532	**Page Setup** dialog: Text in Title Bar
vcTXEPrctDtPageNumberingItems 2582	Text in the **Page Setup** dialog: **Row.Column|Column.Row|Page/Count**
vcTXEPrctDtPageNumbers 2518	Text in the **Page Setup** dialog: **Page numbering**
vcTXEPrctDtPagePadding 2585	Text in the **Page Setup** dialog: **&Pad pages with space**
vcTXEPrctDtPagePreview 2533	**Print Preview** dialog: Text in Title Bar
vcTXEPrctDtPagesMaxHeight 2511	Text in the **Page Setup** dialog: **Maximum height**
vcTXEPrctDtPagesMaxWidth 2510	Text in the **Page Setup** dialog: **Maximum. width**
vcTXEPrctDtPercent 2509	Text in the **Page Setup** dialog: **%**
vcTXEPrctDtPrintDate 2564	Text in **Page Setup** dialog: **Additionally print current &date**
vcTXEPrctDtPrintingPage 2556	Text in Print Busy Box: **Printing page %1 of %2 on**
vcTXEPrctDtReduceExpand 2507	Text in the **Page Setup** dialog: **Zoom factor**
vcTXEPrctDtRepeatTable 2565	Text in the **Page Setup** dialog: **Repeat title/table/time scale/legend**
vcTXEPrctDtRight 2522	Text in the **Page Setup** dialog: **Right**
vcTXEPrctDtScaling 2527	Text in the **Page Setup** dialog: **Scaling**
vcTXEPrctDtScalingMode 2578	Text in the **Page Setup** dialog: **&Mode:**
vcTXEPrctDtStatusBarCurrentValues 2586	Text in the **Status bar** of the **Page Setup** dialog: **Page %1 selected (in row %2, column %3)**
vcTXEPrctDtStatusBarSelectedPage 2587	Text in the **Status bar** of the **Page Setup** dialog: **Page %1 selected (in row %2, column %3)**
vcTXEPrctDtTableColumnRange 2575	Text in the **Page Layout** dialog: **Show table columns (e.g. 1-5;7)**
vcTXEPrctDtTimeColumnEnd 2590	Text in **Page Setup** dialog: **Time scale end:**
vcTXEPrctDtTimeColumnStart 2589	Text in **Page Setup** dialog: **Time scale start:**
vcTXEPrctDtTop 2519	Text in the **Page Setup** dialog: **Top**
vcTXEPrctDtZoomFactor 2579	Text in the **Page Setup** dialog: **&Zoom factor:**
vcTXEPrctMtAdjustBottomAndTopMargin 2437	Message text: **The bottom margin is out of range and therefore will be reduced to %1 cm.\r\nIn addition, the top margin will be adjusted to %2 cm.**
vcTXEPrctMtAdjustLeftAndRightMargin 2434	Message text: **The left margin is out of range and therefore will be reduced to %1 cm.\r\nIn addition, the right margin will be reduced to %2 cm.**
vcTXEPrctMtAdjustRightAndLeftMargin 2435	Message text: **The right margin is out of range and therefore will be reduced to %1 cm.\r\nIn addition, the left margin will be adjusted to %2 cm.**
vcTXEPrctMtAdjustTopAndBottomMargin 2436	Message text: **The top margin is out of range and therefore will be reduced to %1 cm.\r\nIn addition, the bottom margin will be reduced to %2 cm.**
vcTXEPrctMtBottomMargin 2409	Message text: **Bottom margin is out of range and therefore will be reduced to %s cm.**
vcTXEPrctMtIncompatibleVcVersion 2414	Message text: **VcVersion incompatible**
vcTXEPrctMtLeftMargin 2406	Message text: **Left margin is out of range and therefore will be reduced to %s cm.**

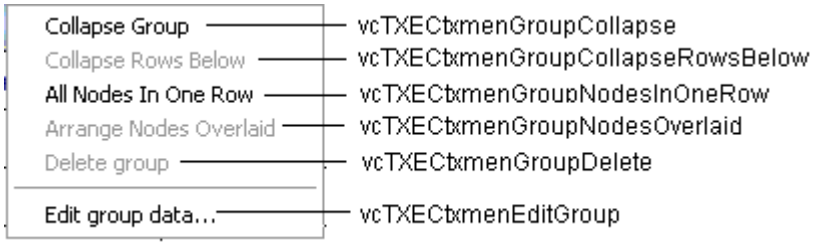
	vcTXEPrctMtPrinterNotInstalled 2411	Message text: **Printer not installed**
	vcTXEPrctMtPrintingNotPossible 2402	Message text: **Printing not possible at time**
	vcTXEPrctMtRightMargin 2408	Message text: **Right margin is out of range and therefore will be reduced to %s cm.**
	vcTXEPrctMtSelectPaperSize 2413	Message text: **Selected paper size too small**
	vcTXEPrctMtTopMargin 2407	Message text: **Top margin is out of range and therefore will be reduced to %s cm.**
	vcTXEPrctMtValueOutOfRange 2404	Message text: **Value out of range %1 to %2**
	vcTXEPrctMtWillBeAdjustedTo 2410	Message text: **Will be adjusted to...**
	vcTXERelTypeLongFF 3001	Text in the **Edit links** dialog: **Finish-to-finish (FF)**
	vcTXERelTypeLongFS 3000	Text in the **Edit links** dialog: **Finish-to-start (FS)**
	vcTXERelTypeLongSF 3003	Text in the **Edit links** dialog: **Start-to-finish (SF)**
	vcTXERelTypeLongSS 3002	Text in the **Edit links** dialog: **Start-to-start (SS)**
	vcTXERibAM 2225	ribbon text for **am**
	vcTXERibCW 2223	ribbon text for **calendar week**
	vcTXERibDay0 2212	ribbon text for **Monday**
	vcTXERibDay1 2213	ribbon text for **Tuesday**
	vcTXERibDay2 2214	ribbon text for **Wednesday**
	vcTXERibDay3 2215	ribbon text for **Thursday**
	vcTXERibDay4 2216	ribbon text for **Friday**
	vcTXERibDay5 2217	ribbon text for **Saturday**
	vcTXERibDay6 2218	ribbon text for **Sunday**
	vcTXERibMon0 2200	ribbon text for **January**
	vcTXERibMon1 2201	ribbon text for **February**
	vcTXERibMon10 2210	ribbon text for **November**
	vcTXERibMon11 2211	ribbon text for **December**
	vcTXERibMon2 2202	ribbon text for **March**
	vcTXERibMon3 2203	ribbon text for **April**
	vcTXERibMon4 2204	ribbon text for **Mai**
	vcTXERibMon5 2205	ribbon text for **June**
	vcTXERibMon6 2206	ribbon text for **July**
	vcTXERibMon7 2207	ribbon text for **August**
	vcTXERibMon8 2208	ribbon text for **September**
	vcTXERibMon9 2209	ribbon text for **October**
	vcTXERiboClock 2224	ribbon text for **o'clock**
	vcTXERibPM 2226	ribbon text for **pm**
	vcTXERibQuar0 2219	ribbon text for **first quarter**
	vcTXERibQuar1 2220	ribbon text for **second quarter**
	vcTXERibQuar2 2221	ribbon text for **third quarter**
	vcTXERibQuar3 2222	ribbon text for **fourth quarter**
⇦ textEntry	String	Text entry to replace the default text
	Possible Values:	Name of the color map
⇨ returnStatus	Variant	Return status



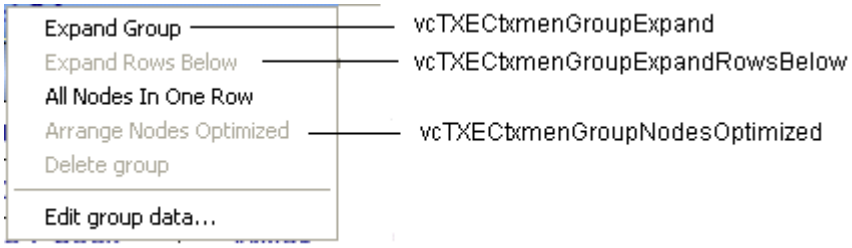
Constants of the diagram's context menu



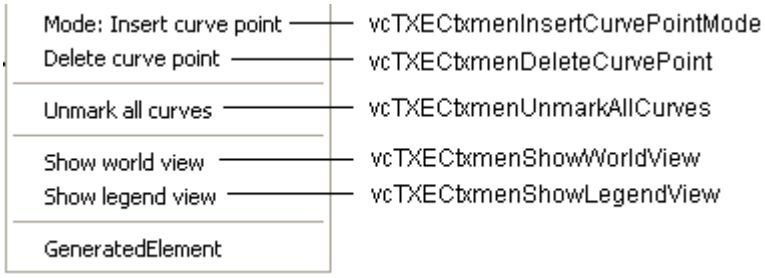
Constants of the context menu for nodes



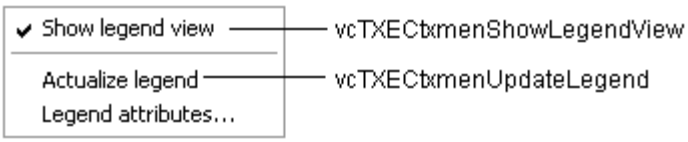
Constants of the context menu for groups with no groups collapsed



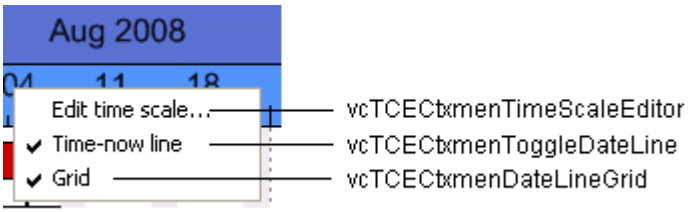
Constants of the context menu for groups with no groups expanded



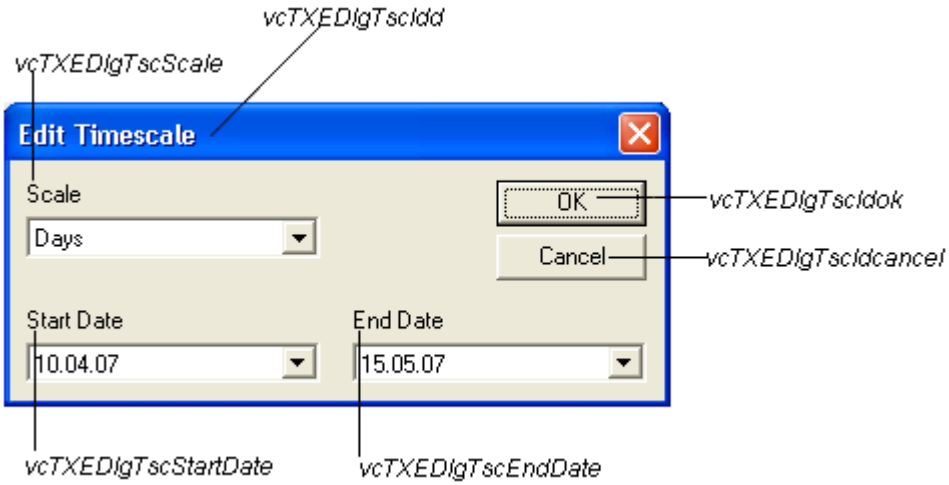
Constants of the context menu for histograms



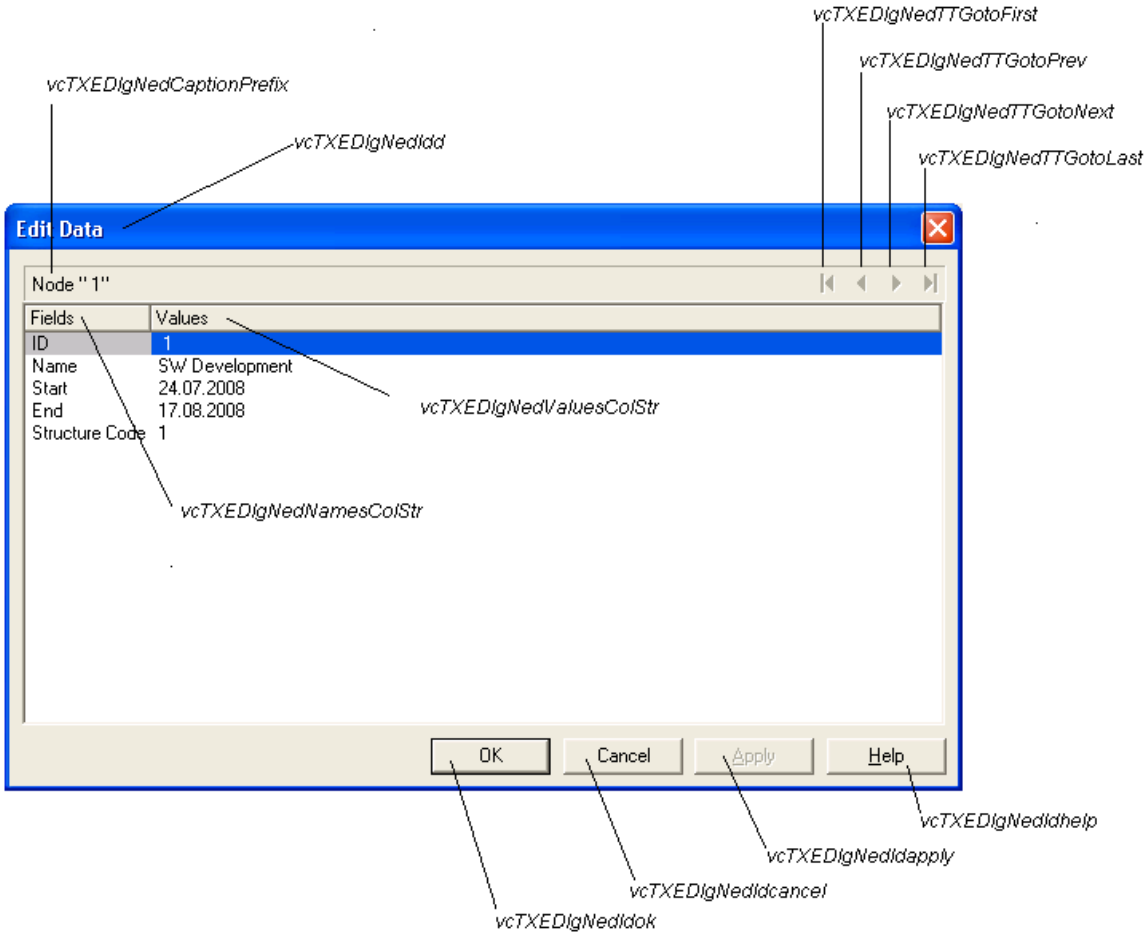
Constants of the legend's context menu



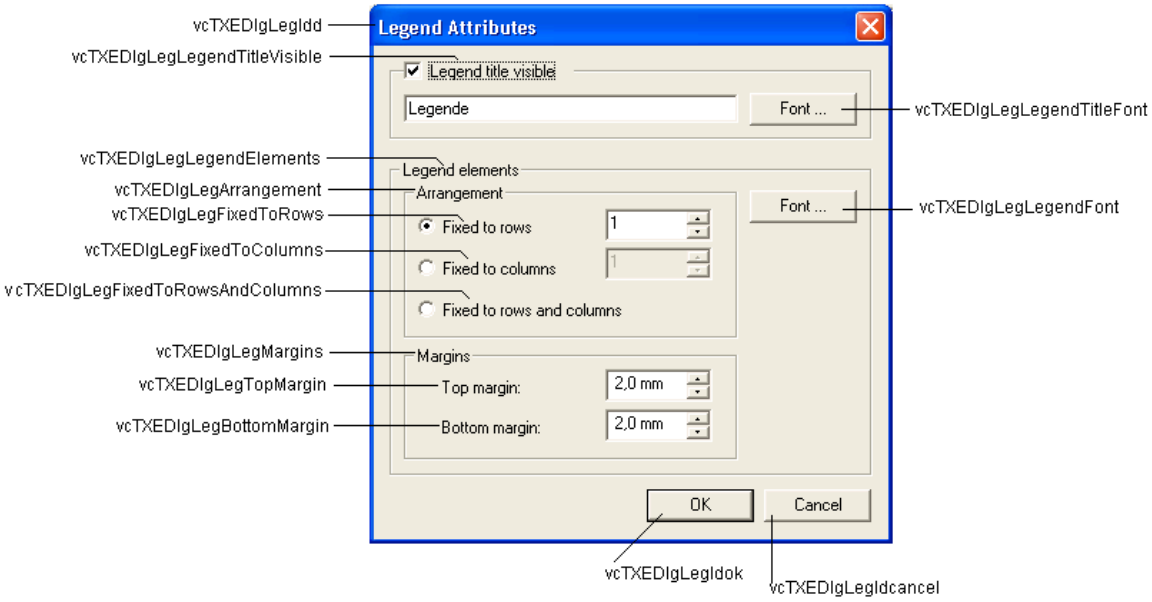
Constants of the time scale's context menu



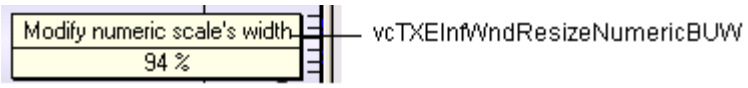
*Constants of the dialog **Edit Time Scale***



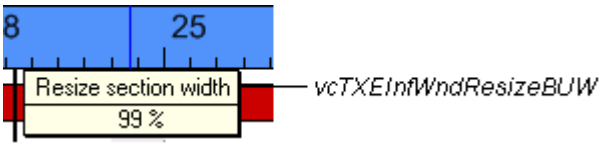
*Constants of the dialogs **Edit data**, **Edit group** and **Edit link**, here illustrated by the **Edit data** dialog*



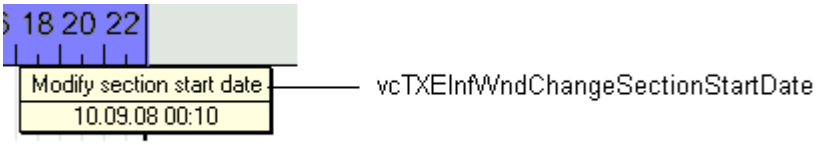
*Constants of the **Legend attributes** dialog*



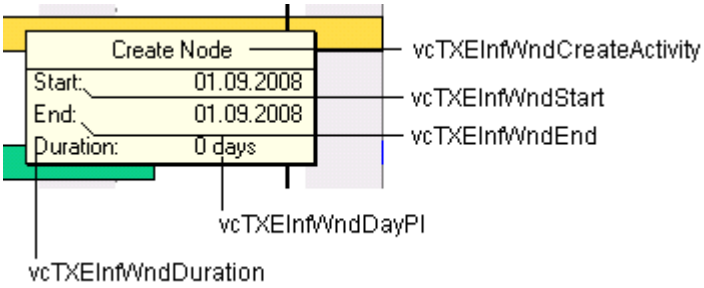
Constant of the tooltip text that appears on resizing the basic unit width of the ***numeric scale in the histogram***



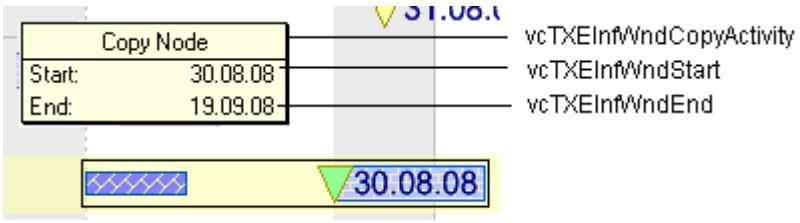
Constants of the tooltip text that appears on resizing the ***time scale section width***



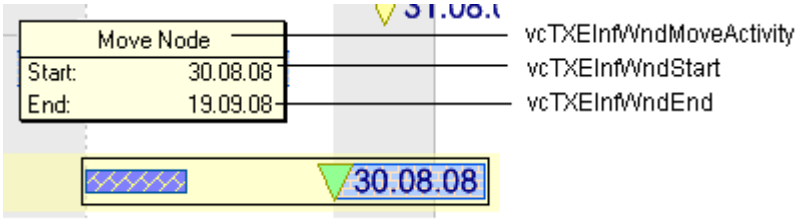
Constants of the tooltip text that appears on modifying the ***start date of a time scale section***



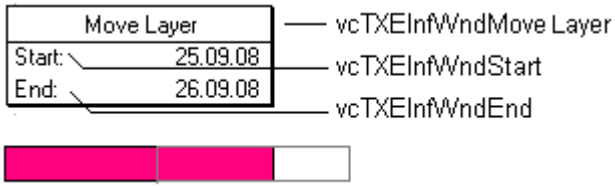
Constants of the tooltip text that appears on ***creating a node***



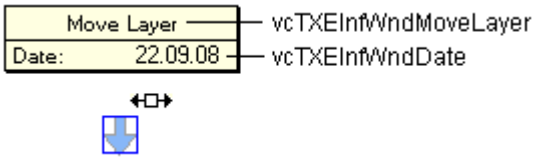
Constants of the tooltip text that appears on ***copying a node***



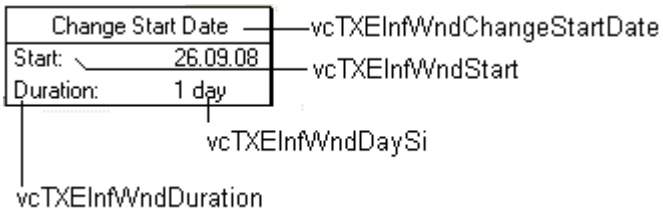
Constants of the tooltip text that appears on ***moving a node***



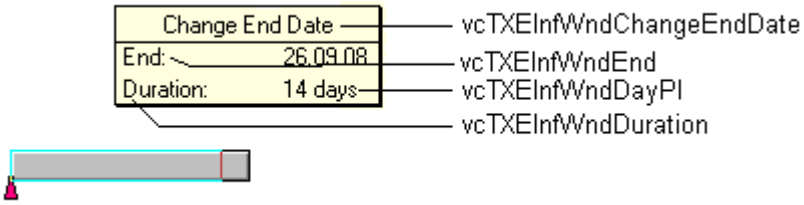
Constants of the tooltip text that appears on ***moving a layer***



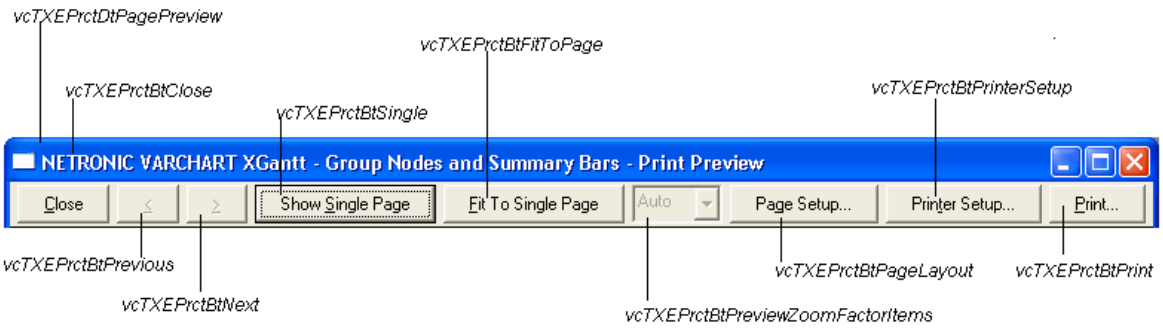
Constants of the tooltip text that appears on ***moving a symbol layer***



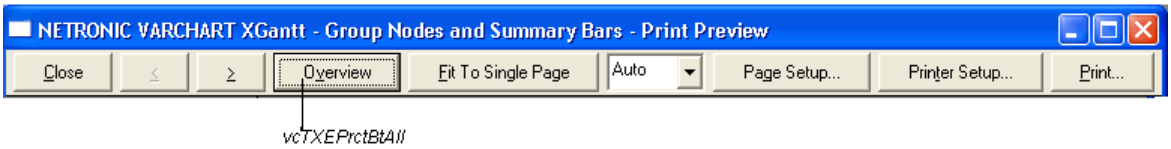
Constants of the tooltip text that appears on modifying the ***start date of a node***



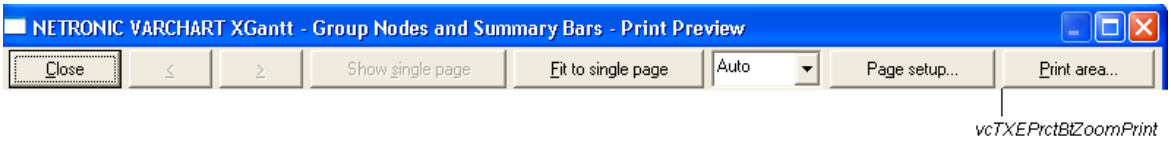
Constants of the tooltip text that appears on modifying the ***end date of a node***



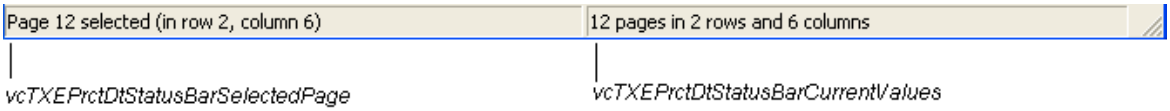
Constants of the button texts of the ***Print preview Overview***



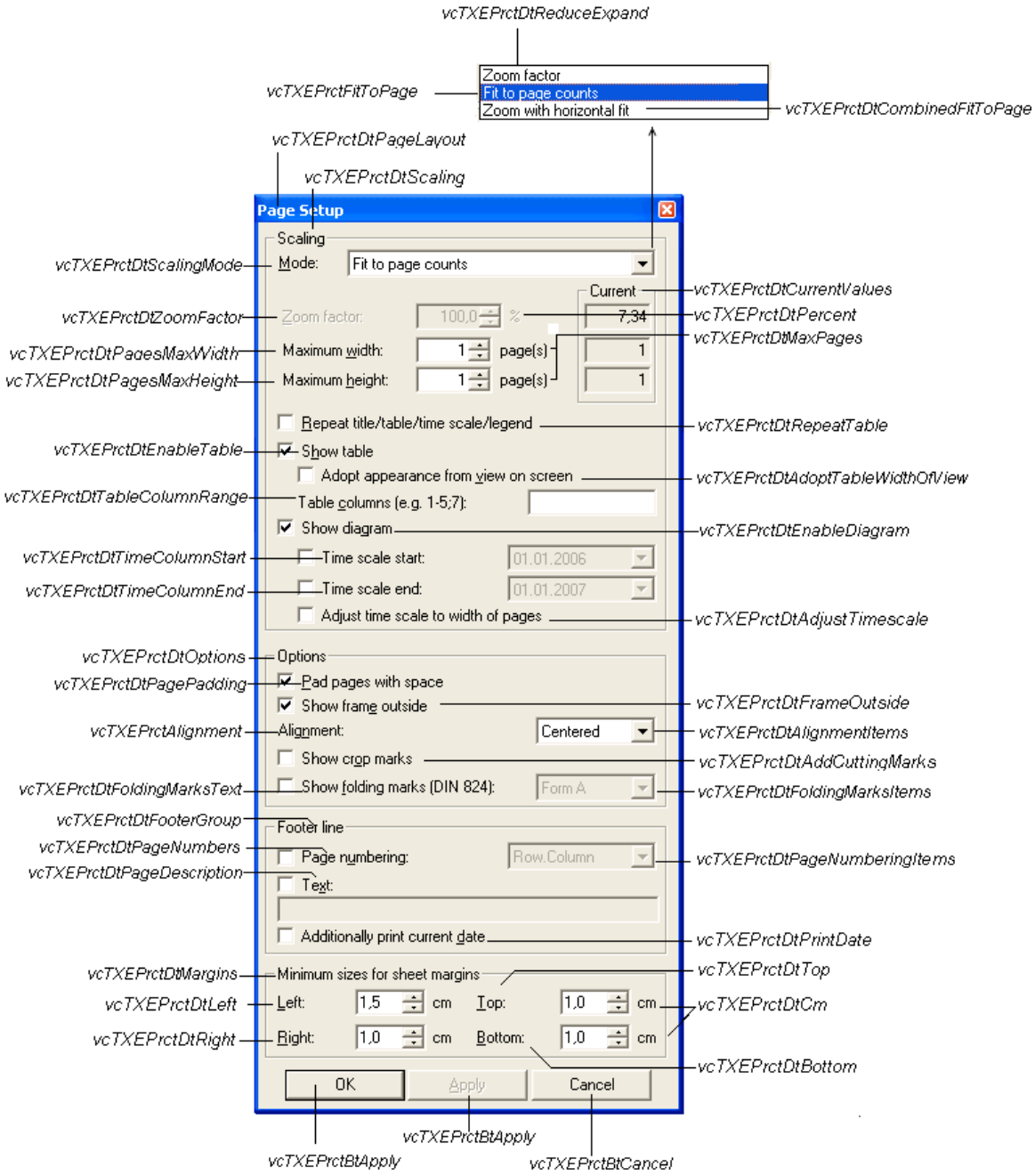
*Constants of the button texts of the **Print Preview** dialog*



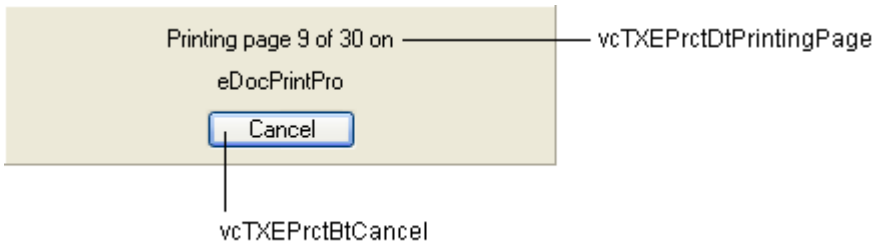
*Constants of the button texts of the **Print Preview** dialog*



*Constants of the status bar in the dialog **Print Preview***



*Constants of the **Page Setup** dialog*



*Constants of the info box **Printing***

Example Code

```
Private Sub VcGantt1_OnSupplyTextEntry(ByVal controlIndex As _
                                VcGanttLib.TextEntryIndexEnum, _
                                TextEntry As String, _
                                returnStatus As Variant)
        Select Case controlIndex
        Case vcTXEPrctBtNext
            TextEntry = "Next page"
        Case vcTXEPrctBtPrevious
            TextEntry = "Previous page"
        Case vcTXEErrTxtInvalidNodePosition
            TextEntry = " Invalid Node Position !"
        End Select
End Sub
```

OnSupplyTextEntryAsVariant

Event of VcGantt

This event is identical with the event **OnSupplyTextEntry** except for the parameters. It was necessary to implement this event because some languages (e.g. VBScript) can use parameters by Reference (indicated by ⇦) only if the type of these parameters is VARIANT.

OnTableCaptionLClick

Event of VcGantt

This event occurs when the user clicks the left mouse button on a table caption. The table object, the column number and the cursor position (x,y-coordinates) are returned. If the diagram is not grouped or hierachically sorted, the activities will be sorted according to the table column hit.

	Data Type	Explanation
Parameter:		
⇨ table	VcTable	Table hit
⇨ columnNumber	Long	Index of the table column hit
⇨ x	Long	X coordinate of the mouse cursor
⇨ y	Long	Y coordinate of the mouse cursor
⇦ returnStatus	Variant	Return staus

Example Code

```
Private Sub VcGantt1_OnTableCaptionLClick(ByVal Table As _
                                VcGanttLib.VcTable, ByVal columnNumber _
                                As Long, ByVal x As Long, ByVal y As Long, _
                                returnStatus As Variant)

    For i = 1 To 2
```

```
        VcGantt1.Table.FormatCollection.FirstFormat.FieldBackgroundColor(i) = _
                                RGB(222, 211, 33)
    Next i

End Sub
```

OnTableCaptionLDblClick

Event of VcGantt

This event occurs when the user double-clicks the left mouse button on a table heading. The table object, the column number and the cursor position (x,y-coordinates) are returned.

	Data Type	Explanation
Parameter:		
⇨ table	VcTable	Table hit
⇨ columnNumber	Long	Index of the column hit
⇨ x	Long	X coordinate of the mouse cursor
⇨ y	Long	Y coordinate of the mouse cursor
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcGantt1_OnTableCaptionLDblClick(ByVal Table As _
                                VcGanttLib.VcTable, ByVal columnNumber _
                                As Long, ByVal x As Long, _
                                ByVal y As Long, _
                                returnStatus As Variant)

    VcGantt1.Table.Visible = False

End Sub
```

OnTableCaptionRClick

Event of VcGantt

This event occurs when the user presses the right mouse button on a table title. The table object, the column number and the cursor position (x,y-coordinates) are returned. By setting the return status you can inhibit the integrated context menu to pop up and replace it by a context menu of your own at the coordinates delivered.

	Data Type	Explanation
Parameter:		
⇨ table	VcTable	Table hit
⇨ columnNumber	Long	Index of the hit table

⇨ x	Long	X coordinate of the mouse cursor
⇨ y	Long	Y coordinate of the mouse cursor
⇔ returnStatus	Variant	Return status
	Possible Values: vcRetStatNoPopup 4 vcRetStatOK 1	 The context menu will be inhibited. The context menu will appear.

Example Code

```
Private Sub VcGantt1_OnTableCaptionRClick(ByVal Table As _
                                VcGanttLib.VcTable, ByVal columnNumber _
                                As Long, ByVal x As Long, _
                                ByVal y As Long, _
                                returnStatus As Variant)

   'start a popup menu at the current cursor position
    PopupMenu mnuTableCaptionPopup

End Sub
```

OnTableColumnWidth

Event of VcGantt

This event occurs when the user modifies the width of a table column. The table, the index and the current width (as 1/100 mm) of the modified column are returned. By setting the return status, you can inhibit the modification.

	Data Type	Explanation
Parameter:		
⇨ table	VcTable	Table
⇨ index	Integer	index of the column modified
	Possible Values:	Data field index
⇨ currentWidth	Long	New column width
⇔ returnStatus	Variant	Return status
	Possible Values: vcRetStatFalse 0 vcRetStatOK 1	 The width of the table column will not be modified. The width of the table column will be modified.

Example Code

```
Private Sub VcGantt1_OnTableColumnWidth(ByVal Table As _
                                VcGanttLib.VcTable, ByVal index As Integer, _
                                ByVal currentWidth As Long, _
                                returnStatus As Variant)

    If currentWidth > 5000 Then
        returnStatus = vcRetStatFalse
        VcGantt1.Table.ColumnWidth(index) = 5000
    End If
```

```
End Sub
```

OnTableColumnWidthModifyComplete

Event of VcGantt

This event occurs when the user has modified the width of a table column. The table, the index and the current width (as 1/100 mm) of the modified column are returned. By setting the return status, you can inhibit the modification.

	Data Type	Explanation
Parameter:		
⇨ table	VcTable	Table
⇨ index	Integer	index of the column modified
	Possible Values:	Data field index
⇨ currentWidth	Long	New column width
⇔ returnStatus	Variant	Return status
	Possible Values: vcRetStatFalse 0 vcRetStatOK 1	 The width of the table column will not be modified. The width of the table column will be modified.

OnTableWidth

Event of VcGantt

This event occurs when the user modifies the width of the table. The table and the modified table/diagram aspect ratio are returned. By setting the return status you can inhibit the modification.

	Data Type	Explanation
Parameter:		
⇨ table	VcTable	Table
⇨ tableWidthRatio	Long	Ratio of the table width to the width of the the total diagram (including table)
⇔ returnStatus	Variant	Return status
	Possible Values: vcRetStatFalse 0 vcRetStatOK 1	 The width of the table will not be modified. The width of the table will be modified.

Example Code

```
Private Sub VcGantt1_OnTableWidth(ByVal Table As VcGanttLib.VcTable, _
                                  ByVal tableWidthRatio As Long, _
                                  returnStatus As Variant)

      If tableWidthRatio > 30 Then
        returnStatus = vcRetStatFalse
        VcGantt1.TableDiagramWidthRatio = 30
    End If

End Sub
```

OnTableWidthModifyEx

Event of VcGantt

This event occurs when the user modifies the width of the table. The table and the modified table/diagram aspect ratio are returned. By setting the return status you can inhibit the modification.

In contrast to the **OnTableWidth** event this event returns the parameter *tableWidthRatio* as "Double" value, thus achieving a higher level of accuracy. The usage of this event has to be enabled by the **UseHigherTable-DiagramWidthRatioPrecision** property or by activating the corresponding option on the **General** property page.

	Data Type	Explanation
Parameter:		
⇨ table	VcTable	Table
⇨ tableWidthRatio	Double	Ratio of the table width to the width of the the total diagram (including table)
⇔ returnStatus	Variant	Return status

OnTimeScaleChangeComplete

Event of VcGantt

This event occurs after zoomng of the time scale was completed.

	Data Type	Explanation
Parameter:		
⇨ timeScale	VcTimeScale	Time scale modified

OnTimeScaleEndModifyComplete

Event of VcGantt

This event occurs after the modification of the end date of the time scale was completed.

	Data Type	Explanation
Parameter:		
⇨ newEndDate	Date	New end date

OnTimeScaleLClick

Event of VcGantt

This event occurs when the user clicks the left mouse button on the time scale. The TimeScale object and the mouse position (x,y-coordinates) are returned.

	Data Type	Explanation
Parameter:		
⇨ timeScale	VcTimeScale	Time scale hit
⇨ x	Long	X coordinate of the mouse cursor
⇨ y	Long	Y coordinate of the mouse cursor
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcGantt1_OnTimeScaleLClick(ByVal timeScale As _
                                VcGanttLib.VcTimeScale, ByVal x As Long, _
                                ByVal y As Long, returnStatus As Variant)

VcGantt1.TimeScaleCollection.Active.BackgroundColor = RGB(225, 50, 10)

End Sub
```

OnTimeScaleLDblClick

Event of VcGantt

This event occurs when the user double-clicks the left mouse button on the time scale. The TimeScale object and the mouse position (x,y-coordinates) are returned. By setting the return status the appearance of the integrated dialog can be inhibited.

	Data Type	Explanation
Parameter:		
⇨ timeScale	VcTimeScale	Time scale hit
⇨ x	Long	X coordinate of the mouse cursor
⇨ y	Long	Y coordinate of the mouse cursor
⇔ returnStatus	Variant	Return status
	Possible Values: vcRetStatFalse 0 vcRetStatOK 1	 The **Edit time scale** dialog will not appear. The **Edit time scale** dialog will appear.

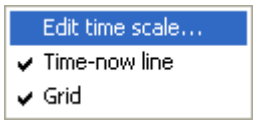
Example Code

```
Private Sub VcGantt1_OnTimeScaleLDblClick(ByVal timeScale As _
                                VcGanttLib.VcTimeScale, ByVal x As Long, _
                                ByVal y As Long, returnStatus As Variant)

' show own "Edit Timescale" dialog
    On Error GoTo CancelError
    frmEditDialog.setTimescale timeScale
    frmEditDialog.Show Modal, Me

    returnStatus = vcRetStatFalse

    Exit Sub

CancelError:
    returnStatus = vcRetStatFalse

End Sub
```

OnTimeScaleRClick

Event of VcGantt

This event occurs when the user clicks the right mouse button on the time scale. The TimeScale object and the mouse position (x,y-coordinates) are returned. At this position you can show your customized context menu. If you set the returnStatus to vcRetStatNoPopup, the integrated context menu will be revoked.



Above: integrated context menu

	Data Type	Explanation
Parameter:		
⇨ timeScale	VcTimeScale	Time scale hit

⇨ x	Long	X coordinate of the mouse cursor
⇨ y	Long	Y coordinate of the mouse cursor
⇔ returnStatus	Variant	Return status
	Possible Values: vcRetStatNoPopup 4 vcRetStatOK 1	 The context menu will be inhibited. The context menu will appear.

Example Code

```
Private Sub VcGantt1_OnTimeScaleRClick(ByVal timeScale As _
                                VcGanttLib.VcTimeScale, _
                                ByVal x As Long, ByVal y As Long, _
                                returnStatus As Variant)

     ' Start own popup menu at the current cursor position
        PopupMenu mnuTimescalePopup

       returnStatus = vcRetStatNoPopup

End Sub
```

OnTimeScaleSectionRescaleCompleteEx

Event of VcGantt

This event occurs when the user has finished rescaling a time scale section. The time scale object, the section index and the new basicUnitWidth are passed.

	Data Type	Explanation
Parameter:		
⇨ timeScale	VcTimeScale	Timescale
⇨ sectionIndex	Integer	Section index
	Possible Values:	Data field index
⇨ newBasicUnitWidth	Long	New width of the basic unit

OnTimeScaleSectionRescaleEx

Event of VcGantt

This event occurs when the user rescales a section of the time scale. The TimeScale object, the section index and the current BasicUnitWidth are returned. By setting the return status you can inhibit the modification.

	Data Type	Explanation
Parameter:		
⇨ timeScale	VcTimeScale	Time scale
⇨ sectionIndex	Integer	Section index
	Possible Values:	Data field index
⇨ newBasicUnitWidth	Long	New width of the basic unit
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcGantt1_OnTimeScaleSectionRescaleEx(ByVal timeScale As _
                                VcGanttLib.VcTimeScale, _
                                ByVal sectionIndex As Integer, _
                                ByVal newBasicUnitWidth As Long, _
                                returnStatus As Variant)

    If newBasicUnitWidth <= 1000 Then
        MsgBox "New basic unit width: " & newBasicUnitWidth
    Else
        MsgBox "The maximum basic unit width is 1000."
        returnStatus = vcRetStatFalse
    End If

End Sub
```

OnTimeScaleSectionStartModify

Event of VcGantt

This event occurs when the user modifies the start date of a section interactively. The TimeScale object, the section index and the current start date are returned.

The data passed by this event can be read, but must not be modified. For modifying them please use **OnTimeScaleSectionStartModifyComplete**.

By setting the return status the modification can be inhibited.

	Data Type	Explanation
Parameter:		
⇨ timeScale	VcTimeScale	Time scale
⇨ sectionIndex	Integer	Section index
	Possible Values:	Data field index
⇨ newStartDate	Date/Time	Date
⇨ returnStatus	Variant	Return status
	Possible Values:	

vcRetStatFalse 0	The modification will be revoked.
vcRetStatOK 1	The modification will be accepted.

Example Code

```
Private Sub VcGantt1_OnTimeScaleSectionStartModify(ByVal timeScale As _
                                VcGanttLib.VcTimeScale, _
                                ByVal sectionIndex As Integer, _
                                ByVal newStartDate As Date, _
                                returnStatus As Variant)

    If MsgBox("Do you want to change the start of section No. " & sectionIndex _
                                & " to " & newStartDate & "?", vbOKCancel) _
                                = vbCancel Then
        returnStatus = vcRetStatFalse
    End If

End Sub
```

OnTimeScaleStartModifyComplete

Event of VcGantt

This event occurs after the modification of the start date of the time scale was completed.

	Data Type	Explanation
Parameter:		
⇨ newStartDate	Date	New start date

OnToolTipText

Event of VcGantt

This event only occurs when the VcGantt property **ShowToolTip** is set to **True** or when the check box **Show tooltip** on the **General** property page is activated. You can use this event for displaying information on the object hit by tooltip texts. The event occurs when the cursor moves on a VcGantt object. The event returns the object, the object type and the coordinates of the mouse position. By setting the returnStatus to **vcRetStatFalse** you can revoke the tooltip.

In case of a calendar grid, a tool tip text will only be retrieved if the calender grid could be identified; i.e. if the calendar grid property **Identifiable** had been set to **True**.

	Data Type	Explanation
Parameter:		
⇨ hitObject	Object	Object hit
⇨ hitObjectType	VcObjectTypeEnum	Type of the object hit
	Possible Values:	
	vcObjTypeBox 15	object type **box**
	vcObjTypeCalendarGrid 18	object type **calendar grid**
	vcObjTypeCurve 12	object type **curve**
	vcObjTypeDateLine 9	object type **date line**
	vcObjTypeGroup 7	object type **group**
	vcObjTypeGroupInDiagram 11	object type **group in diagram area**
	vcObjTypeGroupInTable 7	object type **group in table area**
	vcObjTypeHistogram 13	object type **histogram**
	vcObjTypeLayer 8	object type **layer**
	vcObjTypeLinkCollection 3	object type **link collection**
	vcObjTypeNodeInDiagram 2	object type **node in diagram area**
	vcObjTypeNodeInLegend 17	object type **node in legend area**
	vcObjTypeNodeInTable 1	object type **node in table area**
	vcObjTypeNone 0	no object
	vcObjTypeNumericScale 10	object type **numeric scale**
	vcObjTypeSummaryNode 14	object type **summary bar**
	vcObjTypeTable 4	object type **table**
	vcObjTypeTableCaption 5	object type **table caption**
	vcObjTypeTimeScale 6	object type **time scale**
⇨ x	Long	X coordinate of the mouse cursor
⇨ y	Long	Y value of the mouse cursor
⇦ ToolTipText	String	Tooltip text, can contain 1024 characters maximum
	Possible Values:	Name of the color map
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcGantt1_OnToolTipText(ByVal hitObject As Object, _
                                   ByVal hitObjectType As _
                                   VcGanttLib.VcObjectTypeEnum, ByVal x As Long, _
                                   ByVal y As Long, ToolTipText As String, _
                                   returnStatus As Variant)

    Select Case hitObjectType
        Case vcObjTypeNodeInDiagram
            ToolTipText = hitObject.DataField(1)

        Case Else
    End Select

End Sub
```

OnToolTipTextAsVariant

Event of VcGantt

This event is identical with the event **OnToolTipText** except for the parameters. It was necessary to implement this event because some languages

(e.g. VBScript) can use parameters by Reference (indicated by ⇦) only if the type of these parameters is VARIANT.

Example Code

```
Private Sub VcGantt1_OnToolTipTextAsVariant(ByVal hitObject As Object, ByVal
hitObjectType As VcGanttLib.VcObjectTypeEnum, ByVal x As Long, ByVal y As Long,
ToolTipText As Variant, returnStatus As Variant)
    Select Case hitObjectType
        Case VcObjectTypeEnum.vcObjTypeNodeInDiagram
            ToolTipText = hitObject.DataField(0)
    End Select
End Sub
```

OnViewComponentsSizeModifyComplete

Event of VcGantt

This event occurs when at run time the size of a graphical element of the VARCHART ActiveX control (time scale, diagram, histogram, table, table caption etc.) was modified. To react to the event by API, you need to retrieve the position and the size of all graphical elements of the VARCHART ActiveX control.

Note:

1. The position refers to the origin of the graphical element of the VARCHART ActiveX control.

2. The values returned are pixel values.

	Data Type	Explanation
Parameter: ⇨ (no parameter)		

Example Code

```
Private Sub VcGantt1_OnViewComponentsSizeModifyComplete()
   Dim x As Long
   Dim y As Long
   Dim width As Long
   Dim height As Long
   Dim scMod As Long
   scMod = ScaleMode
   ScaleMode = vbPixels

   VcGantt1.GetViewComponentSize vcHistogramVerScaleComponent, x, y, _
                       width, height

   ' plus 6 because of the sash
   Text1.Top = VcGantt1.Top + y + 6
   Text1.Left = VcGantt1.Left + x
   ' minus 25 because of the numeric scale
   Text1.width = width - 25
```

```
   ' minus 6 because of the sash
   Text1.height = height - 6
   ScaleMode = scMod
End Sub
```

OnWorldViewClosed

Event of VcGantt

This event occurs when the worldview popup window is closed.

	Data Type	Explanation
Parameter: ⇦ (no parameter)		

Example Code

```
Private Sub VcGantt1_OnWorldViewClosed()
    MsgBox "Do you want to close the worldview window?", vbOKCancel
End Sub
```

OnZoomFactorModifyComplete

Event of VcGantt

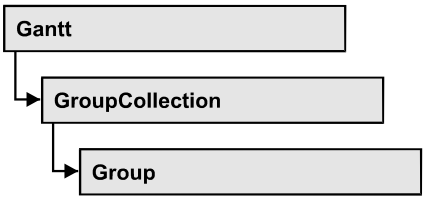
This events occurs if the user modified the size of the rectangle in the world view or if he zoomed marked objects. You can zoom smoothly by keeping the **Ctrl** key pressed while turning the mouse wheel, or in discrete steps while using the **Plus** or **Minus** keys in the number pad.

	Data Type	Explanation
Parameter: ⇦ (no parameter)		

Example Code

```
Private Sub VcGantt1_OnZoomFactorModifyComplete()
     MsgBox "Zoomfactor: " & VcGantt1.ZoomFactor
End Sub
```

7.38 VcGroup



A group contains all nodes that have the same value in the grouping field. This value can be retrieved as group name. The nodes that form a group can be accessed by the NodeCollection property.

Properties

- BodyCollapsed
- DataField
- GroupingLevel
- GroupInvisible
- ID
- MarkGroup
- Name
- NodeCollection
- NodesInHeader
- NodesOverlaid
- RowsBelowCollapsed
- SubGroups
- SuperGroup
- Visible

Methods

- DataRecord
- DeleteGroup
- RelatedDataRecord
- ReOptimizeNodes
- UpdateGroup

Properties

BodyCollapsed

Property of VcGroup

This property lets you set or retrieve, whether (True) or not (False) a group is collapsed.

	Data Type	Explanation
Property value	Boolean	Group collapsed/expanded
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
Private Sub VcGantt1_OnGroupLClick(ByVal group As VcGanttLib.VcGroup,  _
                                   ByVal x As Long, ByVal y As Long,  _
                                   returnStatus As Variant)

   If body.Collapsed = False Then
      body.Collapsed = True
   Else
      body.Collapsed = False
   End If

End Sub
```

DataField

Property of VcGroup

This property lets you set or retrieve the contents of a DataField of the group record. The group record is copy of the node record of the first node added to the group. The data field is referred to by its field index. To update the group, the **UpdateGroup** method needs to be invoked.

	Data Type	Explanation
Parameter: ⇨ index	Integer	Index of the data field
	Possible Values:	Data field index
Property value	Void	

Example Code

```
Dim groupCltn As VcGroupCollection
```

```
Dim group As VcGroup
Dim nodeCltn As VcNodeCollection
Dim node As VcNode

Set groupCltn = VcGantt1.groupCollection

For Each group In groupCltn
    Set nodeCltn = group.nodeCollection

For Each node In nodeCltn
     If node.DataField(3) > group.DataField(3) Then
         group.DataField(3) = node.DataField(3)
     End If
Next node

group.UpdateGroup
Next group
```

GroupingLevel

Read Only Property of VcGroup

This property lets you enquire the grouping level of the group, if there are several levels of grouping. At maximum, 25 grouping levels are possible.

	Data Type	Explanation
Property value	Integer	Grouping level of the group
	Possible Values:	Data field index

Example Code

```
Dim group As VcGroup
Dim superGroup As VcGroup

Dim nodeCollection As VcNodeCollection
Dim node As VcNode

Set nodeCollection = VcGantt1.nodeCollection
Set node = nodeCollection.FirstNode
Set group = node.SuperGroup

If group.GroupingLevel > 0 Then
    Set superGroup = group.superGroup
End If
```

GroupInvisible

Property of VcGroup

This property lets you set or retrieve whether this group is to be displayed. The default value is the value that was specified in the group level layout.

	Data Type	Explanation

ID

Read Only Property of VcGroup

By this property you can retrieve the ID of a group.

	Data Type	Explanation
Property value	String **Possible Values:**	Group ID Name of the color map

Example Code

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup
Dim groupName As String
Set groupCltn = VcGantt1.GroupCollection
Set group = groupCltn.FirstGroup
groupID = group.ID
MsgBox group.ID
```

MarkGroup

Property of VcGroup

This property lets you set or retrieve whether a group is marked.

	Data Type	Explanation

Example Code

```
Dim groupCltn As VcGroupCollection
Dim group As VcNode
Set nodeCltn = VcGantt1.nodeCollection
nodeCltn.SelectNodes (vcSelected)

For Each node In nodeCltn
      Group.MarkGroup = False
 Next node
```

Name

Read Only Property of VcGroup

This property lets you retrieve the name of a group (= the value of the grouping field GroupField).

	Data Type	Explanation
Property value	String	Group name
	Possible Values:	Name of the color map

Example Code

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup
Dim groupName As String

Set groupCltn = VcGantt1.GroupCollection
Set group = groupCltn.FirstGroup

groupName  = group.Name
```

NodeCollection

Read Only Property of VcGroup

This property gives access to each node of a group.

	Data Type	Explanation
Property value	VcNodeCollection	NodeCollection object

Example Code

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup
Dim nodeCltn As VcNodeCollection

Set groupCltn = VcGantt1.GroupCollection
Set group = groupCltn.FirstGroup
Set nodeCltn = group.NodeCollection
```

NodesInHeader

Property of VcGroup

This property lets you set or retrieve whether (True) or not (False) the node objects of the group are positioned the same row.

	Data Type	Explanation

Example Code

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup

Set groupCltn = VcGantt1.GroupCollection
Set group = groupCltn.GroupByName("A")

group.NodesInHeader = True
```

NodesOverlaid

Property of VcGroup

This property lets you set or retrieve whether (False) the node layout is optimized or if nodes overlap (True).

	Data Type	Explanation
Property value	Boolean **Possible Values:**	The node layout is/is not at its optimum Group invisible/visible group nodes are/are not visible

RowsBelowCollapsed

Property of VcGroup

This property applies to multi-level grouping (n levels), that is, to the levels from no.1 to (n-1). If you have chosen for the group all nodes in one row, setting this property to **True** will collapse only the subgroups of the selected group. If instead you collapse the group by the **Collapsed** property, in addition groups that do not belong to a subgroup will be collapsed as well.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	Rows below the top row are/are not collapsed Group invisible/visible group nodes are/are not visible

Example Code

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup

Set groupCltn = VcGantt1.GroupCollection
```

```
Set group = groupCltn.GroupByName("A")

group.RowsBelowCollapsed = True
```

SubGroups

Read Only Property of VcGroup

In a multi-level grouping arrangement, this property lets you enquire subgroups, that are returned by a group collection object.

	Data Type	Explanation
Property value	VcGroupCollection	GroupCollection object containing the subgroups

Example Code

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup
Dim subGroupCltn As VcGroupCollection

Set groupCltn = VcGantt1.GroupCollection
Set group = groupCltn.GroupByName("A")

Set subGroupCltn = group.SubGroups
```

SuperGroup

Read Only Property of VcGroup

In a multi-level grouping arrangement, this property lets you enquire the parent group of this group.

	Data Type	Explanation
Property value	VcGroup	Parent group

Example Code

```
Dim group As VcGroup
Dim superGroup As VcGroup

Dim nodeCollection As VcNodeCollection
Dim node As VcNode

Set nodeCollection = VcGantt1.NodeCollection
Set node = nodeCollection.FirstNode
Set group = node.SuperGroup

If group.GroupingLevel > 0 Then
    Set superGroup = group.SuperGroup
End If
```

Visible

Property of VcGroup

This property lets you set or retrieve whether (True) or not (False) this group is visible.

	Data Type	Explanation
Property value	Boolean	Group visible/invisible
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup

Set groupCltn = VcGantt1.GroupCollection
Set group = groupCltn.GroupByName("A")

group.Visible = False
```

Methods

DataRecord

Method of VcGroup

This property lets you retrieve the group as a data record object. The properties of the data record object give access to the corresponding data table and the data table collection.

	Data Type	Explanation
Return value	VcDataRecord	Data record returned

DeleteGroup

Method of VcGroup

This method lets you delete a group. Deleting a group is only possible when it doesn´t contain any activity. Possibly you have to delete all activities of the group before you can delete the group.

	Data Type	Explanation
Return value	Boolean	Group was (True) / was not (False) deleted successfully

Example Code

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup
Dim nodeCltn As VcNodeCollection
Dim node As VcNode

Set groupCltn = VcGantt1.GroupCollection
Set group = groupCltn.GroupByName ("A")
Set nodeCltn = group.NodeCollection

For Each node In nodeCltn
   node.DeleteNode
Next node

group.DeleteGroup
```

RelatedDataRecord

Method of VcGroup

This method lets you retrieve a data record from a data table that is related to the group data table. The index passed by the parameter denotes the field in the data record that holds the key of the related data record.

	Data Type	Explanation
Parameter:		
⇨ index	Integer	Index of data field that holds the key
	Possible Values:	Data field index
Return value	VcDataRecord	Related data record returned

ReOptimizeNodes

Method of VcGroup

If the property **VcGantt.GroupOptimizationOnInteractionsEnabled** was set to **false** and if the nodes of the group are in the optimized state of display, this property allows to manually update the optimized arrangement after an interaction.

	Data Type	Explanation
Return value	Void	

UpdateGroup

Method of VcGroup

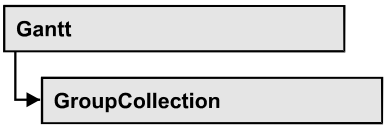
This method lets you update a group after having changed a data field by the **DataField** property.

	Data Type	Explanation
Return value	Boolean	Group successfully/not successfully updated

Example Code

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup
Dim nodeCltn As VcNodeCollection
Dim node As VcNode

Set groupCltn = VcGantt1.GroupCollection
Set group = groupCltn.GroupByName("A")
Set nodeCltn = group.NodeCollection

group.DataField(3) = nodeCltn.FirstNode.DataField(3)

For Each node In nodeCltn
    If node.DataField(3) > group.DataField(3) Then
        group.DataField(3) = node.DataField(3)
    End If
Next node

group.UpdateGroup
```

7.39 VcGroupCollection



If nodes were grouped, an object of the type VcGroupCollection contains all available groups.You can access all objects in an iterative loop by **For Each group In GroupCollection** or by the methods **First...** and **Next...**. You can access a single group using the method **GroupByName**. The number of groups in the collection object can be retrieved by the property **Count**.

Properties

- _NewEnum
- Count

Methods

- FirstGroup
- GroupByName
- NextGroup
- SelectGroups

Properties

_NewEnum

Read Only Property of VcGroupCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all group objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method **GetEnumerator** is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim group As VcGroup
```

```
For Each group In VcGantt1.GroupCollection
   Debug.Print group.Name
Next
```

Count

Read Only Property of VcGroupCollection

This property lets you retrieve the number of groups in the group collection.

	Data Type	Explanation
Property value	Long	Number of nodes

Example Code

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup
Dim numberOfGroups As Integer

Set groupCltn = VcGantt1.GroupCollection
numberOfGroups = groupCltn.Count
```

Methods

FirstGroup

Method of VcGroupCollection

This method can be used to access the initial value, i.e. the first group of a group collection, and then to continue in a forward iteration loop by the method **NextGroup** for the groups following. If there is no group in the group collection, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcGroup	First group of the GroupCollection

Example Code

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup

Set groupCltn = VcGantt1.GroupCollection
Set group = groupCltn.FirstGroup
```

GroupByName

Method of VcGroupCollection

By this method you can get a group by its name. If a group of the specified name does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter:		
⇨ Rückgabewert	VcGroup	Group
⇨ groupName	String	Name of group
	Possible Values:	Name of the color map
Return value	VcGroup	Group

Example Code

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup

Set groupCltn = VcGantt1.GroupCollection
Set group = groupCltn.GroupByName ("Group A")
```

NextGroup

Method of VcGroupCollection

This method can be used in a forward iteration loop to retrieve subsequent groups from a group collection after initializing the loop by the method **FirstGroup**. If there is no group left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcGroup	Subsequent group

Example Code

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup

Set groupCltn = VcGantt1.GroupCollection
Set group = groupCltn.FirstGroup

While Not group Is Nothing
    List1.AddItem group.Name
    Set group = groupCltn.NextGroup
Wend
```

SelectGroups

Method of VcGroupCollection

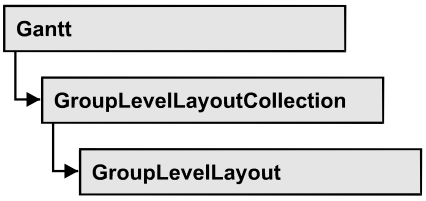
This method lets you specify the groups that the group collection is to contain.

	Data Type	Explanation
Parameter:		
⇨ groupSelType	GroupSelectionTypeEnum	Type of group to be selected
	Possible Values:	
	vcAllGroups 0	All groups selected
	vcCollapsedGroups 1	Collapsed groups selected
	vcExpandedGroups 2	Expanded groups selected
	vcInvisibleGroups 5	Invisible groups selected
	vcSelectedGroups 3	Selected groups selected
	vcVisibleGroups 4	Visible groups selected
Return value	Long	Number of groups selected

Example Code

```
Dim groupCltn As VcGroupCollection

Set groupCltn = VcGantt1.GroupCollection
groupCltn.SelectGroups (vcAllGroups)
```

7.40 VcGroupLevelLayout



An object of the type VcGroupLevelLayout defines the content and the appearance of grouping levels. For this, the name of the grouping level, the level number, the grouping field, sorting and sorting order can serve, as well as various options concerning the design of calendar and line grids and of separation lines.

Properties

- AllowVerticalGroupMovementViaDiagram
- AllowVerticalGroupMovementViaTable
- AutoCollapseGroups
- AutoExpandTargetGroup
- BodiesCollapsed
- BodiesCollapsedDataFieldIndex
- BodiesCollapsedMapName
- CalendarGridName
- CalendarGridsWithChildGroups
- CalendarNameDataFieldIndex
- DateLineGridName
- DateLineGridsWithChildGroups
- DateLineName
- DateLinesWithChildGroups
- GroupDataFieldIndex
- GroupsInvisible
- GroupsInvisibleCollapsedMapName
- GroupsInvisibleDataFieldIndex
- Level
- ModificationsAllowed
- Name
- NodesInHeaders
- NodesInHeadersDataFieldIndex
- NodesInHeadersMapName
- NodesOverlaid
- OptimizedNodesSortDataFieldIndex

- OptimizedNodesSortOrder
- OverlaidNodesSortDataFieldIndex
- OverlaidNodesSortOrder
- PagebreakMode
- RestoreAutoCollapsedGroups
- RestoreAutoExpandedGroups
- RowBackColorAsARGB
- RowBackColorDataFieldIndex
- RowBackColorMapName
- RowPattern
- RowPatternColorAsARGB
- RowPatternColorDataFieldIndex
- RowPatternColorMapName
- RowPatternDataFieldIndex
- RowPatternMapName
- SeparationLineColor
- SeparationLineColorDataFieldIndex
- SeparationLineColorMapName
- SeparationLineThickness
- SeparationLineType
- ShowCalendarGrids
- ShowDateLineGrids
- ShowDateLines
- ShowGroupNodes
- ShowSeparationLines
- ShowSeparationLinesAtTop
- SortDataFieldIndex
- SortOrder
- Specification
- SummaryBarsVisible
- Visible

Properties

AllowVerticalGroupMovementViaDiagram

Property of VcGroupLevelLayout

This property lets you set or retrieve whether groups are allowed to be moved vertically in the diagram. This property also can be set in the **Grouping** dialog.

	Data Type	Explanation
Property value	Boolean	Vertical group movement in diagram enabled/disabled **Default value:** True
	Possible Values:	Group invisible/visible group nodes are/are not visible

AllowVerticalGroupMovementViaTable

Property of VcGroupLevelLayout

This property lets you set or retrieve whether groups are allowed to be moved vertically in the table. This property also can be set in the **Grouping** dialog.

	Data Type	Explanation
Property value	Boolean	Vertical group movement in table enabled/disabled **Default value:** true
	Possible Values:	Group invisible/visible group nodes are/are not visible

AutoCollapseGroups

Property of VcGroupLevelLayout

This property lets you set or retrieve whether in the group level layout the groups are to becollapsed automatically on interactions.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	Groups are/are not collapsed automatically on interactions Group invisible/visible group nodes are/are not visible

AutoExpandTargetGroup

Property of VcGroupLevelLayout

This property lets you set or retrieve whether in the group level layout the groups are to be expanded automatically on interactions.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	Target groups are/are not expanded automatically on interactions Group invisible/visible group nodes are/are not visible

BodiesCollapsed

Property of VcGroupLevelLayout

This property lets you set or retrieve, whether (True) or not (False) the groups of this group level are collapsed.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	Groups collapsed/expanded Group invisible/visible group nodes are/are not visible

BodiesCollapsedDataFieldIndex

Property of VcGroupLevelLayout

This property lets you set or retrieve the data field index for the collapsed bodies of this grouping level. This property also can be set in the **Grouping** dialog.

	Data Type	Explanation
Property value	Long	This levels groups bodies collapsed data field index

BodiesCollapsedMapName

Property of VcGroupLevelLayout

This property lets you set or retrieve the name of a map for the bodies collapsed on this group level. If set to "" or if the property **Bodies-CollaspsedDataFieldIndex** is set to -1, then no map will be used.

	Data Type	Explanation
Property value	Long	This levels groups bodies collapsed map name

CalendarGridName

Property of VcGroupLevelLayout

This property lets you set or retrieve the name of the calendar grid for this group level layout. You can also set this property in the **Grouping** dialog.

	Data Type	Explanation
Parameter: Rückgabewert	String **Possible Values:**	Name of the calendar grid Name of the color map
Property value	String **Possible Values:**	name of the calendar grid Name of the color map

CalendarGridsWithChildGroups

Property of VcGroupLevelLayout

This property lets you set or retrieve whether calendar grids are also displayed for subgroups. You can also set this property in the **Grouping** dialog.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	Calendar grid for subgroups are/are not displayed Group invisible/visible group nodes are/are not visible

CalendarNameDataFieldIndex

Property of VcGroupLevelLayout

This property lets you set or retrieve the index of the data field for storing the name of the calendar to apply to the group level layout. This is only possible as long as no data was loaded.

	Data Type	Explanation
Property value	Long	Index of the data field which contains the name of the calendar

DateLineGridName

Property of VcGroupLevelLayout

This property lets you set or retrieve the name of the date line grid for this group level layout. You can also set this property in the **Grouping** dialog.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the date line grid Name of the color map

DateLineGridsWithChildGroups

Property of VcGroupLevelLayout

This property lets you set or retrieve whether the date line grids are also displayed for subgroups. You can also set this property in the **Grouping** dialog.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	Date line grids for subgroups are/are not displayed Group invisible/visible group nodes are/are not visible

DateLineName

Property of VcGroupLevelLayout

This property lets you set or retrieve the name of the date line for this group level layout. You can also set this property in the **Grouping** dialog.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the date line Name of the color map

DateLinesWithChildGroups

Property of VcGroupLevelLayout

This property lets you set or retrieve whether the date lines are to be displayed for all group elements. This property also can be set in the **Grouping** dialog.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	Date lines for subgroups are/are not displayed Group invisible/visible group nodes are/are not visible

GroupDataFieldIndex

Property of VcGroupLevelLayout

This property lets you set or retrieve the data field index used for grouping of this VcGroupLevelLayout object.

	Data Type	Explanation
Property value	Long	Index used for grouping of this VcGroupLevelLayout object

GroupsInvisible

Property of VcGroupLevelLayout

This property lets you set or retrieve whether this level's groups are displayed. This property also can be set in the **Grouping** dialog.

	Data Type	Explanation

GroupsInvisibleCollapsedMapName

Property of VcGroupLevelLayout

This property lets you set or retrieve the name of a map for the for the invisible groups on this group level. If set to "" or if the property **Bodies-CollaspsedDataFieldIndex** is set to -1, then no map will be used.

	Data Type	Explanation

GroupsInvisibleDataFieldIndex

Property of VcGroupLevelLayout

This property lets you set or retrieve the data field index for the invisible groups of this grouping level. This property also can be set in the **Grouping** dialog.

	Data Type	Explanation

Level

Read Only Property of VcGroupLevelLayout

This property lets you enquire the grouping level of this group level layout. At maximum, 25 grouping levels are possible.

	Data Type	Explanation
Property value	Integer	Grouping level of the group level layout
	Possible Values:	Data field index

ModificationsAllowed

Property of VcGroupLevelLayout

This property lets you specify whether the user can collapse expanded groups of this level and vice versa. The user can collapse/expand groups by double-clicking on the group heading in the table section, by clicking on the minus or plus sign next to the group heading or by the context menu for groups. This property also can be set in the **Grouping** dialog.

	Data Type	Explanation
Property value	Boolean	Modifications allowed (True)/ not allowed (False)
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
VcGroupLevelLayout.ModificationsAllowed(0) = False
```

Name

Property of VcGroupLevelLayout

This property lets you retrieve the name of a group level layout.

	Data Type	Explanation
Property value	String	Name of the group level
	Possible Values:	Name of the color map

Example Code

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup
Dim groupName As String

Set groupCltn = VcGantt1.GroupCollection
Set group = groupCltn.FirstGroup

groupName  = group.Name
```

NodesInHeaders

Property of VcGroupLevelLayout

This property lets you set or retrieve whether (True) or not (False) the node objects of the group of this level are positioned the same row.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	All nodes of the group are/are not in the same row Group invisible/visible group nodes are/are not visible

NodesInHeadersDataFieldIndex

Property of VcGroupLevelLayout

This property lets you set or retrieve the data field index for the nodes in headers of this grouping level. This property also can be set in the **Grouping** dialog.

	Data Type	Explanation
Property value	Long	Data field index

NodesInHeadersMapName

Property of VcGroupLevelLayout

This property lets you set or retrieve the map name for the nodes in headers on this grouping level. If set to "" or if the property **NodesInHeadersDataFieldIndex** is set to -1, then no map will be used.

	Data Type	Explanation
Property value	Long	Map name

NodesOverlaid

Property of VcGroupLevelLayout

This property lets you set or retrieve whether the node layout on this group level is optimized (False) or if nodes overlap (True).

	Data Type	Explanation
Property value	Boolean **Possible Values:**	The node layout is/is not at its optimum Group invisible/visible group nodes are/are not visible

Example Code

```
group.LevelLayout.NodesOverlaid = True
```

OptimizedNodesSortDataFieldIndex

Property of VcGroupLevelLayout

This property lets you set or retrieve the index of a data field that contains the sorting criterion (the drawing priority) for the display of several nodes in a single row. Setting this property only makes sense if the property **NodesArrangedOptimized** was set to **True**. This property also can be set in the **Grouping** dialog.

	Data Type	Explanation
Property value	Long	Index of the data field that holds the sorting criterion

OptimizedNodesSortOrder

Property of VcGroupLevelLayout

This property lets you set or retrieve the sorting direction of the sorting criterion, which was selected by the property **OptimizedNodesSortDataFieldIndex**. Setting this property only makes sense if the property **NodesArrangedOptimized** was set to **True**. This property also can be set in the **Grouping** dialog.

	Data Type	Explanation
Property value	SortOrderEnum **Possible Values:** vcAscending 1 vcDescending 2	Direction of the sorting order **Default value:** vcAscending ascending order descending order

OverlaidNodesSortDataFieldIndex

Property of VcGroupLevelLayout

This property lets you set or retrieve the index of a data field that contains the sorting criterion (the drawing priority) for the display of several nodes in a single row. Setting this property only makes sense if the property **NodesArrangedOptimized** was set to **False**. This property also can be set in the **Grouping** dialog.

	Data Type	Explanation
Property value	Long	Index of the data field that holds the sorting criterion

OverlaidNodesSortOrder

Property of VcGroupLevelLayout

This property lets you set or retrieve the sorting direction of the sorting criterion, which was selected by the property **OverlaidNodesSortDataFieldIndex**. Setting this property only makes sense if the property **NodesArrangedOptimized** was set to **False**. This property also can be set in the **Grouping** dialog.

	Data Type	Explanation
Property value	SortOrderEnum **Possible Values:** vcAscending 1 vcDescending 2	Direction of the sorting order **Default value:** vcAscending ascending order descending order

PagebreakMode

Property of VcGroupLevelLayout

This property lets you set or retrieve whether and when page breaks after groups are to be carried out. This property also can be set in the **Grouping** dialog.

	Data Type	Explanation
Property value	PagebreakModeEnum	Page break mode **Default value:** vcPagebreakNone
	Possible Values: vcPagebreakAfterEachGroup 1 vcPagebreakNone 0 vcPagebreakOnPageFull 2	 Pagebreak after each group No pagebreak Pagebreak if following group wouldn't fit on page completely

RestoreAutoCollapsedGroups

Property of VcGroupLevelLayout

This property lets you set or retrieve whether in the group level layout automatically collapsed groups are to be restored automatically on interactions.

	Data Type	Explanation
Property value	Boolean	Automatically collapsed groups are/are not restored automatically on interactions
	Possible Values:	Group invisible/visible group nodes are/are not visible

RestoreAutoExpandedGroups

Property of VcGroupLevelLayout

This property lets you set or retrieve whether in the group level layout automatically expanded groups are to be restored automatically on interactions.

	Data Type	Explanation
Property value	Boolean	Automatically expanded groups are/are not restored automatically on interactions
	Possible Values:	

		Group invisible/visible group nodes are/are not visible

RowBackColorAsARGB

Property of VcGroupLevelLayout

This property lets you set or retrieve the background color of the group title row. The default color is white.

	Data Type	Explanation
Property value	Color	ARGB color values ({0...255},{0...255},{0...255},{0...255})

Example Code

```
Dim groupLevelLayoutCltn As VcGroupLevelLayoutCollection
Dim groupLevelLayout  As VcGroupLevelLayout

Set groupLevelLayoutCltn = VcGantt1.GroupLevelLayoutCollection
Set groupLevelLayout = groupLevelLayoutCltn.FirstGroupLevelLayout

groupLevelLayout.RowBackColor = RGB(128, 128, 128)
```

RowBackColorDataFieldIndex

Property of VcGroupLevelLayout

This property lets you set or retrieve the data field index to be used with a color map specified by the property **RowBackColorMapName**. If you set this property to **-1**, no map will be used.

	Data Type	Explanation
Property value	Integer **Possible Values:**	Data field index Data field index

RowBackColorMapName

Property of VcGroupLevelLayout

This property lets you set or retrieve the name of a color map (type vcColorMap). If set to "", no map will be used. If a map name and additionally a data field index is specified in the property

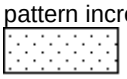
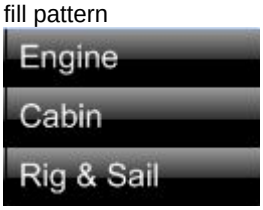
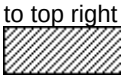
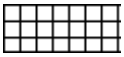
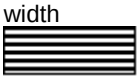
RowBackColorDataFieldIndex, then the background color is controlled by the map. If no data field entry applies, the background color that is specified in the property **RowBackColor** will be used.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the color map Name of the color map

RowPattern

Read Only Property of VcGroupLevelLayout

This property lets you set or retrieve the background pattern of the group title row of this group level.

	Data Type	Explanation
Property value	FillPatternEnum **Possible Values:**	Pattern type
	vc05PercentPattern... vc90PercentPattern 01 - 11	Dots in foreground color on background color, the density of the foreground pattern increasing with the percentage 
	vcAeroGlassPattern 40	Vertical color gradient in the color of the fill pattern 
	vcBDiagonalPattern 5	Diagonal lines slanting from bottom left to top right 
	vcCrossPattern 6	Cross-hatch pattern 
	vcDarkDownwardDiagonalPattern 2014	Diagonal lines slanting from top left to bottom right; spaced 50% closer than vcFDiagonalPattern and of twice the line width 
	vcDarkHorizontalPattern 2023	Horizontal lines spaced 50% closer than vcHorizontalPattern and of twice the line width 

vcDarkUpwardDiagonalPattern 2015	Diagonal lines slanting from bottom left to top right, spaced 50% closer than vcBDiagonalPattern and of twice the line width
vcDarkVerticalPattern 2022	Vertical lines spaced 50% closer than vcVerticalPattern and of of twice the line width
vcDashedDownwardDiagonalPattern 2024	Dashed diagonal lines from top left to bottom right
vcDashedHorizontalPattern 2026	Dashed horizontal lines
vcDashedUpwardDiagonalPattern 2025	Dashed diagonal lines from bottom left to top right
vcDashedVerticalPattern 2027	Dashed vertical lines
vcDiagCrossPattern 7	Diagonal cross-hatch pattern, small
vcDiagonalBrickPattern 2032	Diagonal brick pattern
vcDivotPattern 2036	Divot pattern
vcDottedDiamondPattern 2038	Diagonal cross-hatch pattern of dotted lines
vcDottedGridPattern 2037	Cross-hatch pattern of dotted lines
vcFDiagonalPattern 4	Diagonal lines slanting from top left to bottom right
vcHorizontalBrickPattern 2033	Horizontal brick pattern
vcHorizontalGradientPattern 52	Horizontal color gradient
vcHorizontalPattern 3	Horizontal lines
vcLargeCheckerboardPattern 2044	Checkerboard pattern showing squares of twice the size of vcSmallChecker-BoardPattern
vcLargeConfettiPattern 2029	Confetti pattern, large
vcLightDownwardDiagonalPattern 2012	Diagonal lines slanting to from top left to bottom right; spaced 50% closer than vcBDiagonalPattern

vcLightHorizontalPattern 2019	Horizontal lines spaced 50% closer than vcHorizontalPattern
vcLightUpwardDiagonalPattern 2013	Diagonal lines slanting from bottom left to top right, spaced 50% closer than vcBDiagonalPattern
vcLightVerticalPattern 2018	Vertical lines spaced 50% closer than vcVerticalPattern
vcNarrowHorizontalPattern 2021	Horizontal lines spaced 75 % closer than vcHorizontalPattern
vcNarrowVerticalPattern 2020	Vertical lines spaced 75% closer than vcVerticalPattern
vcNoPattern 1276	No fill pattern
vcOutlinedDiamondPattern 2045	Diagonal cross-hatch pattern, large
vcPlaidPattern 2035	Plaid pattern
vcShinglePattern 2039	Diagonal shingle pattern
vcSmallCheckerBoardPattern 2043	Checkerboard pattern
vcSmallConfettiPattern 2028	Confetti pattern
vcSmallGridPattern 2042	Cross-hatch pattern spaced 50% closer than vcCrossPattern
vcSolidDiamondPattern 2046	Checkerboard pattern showing diagonal squares
vcSpherePattern 2041	Checkerboard of spheres
vcTrellisPattern 2040	Trellis pattern
vcVerticalBottomLightedConvexPattern 43	Vertical color gradient from dark to bright
vcVerticalConcavePattern 40	Vertical color gradient from dark to bright to dark
vcVerticalConvexPattern 41	Vertical color gradient from bright to dark to bright
vcVerticalGradientPattern 62	Vertical color gradient
vcVerticalPattern 2	Vertical lines

vcVerticalTopLightedConvexPattern 42	Vertical color gradient from bright to dark
vcWavePattern 2031	Horizontal wave pattern
vcWeavePattern 2034	Interwoven stripe pattern
vcWideDownwardDiagonalPattern 2016	Diagonal lines slanting from top left to bottom right, showing the same spacing but three times the line width of vcF-DiagonalPattern
vcWideUpwardDiagonalPattern 2017	Diagonal lines slanting from bottom left to top right right, showing the same spacing but three times the line width of vcBDiagonalPattern
vcZigZagPattern 2030	Horizontal zig-zag lines

RowPatternColorAsARGB

Property of VcGroupLevelLayout

This property lets you set or retrieve the pattern color of the group title row of this group level. Color values have a transparency or alpha value, followed by a value for a red, a blue and a green partition (ARGB). The values range between 0..255. An alpha value of 0 equals complete transparency, whereas 255 represents a completely solid color. When casting an RGB value on an ARGB value, an alpha value of 255 has to be added.

Also see **set/getRowBackColorAsARGB**.

If in the property **RowPatternColorMapName** a map is specified, the map will control the pattern color in dependence of the data.

	Data Type	Explanation
Property value	Color	ARGB color values ({0...255},{0...255},{0...255},{0...255})

Example Code

```
Dim groupLevelLayout As VcGroupLevelLayout

Set groupLevelLayout =
VcGantt1.GroupLevelLayoutCollection.GroupLevelLayoutByIndex(0)
groupLevelLayout.RowPatternColorAsARGB = &h88FF0A06
```

RowPatternColorDataFieldIndex

Property of VcGroupLevelLayout

This property lets you set or retrieve the data field index that has to be specified if the property **RowPatternColorMapName** is used. If you set this property to **-1**, no map will be used.

	Data Type	Explanation
Property value	Integer	Data field index
	Possible Values:	Data field index

RowPatternColorMapName

Property of VcGroupLevelLayout

This property lets you set or retrieve the name of a color map (type vcColorMap). If set to "", no map will be used. Only if a map name and a data field index are specified in the property **RowPatternColorDataFieldIndex**, the pattern color is controlled by the map. If no data field entry applies, the pattern color of the group title row that is specified in the property **RowPatternColor** will be used.

	Data Type	Explanation
Property value	String	Name of the color map
	Possible Values:	Name of the color map

RowPatternDataFieldIndex

Property of VcGroupLevelLayout

This property lets you set or retrieve the data field index to be used together with the property **RowPatternMapName**. If you set this property to **-1**, no map will be used.

	Data Type	Explanation
Property value	Integer	Data field index
	Possible Values:	Data field index

RowPatternMapName

Property of VcGroupLevelLayout

This property lets you set or retrieve the name of a pattern map (type vcPatternMap). If set to "", no map will be used. Only if a map name and additionally a data field index are specified in the property **RowPatternDataFieldIndex**, the pattern is controlled by the map. If no data field entry applies, the pattern of the layer that is specified in the property **RowPattern** will be used.

	Data Type	Explanation
Property value	String	Name of the pattern map
	Possible Values:	Name of the color map

SeparationLineColor

Property of VcGroupLevelLayout

This property lets you set or retrieve the color of the separation lines of the the grouping levels.

This property also can be set in the **Grouping** dialog, section **Groupwise**, field **Separation Line**.

	Data Type	Explanation
Property value	Color	Color value ({0...255},{0...255},{0...255})

SeparationLineColorDataFieldIndex

Property of VcGroupLevelLayout

This property lets you set or retrieve the data field index to be used with a map specified by the property **SeparationLineColorMapName**. If you set this property to **-1**, no map will be used.

	Data Type	Explanation
Property value	Long	Data field index

SeparationLineColorMapName

Property of VcGroupLevelLayout

This property lets you set or retrieve the name of a map for the separation line color. If set to "" or if the property **GroupLevelLayoutLineColorDataFieldIndex** is set to -1, then no map will be used.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the color map Name of the color map

SeparationLineThickness

Property of VcGroupLevelLayout

This property lets you set or retrieve the line thickness of a separation line between grouping levels.

If you set this property to values between 1 and 4, an absolute line thickness is defined in pixels. Irrespective of the zoom factor a line will always show the same line thickness in pixels. When printing though, the line thickness is adapted for the sake of legibility and becomes dependent of the zoom factor:

Value	Points	mm
1	1/2 point	0.09 mm
2	1 point	0.18 mm
3	3/2 points	0.26 mm
4	2 points	0.35 mm

A point equals 1/72 inch and represents the unit of the font size.

If you set this property to values between 5 and 1,000, the line thickness is defined in 1/100 mm, so the lines will be displayed in a true thickness in pixels that depends on the zoom factor.

	Data Type	Explanation
Property value	Long	Line thickness LineType {1...4}: line thickness in pixels LineType {5...1000}: line thickness in 1/100 mm

SeparationLineType

Property of VcGroupLevelLayout

This property lets you set or retrieve the line type of a date line.

This property also can be set in the **Grouping** dialog, section **Groupwise**, field **Separation Line**.

	Data Type	Explanation
Property value	LineTypeEnum	Type of separation lines of hierarchy levels
	Possible Values:	
	vcDashed 4	Line dashed
	vcDashedDotted 5	Line dashed-dotted
	vcDotted 3	Line dotted
	vcLineType0 100	Line Type 0
	vcLineType1 101	Line Type 1
	vcLineType10 110	Line Type 10
	vcLineType11 111	Line Type 11
	vcLineType12 112	Line Type 12
	vcLineType13 113	Line Type 13
	vcLineType14 114	Line Type 14
	vcLineType15 115	Line Type 15
	vcLineType16 116	Line Type 16
	vcLineType17 117	Line Type 17
	vcLineType18 118	Line Type 18
	vcLineType2 102	Line Type 2
	vcLineType3 103	Line Type 3

vcLineType4 104	Line Type 4
vcLineType5 105	Line Type 5
vcLineType6 106	Line Type 6
vcLineType7 107	Line Type 7
vcLineType8 108	Line Type 8
vcLineType9 109	Line Type 9
vcNone 1	No line type
vcNotSet -1	No line type assigned
vcSolid 2	Line solid

ShowCalendarGrids

Property of VcGroupLevelLayout

This property lets you set or retrieve whether workfree periods are marked by background color and/or a pattern. This property also can be set in the **Grouping** dialog.

	Data Type	Explanation
Property value	Boolean	Workfree periods are/are not accentuated
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
End Sub
```

ShowDateLineGrids

Property of VcGroupLevelLayout

This property lets you set or retrieve whether a vertical date grid is displayed. This property also can be set in the **Grouping** dialog.

	Data Type	Explanation
Property value	Boolean	Date grids are/are not displayed.
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
End Sub
```

ShowDateLines

Property of VcGroupLevelLayout

This property lets you set or retrieve whether date lines are to be displayed. This property also can be set in the **Grouping** dialog.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	Date lines are/are not displayed. Group invisible/visible group nodes are/are not visible

ShowGroupNodes

Property of VcGroupLevelLayout

This property lets you set or retrieve whether the group nodes of this level are displayed. This property also can be set in the **Grouping** dialog.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	group nodes are/are not visible Group invisible/visible group nodes are/are not visible

ShowSeparationLines

Property of VcGroupLevelLayout

This property lets you set or retrieve whether separation lines are to be displayed between grouping levels.

This property also can be set in the **Groupwise** section of the **Grouping** dialog.

	Data Type	Explanation

ShowSeparationLinesAtTop

Property of VcGroupLevelLayout

This property lets you set or retrieve whether separation lines between groups are to be displayed above the group (or below).

This property also can be set in the **Groupwise** section of the **Grouping** dialog.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	Separation lines at top are displayed/not displayed Group invisible/visible group nodes are/are not visible

SortDataFieldIndex

Property of VcGroupLevelLayout

This property lets you set/retrieve the data field index the groups of this grouping level are sorted by. This property also can be set in the **Grouping** dialog.

	Data Type	Explanation
Parameter: ⇨ sortlevel	Integer **Possible Values:**	Sorting level Data field index
Property value	Long	Index of the data field that holds the sorting criterion

SortOrder

Property of VcGroupLevelLayout

This property lets you specify the sorting order of groups (ascending or descending). The property **SortDataFieldIndex** lets you specify the field the groups are sorted by. This property also can be set in the **Grouping** dialog.

	Data Type	Explanation
Parameter: ⇨ sortLevel	Integer **Possible Values:**	Sorting level Data field index
Property value	SortOrderEnum **Possible Values:** vcAscending 1 vcDescending 2	Direction of the sorting order **Default value:** vcAscending ascending order descending order

Example Code

```
VcGantt1.VcGroupLevelLayout.SortOrderField (0) = 12
VcGantt1.VcGroupLevelLayout (0) = vcAscending
VcGantt1.VcGroupLevelLayout
```

Specification

Read Only Property of VcGroupLevelLayout

This property lets you retrieve the specification of a group level layout. A specification is a string that contains legible ASCII characters from 32 to 127 only, so it can be stored without problems to text files or data bases. This allows for persistency. A specification can be used to create a group level layout by the method **VcGroupLevelLayoutCollection.AddBy-Specification**.

	Data Type	Explanation
Property value	String **Possible Values:**	Specification of group level layout Name of the color map

SummaryBarsVisible

Property of VcGroupLevelLayout

This property lets you set or retrieve whether summary bars are be displayed or not.

This property also can be set in the **Groupwise** section of the **Grouping** dialog.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	Summary bars visible (True)/ invisible (False) Group invisible/visible group nodes are/are not visible

Example Code

```
VcGroupLevelLayout.SummaryBarsVisible (-1) = False
```

Visible

Property of VcGroupLevelLayout

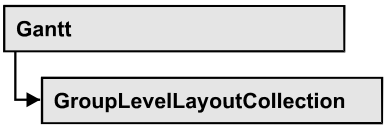
This property lets you set or retrieve whether (True) or not (False) this group level is visible.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	Group level visible/invisible Group invisible/visible group nodes are/are not visible

Example Code

```
Dim groupLevelLayoutCltn As VcGroupLevelLayoutCollection
Dim groupLevelLayout As VcGroupLevelLayout

Set groupLevelLayoutCltn = VcGantt1.GroupLevelLayoutCollection
Set groupLevelLayout = groupLevelLayoutCltn.GroupLevelLayoutByName("A")

groupLevelLayout.Visible = False
```

7.41 VcGroupLevelLayoutCollection



If nodes were grouped, an object of the type VcGroupLevelLayoutCollection contains all available layouts. You can access all objects in an iterative loop by **For Each groupLevelLayout In GroupLevelLayoutCollection** or by the methods **First...** and **Next...**. You can access a single layout using the methods **GroupLevelLayoutByName** and **GroupLevelLayoutIndex**. The number of layouts in the collection object can be retrieved by the property **Count**. The methods **Add**, **Copy** and **Remove** allow to handle the layouts in the corresponding way.

Properties

- _NewEnum
- Count

Methods

- Add
- AddBySpecification
- Copy
- FirstGroupLevelLayout
- GroupLevelLayoutByIndex
- GroupLevelLayoutByName
- NextGroupLevelLayout
- Remove
- Update

Properties

_NewEnum

Read Only Property of VcGroupLevelLayoutCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all map objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method

GetEnumerator is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim groupLevelLayout As VcGroupLevelLayout

For Each groupLevelLayout In VcGantt1.GroupLevelLayout
   Debug.Print groupLevelLayout.Count
Next
```

Count

Read Only Property of VcGroupLevelLayoutCollection

This property lets you retrieve the number of group level layouts in the GroupLevelLayoutCollection object.

	Data Type	Explanation
Property value	Long	Number of group level layouts

Example Code

```
Dim groupLevelLayoutCltn As Vc GroupLevelLayoutCollection
Dim numberOfGroupLevelLayouts As Long
Set groupLevelLayoutCltn = VcGantt1.GroupLevelLayoutCollection
numberOfGroupLevelLayouts = groupLevelLayoutCltn.Count
```

Methods

Add

Method of VcGroupLevelLayoutCollection

This method lets you create a group level layout as a member of the GroupLevelLayoutCollection. If the name was not used before, the new group level layout object will be returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
Parameter:		
⇨ groupLevelLayoutName	String **Possible Values:**	Name of group level layout

		Name of the color map
Return value	VcGroupLevelLayout	New group level layout object

Example Code

```
Set newGroupLevelLayout =
VcGantt1.GroupLevelLayoutCollection.Add("GroupingLevel1")
```

AddBySpecification

Method of VcGroupLevelLayoutCollection

This method lets you create a group level layout by using a group level layout specification. This way of creating allows group level layout objects to become persistent. The specification of a group level layout can be saved and re-loaded (see VcGroupLevelLayout property **Specification**). In a subsequent session the group level layout can be created again from the specification and is identified by its name.

	Data Type	**Explanation**
Parameter:		
⇨ Specification	String **Possible Values:**	Group level layout specification Name of the color map
Return value	VcGroupLevelLayout	New group level layout object

Copy

Method of VcGroupLevelLayoutCollection

By this method you can copy a group level layout. If the group level layout that is to be copied exists, and if the name for the new group level layout does not yet exist, the new group level layout object is returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	**Explanation**
Parameter:		
⇨ groupLevelLayoutName	String **Possible Values:**	Name of the group level layout to be copied Name of the color map
⇨ newGroupLevelLayoutName	String **Possible Values:**	Name of the new group level layout

		Name of the color map
Return value	VcGroupLevelLayout	Group level layout object

Example Code

```
Dim groupLevelLayoutCltn As VcGroupLevelLayoutCollection
Dim groupLevelLayout As VcGroupLevelLayout
Set groupLevelLayoutCltn = VcGantt1.GroupLevelLayoutCollection
Set groupLevelLayout = groupLevelLayoutCltn.Copy("CurrentGroupLevelLayout",
"NewGroupLevelLayout")
```

FirstGroupLevelLayout

Method of VcGroupLevelLayoutCollection

This method can be used to access the initial value, i.e. the first group level layout of a group level layout collection and then to continue in a forward iteration loop by the method **NextGroupLevelLayout** for the group level layouts following. If there is no group level layout in the GroupLevelLayoutCollection, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcGroupLevelLayout	First group level layout

Example Code

```
Dim groupLevelLayoutCltn As VcGroupLevelLayoutCollection
Dim groupLevelLayout As VcGroupLevelLayout
Set groupLevelLayoutCltn = VcGantt1.GroupLevelLayoutCollection
groupLevelLayoutCltn.SelectgroupLevelLayouts (vcAnyGroupLevelLayout)
Set groupLevelLayout = groupLevelLayoutCltn.FirstGroupLevelLayout
```

GroupLevelLayoutByIndex

Method of VcGroupLevelLayoutCollection

This method lets you access a certain group level layout by its index. If a group level layout of the specified index does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter:		
⇨ index	Integer **Possible Values:**	Index of the group level layout Data field index
Return value	VcGroupLevelLayout	Group level layout object returned

Example Code

```
Dim groupLevelLayoutCltn As VcGroupLevelLayout
Dim dateLine As VcDateLine
Set groupLevelLayoutCltn = VcGantt1.GroupLevelLayout
Set groupLevelLayout = groupLevelLayoutCltn.GroupLevelLayoutByIndex(2)
MsgBox groupLevelLayout.Name
```

GroupLevelLayoutByName

Method of VcGroupLevelLayoutCollection

This method is used to access a group level layout by its name. If a group level layout of the specified name does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter:		
⇨ groupLevelLayoutName	String **Possible Values:**	Name of the group level layout Name of the color map
Return value	VcGroupLevelLayout	Group level layout

Example Code

```
Dim groupLevelLayoutCltn As VcGroupLevelLayoutCollection
Dim groupLevelLayout As VcGroupLevelLayout
Set groupLevelLayoutCltn = VcGantt1.GroupLevelLayoutCollection
Set groupLevelLayout = groupLevelLayoutCltn.GroupLevelLayoutByName("Grouping
level A")
```

NextGroupLevelLayout

Method of VcGroupLevelLayoutCollection

This method can be used in a forward iteration loop to retrieve subsequent group level layouts from a GroupLevelLayoutCollection after initializing the loop by the method **FirstGroupLevelLayout**. If there is no group level layout left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcGroupLevelLayout	Subsequent group level layout

Example Code

```
Dim groupLevelLayoutCltn As VcGroupLevelLayoutCollection
Dim groupLevelLayout As VcGroupLevelLayout
Set groupLevelLayoutCltn = VcGantt1.GroupLevelLayoutCollection
Set groupLevelLayout = groupLevelLayoutrCltn.FirstGroupLevelLayout
While Not groupLevelLayout Is Nothing
```

```
Listbox.AddItem groupLevelLayout.Name
Set groupLevelLayout = groupLevelLayoutCltn.NextGroupLevelLayout
Wend
```

Remove

Method of VcGroupLevelLayoutCollection

This method lets you delete a group level layouts. If the group level layout is used in another object, it cannot be deleted. Then False will be returned, otherwise True.

	Data Type	Explanation
Parameter:		
⇨ groupLevelLayoutName	String	Group level layout name
	Possible Values:	Name of the color map
Return value	Boolean	Group level layout deleted (True)/not deleted (False)

Example Code

```
Dim groupLevelLayoutCltn As VcGroupLevelLayoutCollection
Dim groupLevelLayout As VcGroupLevelLayout
Set groupLevelLayoutCltn = VcGantt1.GroupLevelLayoutCollection
Set groupLevelLayout = groupLevelLayoutCltn.FormatByIndex(1)
groupLevelLayoutCltn.Remove (groupLevelLayout.Name)
```

Update

Method of VcGroupLevelLayoutCollection

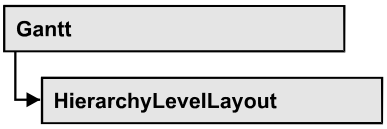
This method has to be used when group level layout modifications have been carried out. The method **Update** updates all objects that are concerned by the group level layout you have edited. You should call this method at the end of the code that defines the group level layouts and the group level layout collection. Otherwise the update will be processed before all group level layout definitions are processed.

	Data Type	Explanation
Return value	Boolean	update successful (True)/ not successful (False)

Example Code

```
Dim groupLevelLayout As VcGroupLevelLayout
Set groupLevelLayout =
VcGantt1.GroupLevelLayout.Collection.GroupLevelLayoutByName("Grouping Level 3")
groupLevelLayout.Update
```

7.42 VcHierarchyLevelLayout



An object of the type **VcHierarchyLevelLayout** defines the content and the appearance of the hierarchical order of nodes.

Properties

- AutoCollapseGroups
- AutoExpandTargetGroup
- BodiesCollapsed
- BodiesCollapsedDataFieldIndex
- BodiesCollapsedMapName
- HierarchyDataFieldIndex
- LevelMaximumForPagebreaks
- NodeSeparationLinesVisible
- NodesInHeaders
- NodesInHeadersDataFieldIndex
- NodesInHeadersMapName
- NodesOverlaid
- PagebreakMode
- RestoreAutoCollapsedGroups
- RestoreAutoExpandedGroups
- SeparationLineColor
- SeparationLineThickness
- SeparationLineType
- ShowSeparationLines
- SummaryBarsVisible

Properties

AutoCollapseGroups

Property of VcHierarchyLevelLayout

This property lets you set or retrieve whether in the hierarchy level layout the groups are to becollapsed automatically on interactions.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	Groups are/are not collapsed automatically on interactions Group invisible/visible group nodes are/are not visible

AutoExpandTargetGroup

Property of VcHierarchyLevelLayout

This property lets you set or retrieve whether in the hierarchy level layout the groups are to be expanded automatically on interactions.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	Target groups are/are not expanded automatically on interactions Group invisible/visible group nodes are/are not visible

BodiesCollapsed

Property of VcHierarchyLevelLayout

This property lets you set or retrieve, whether (True) or not (False) all groups are collapsed.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	Groups are collapsed/are not collapsed Group invisible/visible group nodes are/are not visible

BodiesCollapsedDataFieldIndex

Property of VcHierarchyLevelLayout

This property lets you set or retrieve the data field index for the collapsed bodies of this hierarchy level. This property also can be set in the **Grouping** dialog.

	Data Type	Explanation
Property value	Long	This levels groups bodies collapsed data field index

BodiesCollapsedMapName

Property of VcHierarchyLevelLayout

This property lets you set or retrieve the name of a map for the bodies collapsed on this hierarchy level. If set to "" or if the property **Bodies-CollaspsedDataFieldIndex** is set to -1, then no map will be used.

	Data Type	Explanation
Property value	Long	This levels groups bodies collapsed map name

HierarchyDataFieldIndex

Property of VcHierarchyLevelLayout

This property lets you set/retrieve the data field index used for grouping of this **VcGroupLevelLayout** object

	Data Type	Explanation
Property value	Long	Data field which defines the hierarchical order of activities

LevelMaximumForPagebreaks

Property of VcHierarchyLevelLayout

This property lets you set or retrieve up to which hierarchy level page breaks are to be carried out.

If this property is set to the default -1 the page breaks are carried out on each hierarchy level.

	Data Type	Explanation

NodeSeparationLinesVisible

Read Only Property of VcHierarchyLevelLayout

This property lets you set or retrieve whether or not separation lines are to be displayed.

This property can also be set in the **Node** section of the **Grouping** dialog.

	Data Type	Explanation

Example Code

```
VcHierarchyLevelLayout.NodeSeparationLinesVisible (-1) = False
```

NodesInHeaders

Property of VcHierarchyLevelLayout

This property lets you set or retrieve whether (True) or not (False) the node objects of the group of this level are positioned in the same row.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	All nodes of the group are/are not in the same row Group invisible/visible group nodes are/are not visible

Example Code

```
Dim hierarchyLevelLayoutCltn As VcHierarchyLevelLayoutCollection
Dim hierarchyLevelLayout As VcHierarchyLevelLayout

Se thierarchyLevelLayoutCltn = VcGantt1.HierarchyLevelLayoutCollection
Set hierarchyLevelLayout =
hierarchyLevelLayoutCltn.HierarchyLevelLayoutByName("3")

hierarchyLevelLayout.AllNodesInOneRow = True
```

NodesInHeadersDataFieldIndex

Property of VcHierarchyLevelLayout

This property lets you set or retrieve the data field index for the nodes in headers of this hierarchy level. This property also can be set in the **Grouping** dialog.

	Data Type	Explanation
Property value	Long	

NodesInHeadersMapName

Property of VcHierarchyLevelLayout

This property lets you set or retrieve the map name for the nodes in headers on this grouping level. If set to "" or if the property **NodesInHeadersDataFieldIndex** is set to -1, then no map will be used.

	Data Type	Explanation
Property value	Long	

NodesOverlaid

Property of VcHierarchyLevelLayout

This property lets you set or retrieve whether (False) the node layout on this group level is optimized or if nodes overlap (True).

	Data Type	Explanation
Parameter:		
⇦ Rückgabewert	Boolean **Possible Values:**	The node layout is/is not at its optimum Group invisible/visible group nodes are/are not visible
Property value	Long	

Example Code

```
group.LevelLayout.NodesArrangedOptimized = True
```

PagebreakMode

Property of VcHierarchyLevelLayout

This property lets you set or retrieve whether and when page breaks after groups are to be carried out. This property also can be set in the **Grouping** dialog.

	Data Type	Explanation
Property value	PagebreakModeEnum **Possible Values:** vcPagebreakAfterEachGroup 1 vcPagebreakNone 0 vcPagebreakOnPageFull 2	Page break mode **Default value:** vcPagebreakNone Pagebreak after each group No pagebreak Pagebreak if following group wouldn't fit on page completely

RestoreAutoCollapsedGroups

Property of VcHierarchyLevelLayout

This property lets you set or retrieve whether in the hierarchy level layout automatically collapsed groups are to be restored automatically on interactions.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	Automatically collapsed groups are/are not restored automatically on interactions Group invisible/visible group nodes are/are not visible

RestoreAutoExpandedGroups

Property of VcHierarchyLevelLayout

This property lets you set or retrieve whether in the hierarchy level layout automatically expanded groups are to be restored automatically on interactions.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	Automatically expanded groups are/are not restored automatically on interactions Group invisible/visible group nodes are/are not visible

SeparationLineColor

Property of VcHierarchyLevelLayout

This property lets you set or retrieve the color of the separation lines of the the hierarchy levels.

This property also can be set in the **Hierarchy** section of the **Grouping** by clicking on ··· next to **Separation Line**.

	Data Type	Explanation
Property value	Color	Color value ({0...255},{0...255},{0...255})

Example Code

```
VcHierarchyLevelLayout.SeparationLineColor = RGB(255, 204, 204)
```

SeparationLineThickness

Read Only Property of VcHierarchyLevelLayout

This property lets you set or retrieve the line thickness between hierarchy levels.

If you set this property to values between 1 and 4, an absolute line thickness is defined in pixels. Irrespective of the zoom factor a line will always show the same line thickness in pixels. When printing though, the line thickness is adapted for the sake of legibility and becomes dependent of the zoom factor:

Value	Points	mm
1	1/2 point	0.09 mm
2	1 point	0.18 mm
3	3/2 points	0.26 mm
4	2 points	0.35 mm

A point equals 1/72 inch and represents the unit of the font size.

If you set this property to values between 5 and 1,000, the line thickness is defined in 1/100 mm, so the lines will be displayed in a true thickness in pixels that depends on the zoom factor.

This property also can be set in the **Hierarchy** section of the **Grouping** by clicking on ··· next to **Separation Line**.

	Data Type	Explanation
Property value	Long	Line thickness LineType {1...4}: line thickness in pixels LineType {5...1000}: line thickness in 1/100 mm

SeparationLineType

Read Only Property of VcHierarchyLevelLayout

This property lets you set or retrieve the line type of a date line.

This property also can be set in the **Hierarchy** section of the **Grouping** by clicking on ··· next **Separation Line**.

	Data Type	Explanation
Property value	LineTypeEnum	Type of separation lines of hierarchy levels
	Possible Values:	
	vcDashed 4	Line dashed
	vcDashedDotted 5	Line dashed-dotted
	vcDotted 3	Line dotted
	vcLineType0 100	Line Type 0
	vcLineType1 101	Line Type 1
	vcLineType10 110	Line Type 10
	vcLineType11 111	Line Type 11
	vcLineType12 112	Line Type 12
	vcLineType13 113	Line Type 13
	vcLineType14 114	Line Type 14
	vcLineType15 115	Line Type 15
	vcLineType16 116	Line Type 16
	vcLineType17 117	Line Type 17
	vcLineType18 118	Line Type 18
	vcLineType2 102	Line Type 2
	vcLineType3 103	Line Type 3
	vcLineType4 104	Line Type 4

vcLineType5 105	Line Type 5
vcLineType6 106	Line Type 6
vcLineType7 107	Line Type 7
vcLineType8 108	Line Type 8
vcLineType9 109	Line Type 9
vcNone 1	No line type
vcNotSet -1	No line type assigned
vcSolid 2	Line solid

ShowSeparationLines

Read Only Property of VcHierarchyLevelLayout

This property lets you set or retrieve whether separation lines are to be displayed between hierarchy levels.

This property also can be set in the **Hierarchy** section of the **Grouping** dialog.

	Data Type	Explanation
Property value	Boolean	Separation lines are displayed/not displayed
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
VcHierarchyLevelLayout.ShowSeparationLines = True
```

SummaryBarsVisible

Read Only Property of VcHierarchyLevelLayout

This property lets you set or retrieve whether or not summary bars are to be displayed.

This property also can be set in the **Hierarchy** section of the **Grouping** dialog.

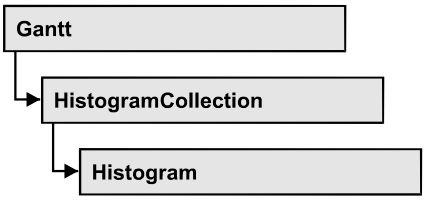
	Data Type	Explanation
Property value	Boolean	summary bars visible (True)/ invisible (False)
	Possible Values:	

	Group invisible/visible group nodes are/are not visible

Example Code

```
VcHierarchyLevelLayout.SummaryBarsVisible (-1) = False
```

7.43 VcHistogram



An object of the type VcHistogram is an element of the object **VcHistogramCollection** and is designed to contain capacity curves referring to the values of the Gantt diagram located above it. You can define a scale and create curves, that can obtain its data from different sources.

Properties

- CalendarName
- CurveCollection
- Name
- NominalScaleMaximum
- NominalScaleMinimum
- NumericScaleCollection
- RowBackColorAsARGB
- RowPattern
- RowPatternColorAsARGB
- ShowCalendarGrids
- Visible

Methods

- FitRangeIntoView
- GetActualScaleValues
- GetActualScaleValuesAsVariant
- GetCurrentYValues
- GetCurrentYValuesAsVariant
- PutInOrderAfter
- ScrollToValue

Properties

CalendarName

Property of VcHistogram

This property lets you assign a calendar to the histogram. The calendar holds the time pattern to be displayed by the grid. The calendar is to be specified by its name.

	Data Type	Explanation
Property value	String **Possible Values:**	Character string that passes the calendar name Name of the color map

CurveCollection

Read Only Property of VcHistogram

This property gives access to the curve collection object, that contains all box formats available.

	Data Type	Explanation
Property value	VcCurveCollection	CurveCollection object

Example Code

```
Dim histogramCltn As VcHistogramCollection
Dim histogram As VcHistogram
Dim curveCltn As VcCurveCollection

Set histogramCltn = VcGantt1.HistogramCollection
Set histogram = histogramCltn.HistogramByName("Histogram_1")
Set curveCltn = histogram.CurveCollection
```

Name

Read Only Property of VcHistogram

This property lets you retrieve the name of a histogram curve.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the histogram

		Name of the color map

Example Code

```
Dim histogramCltn As VcHistogramCollection
Dim histogram As VcHistogram

Set histogramCltn = VcGantt1.HistogramCollection
Set histogram = histogramCltn.Active
MsgBox histogram.Name
```

NominalScaleMaximum

Property of VcHistogram

This property lets you specify the maximum value of the numeric scale of the histogram. If the y values of the histogram curves exceed the maximum value set, the numeric scale will be adapted to the curves´ y values.

	Data Type	Explanation
Property value	Long	Maximum y value

Example Code

```
Dim histogramCltn As VcHistogramCollection
Dim histogram As VcHistogram

Set histogramCltn = VcGantt1.HistogramCollection
Set histogram = histogramCltn.HistogramByName("Histogram_1")

histogram.NominalScaleMaximum (20)
```

NominalScaleMinimum

Property of VcHistogram

This property lets you specify a minimum value of the numeric scale of the histogram.

	Data Type	Explanation
Property value	Long	Minimum y-value

Example Code

```
Dim histogramCltn As VcHistogramCollection
Dim histogram As VcHistogram

Set histogramCltn = VcGantt1.HistogramCollection
Set histogram = histogramCltn.HistogramByName("Histogram_1")

histogram.NominalScaleMinimum (2)
```

NumericScaleCollection

Read Only Property of VcHistogram

This property gives access to the NumericScaleCollection object, that contains all numeric scales available.

	Data Type	Explanation
Property value	VcNumericScaleCollection	NumericScaleCollection object

Example Code

```
Dim histogramCltn As VcHistogramCollection
Dim histogram As VcHistogram
Dim numericScaleCltn As VcNumericScaleCollection

Set histogramCltn = VcGantt1.HistogramCollection
Set histogram = histogramCltn.HistogramByName("Histogram_1")
Set numericScaleCltn = histogram.NumericScaleCollection
```

RowBackColorAsARGB

Property of VcHistogram

This property lets you set or retrieve the background color of the histogram. This property also can be set in the **Administrate Histograms** property page.

	Data Type	Explanation
Property value	System.Drawing.Color	ARGB color values ({0...255},{0...255},{0...255},{0...255})

Example Code

```
VcHistogram.RowBackColor = RGB(255, 0, 0)
```

RowPattern

Property of VcHistogram

This property lets you set or retrieve the background pattern of the histogram.

	Data Type	Explanation
Property value	VcFillPattern	Pattern type

RowPatternColorAsARGB

Property of VcHistogram

This property lets you set or retrieve the pattern color of the histogram Color values have a transparency or alpha value, followed by a value for a red, a blue and a green partition (ARGB). The values range between 0..255. An alpha value of 0 equals complete transparency, whereas 255 represents a completely solid color. When casting an RGB value on an ARGB value, an alpha value of 255 has to be added.

Note:> The ribbon background of the numeric scale has to be transparent for the background to become visible.

	Data Type	Explanation
Property value	**System.Drawing.Color**	**ARGB color values** **({0...255},{0...255},{0...255},{0...255})**

ShowCalendarGrids

Property of VcHistogram

This property lets you set or retrieve whether workfree periods are marked by a background color and/or a pattern. This property also can be set in the Administrate Histograms dialog.

	Data Type	Explanation
Property value	Boolean	Workfree periods are/are not accentuated
	Possible Values:	Group invisible/visible group nodes are/are not visible

Visible

Property of VcHistogram

This property lets you set or retrieve whether the histogram is visible.

	Data Type	Explanation
Property value	Boolean	Histogram visible (True)/ not visible (False)
	Possible Values:	Group invisible/visible

		group nodes are/are not visible

Example Code

```
Dim histogramCltn As VcHistogramCollection
Dim histogram As VcHistogram

Set histogramCltn = VcGantt1.HistogramCollection
Set histogram = histogramCltn.Active
histogram.Visible = True
```

Methods

FitRangeIntoView

Method of VcHistogram

This method lets you match a section of the numeric scale into a window for display. The graduation will change correspondingly. The beginning and the end are set by the **startValue** and **endValue** parameters, respectively. The parameter **gapAsNoOfTimeUnits** is not used. To derive appropriate section limits from existing curves, see **GetCurrentYValues(...)**.

To match histograms in a window please see **VcGantt.FitHistogramsIntoView**

	Data Type	Explanation
Parameter:		
⇨ startValue	Long	Start date of the area to be matched
⇨ endValue	Long	End date of the area to be matched
⇨ gapAsNoOfTimeUnits	Long	Parameter is not used
Return value	Boolean	Area could (True) / could not (False) be matched.

GetActualScaleValues

Method of VcHistogram

This method lets you retrieve the actual minimum and maximum values of the histogram's numeric scale.

	Data Type	Explanation
Parameter:		
⇦ minimumValue	Long	Minimum Y-value of the numeric scale
⇦ maximumValue	Long	Maximum Y-value of the numeric scale
Return value	Boolean	High-low values could (True) / could not (False) be successfully retrieved.

GetActualScaleValuesAsVariant

Method of VcHistogram

This method is identical with the method **ActualScaleValues** except for the parameters. It was necessary to implement this property because some languages (e.g. VBScript) can use parameters by Reference (indicated by ⇦) only if the type of these parameters is VARIANT.

GetCurrentYValues

Method of VcHistogram

This method lets you retrieve the minimum and maximum Y-value of all curves in the histogram. The result can contribute to defining the section of the numeric scale to be displayed (s. **FitRangeIntoView**).

	Data Type	Explanation
Parameter:		
⇦ minValue	Long	Minimum Y-value of all curves
⇦ maxValue	Long	Maximum Y-value of all curves
Return value	Boolean	High-low values could (True) / could not (False) be successfully retrieved.

GetCurrentYValuesAsVariant

Method of VcHistogram

This method is identical with the method **GetCurrentYValues** except for the parameters. It was necessary to implement this property because some languages (e.g. VBScript) can use parameters by Reference (indicated by ⇦) only if the type of these parameters is VARIANT.

PutInOrderAfter

Method of VcHistogram

This method lets you set the histogram behind a histogram specified by name, within the HistogramCollection. If you set the name to "", the histogram will be put in the first position. The order of the histograms determines the order by which they are displayed.

	Data Type	Explanation
Parameter:		
⇨ refName	String	Name of the histogram behind which the current histogram is to be put.
	Possible Values:	Name of the color map
Return value	Void	

Example Code

```
Dim histgrCltn As VcHistogramCollection
Dim histgr1 As VcHistogram
Dim histgr2 As VcHistogram

histgrCltn = VcGantt1.HistogramCollection()
histgr1 = histgrCltn.Add("histgr1")
histgr2 = histgrCltn.Add("histgr2")
histgr1.PutInOrderAfter("histgr2")
histgrCltn.Update()
```

ScrollToValue

Method of VcHistogram

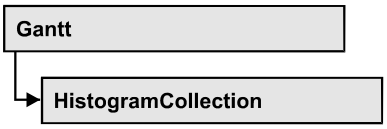
This method allows you to scroll to a defined y value in the histogram and to specify whether that value should be displayed at the top, in the center or at the bottom of the screen.

	Data Type	Explanation
Parameter:		
⇨ value	Long	Y value to be scrolled to
⇨ verAlignment	VerticalAlignmentEnum	Vertical alignment
	Possible Values: vcBottomAligned 2 vcTopAligned 1 vcVerCenterAligned -1	 bottom aligned top aligned vertically centered
Return value	Boolean	Scrolling was (True) / was not (False) performed successfully.

Example Code

```
Dim histogramCltn As VcHistogramCollection
Dim histogram As VcHistogram

Set histogramCltn = VcGantt1.HistogramCollection
Set histogram = histogramCltn.HistogramByName("HISTOGRAM_1")
histogram.ScrollToValue 7, vcCenterAligned
```

7.44 VcHistogramCollection



An object of the type VcHistogramCollection automatically contains all available histograms. You can access all objects in an iterative loop by **For Each histogram In HistogramCollection** or by the methods **First...** and **Next...**. You can access a single histogram using the method **HistogramByName**. The number of groups in the collection object can be retrieved by the property **Count**.

Properties

- _NewEnum
- Active
- Count

Methods

- CreateHistogram
- DeleteHistogram
- FirstHistogram
- HistogramByIndex
- HistogramByName
- NextHistogram

Properties

_NewEnum

Read Only Property of VcHistogramCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all histogram objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method **GetEnumerator** is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim histogram As VcHistogram

For Each histogram In VcGantt1.HistogramCollection
   Debug.Print histogram.Name
Next
```

Active

Property of VcHistogramCollection

This property lets you set or retrieve the histogram currently displayed in the diagram.

A histogram can be **Nothing** in case no user interaction (e. g. marking a curve) has taken place.

	Data Type	Explanation
Property value	VcHistogram	Histogram currently used

Example Code

```
Dim histogramCltn As VcHistogramCollection
Dim histogram As VcHistogram

Set histogramCltn = VcGantt1.HistogramCollection
Set histogram = histogramCltn.Active
```

Count

Read Only Property of VcHistogramCollection

This property lets you retrieve the number of histograms in the HistogramCollection object.

	Data Type	Explanation
Property value	Long	Number of histograms

Example Code

```
Dim histogramCltn As VcHistogramCollection
Dim numberOfHistograms As Long

Set histogramCltn = VcGantt1.HistogramCollection
numberOfHistograms = histogramCltn.Count
```

Methods

CreateHistogram

Method of VcHistogramCollection

By this method you can create a histogram object, which automatically is a member of the HistogramCollection object. The histogram is a copy of the one previously created and therefore contains the same curves.

	Data Type	Explanation
Parameter: ⇨ histogramName	String **Possible Values:**	Name of the histogram to be created Name of the color map
Return value	VcHistogram	Histogram created

Example Code

```
Dim histogramCltn As VcHistogramCollection
Dim histogram As VcHistogram

Set histogram = histogramCltn.CreateHistogram ("Histogram2")
```

DeleteHistogram

Method of VcHistogramCollection

This method lets you delete a histogram from the HistogramCollection object.

	Data Type	Explanation
Parameter: ⇨ histogramName	String **Possible Values:**	Name of the histogram to be deleted Name of the color map
Return value	Boolean	Histogram was (True) / was not (False) deleted successfully.

Example Code

```
Dim histogramCltn As VcHistogramCollection
Dim Deleted As Boolean

Set histogramCltn = VcGantt1.HistogramCollection
Deleted = histogramCltn.DeleteHistogram (String "name")
```

FirstHistogram

Method of VcHistogramCollection

This method can be used to access the initial value, i.e. the first histogram of a histogram collection, and then to continue in a forward iteration loop by the method **NextHistogram** for the histograms following. If there is no histogram in the histogram collection, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcHistogram	First histogram

Example Code

```
Dim HistogramCltn As VcHistogramCollection
Dim Histogram As VcHistogram

Set HistogramCltnn = VcGantt1.HistogramCollection
Set Histogram = HistogramCltn.FirstHistogram
```

HistogramByIndex

Method of VcHistogramCollection

This method lets you access a histogram by its index. If a histogram of the specified index does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter:		
⇨ index	Integer	Index of the histogram
	Possible Values:	Data field index
Return value	VcHistogram	Histogram object returned

HistogramByName

Method of VcHistogramCollection

By this method you can retrieve a histogram by its name. If there is no histogram of this name, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter:		
⇨ histogramName	String	Name of the histogram
	Possible Values:	Name of the color map
Return value	VcHistogram	Histogram

Example Code

```
Dim HistogramCltnAs VcHistogramCollection
Dim histogram As VcHistogram

Set HistogramCltn = VcGantt1.HistogramCollection
Set histogram = HistogramCltn.HistogramByName("Histogram2")
```

NextHistogram

Method of VcHistogramCollection

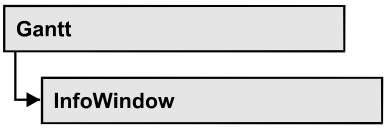
This method can be used in a forward iteration loop to retrieve subsequent histograms from a histogram collection after initializing the loop by the method **FirstHistogram**. If there is no histogram left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcHistogram	Subsequent histogram

Example Code

```
Dim histogramCltn As VcHistogramCollection
Dim histogram As VcHistogram

Set histogramCltn = VcGantt1.HistogramCollection
Set histogram = histogramCltn.FirstHistogram

While Not histogram Is Nothing
    List1.AddItem histogram.Name
    Set histogram = histogramCltn.NextHistogram
Wend
```

7.45 VcInfoWindow



An object of the type VcInfoWindow designates the information window of a node appearing in a Gantt chart when a node is created or modified.

Properties

- OutputFormatForCenterDate
- OutputFormatForDuration
- OutputFormatForEndDate
- OutputFormatForStartDate
- ReferenceDate
- UseReferenceDate
- Visible

Properties

OutputFormatForCenterDate

Property of VcInfoWindow

This property lets you set or retrieve the output format of the a layer's center date (e.g. of a symbol layer) in information windows of nodes. To compose the date you can use the below codes:

D:	first letter of the day of the week (not adjustable)
TD:	Day of the Week (adjustable by using the event **OnSupplyTextEntry**)
DD:	two-digit figure for the day of the month: 01-31
DDD:	first three letters of the day of the week (not adjustable)
M:	first letter of the name of the month (not adjustable)
TM:	name of the month (adjustable by using the event **OnSupplyTextEntry**)
MM:	two-digit figure for the month: 01-12

MMM: first three letters of the name of the month (not adjustable)

YY: two-digit figure for the year

YYYY: four-digit figure for the year

WW: two-digit figure for the number of the calendar week: 01-53

TW: text for "calendar week" (adjustable by using the event **OnSupplyTextEntry**)

Q: one-digit figure for the quarter: 1-4

TQ: name of quarter (adjustable by using the event **OnSupplyTextEntry**)

hh: two-digit figure for the hour in 24 hours format: 00-23

HH: two-digit figure for the hour in 12 hours format: 01-12

Th: Text of "o' clock" (adjustable by using the event **OnSupplyTextEntry**)

TH: "am" or "pm" (adjustable by using the event **OnSupplyTextEntry**)

mm two-digit figure for the minute: 00-59

ss: two-digit figure for the second: 00-59

TS: short date format, as defined in the regional settings of the windows control panel

TL: long date format, as defined in the regional settings of the windows control panel

TT: time format, as defined in the regional settings of the windows control panel

Note: Characters which are not to be interpreted as part of the date should be preceded by a backslash '\'. '\\' for instance results in "\'. The special characters: '**:, /, -**' and **blank** don't need '\' as prefix.

This setting is valid for the table area and for layer annotations in the node area. This property also can be set on the **General** property page.

	Data Type	Explanation
Property value	String **Possible Values:**	String that holds the code of the format to be used; if an empty string is passed, the output format of the Gantt object will be used (see VcGantt.DateOutputFormat).

Name of the color map

OutputFormatForDuration

Property of VcInfoWindow

This property lets you set or retrieve the output format of the duration in information windows of nodes. To compose the date you can use the below codes:

<This property lets you set or retrieve the output format of the duration in information windows of nodes. To compose the date you can use the below codes:

hh: two-digit figure for the hour in 24 hours format: 00-23

mm two-digit figure for the minute: 00-59

ss: two-digit figure for the second: 00-59

xC/XC: *The usage of this format requires a special setting in the .ini file. Please contact NETRONIC if necessary.* You can set a maximum ten-place, simple upward counting, for example "07:16:00", which equals 7 hours, 16 minutes, 0 seconds. The notation is: **xC22:C11:C00**. In written language: Show at least 2 digits for the counters 2...0. The separators are variable and can be replaced by other separators symbols. "x" means: Display a preceding "-" symbol if the value is negative, but no "+" symbol if it is positive. "X" means: Display a preceding "-" symbol if the value is negative and a "+" symbol for positive values.

Note: Characters which are not to be interpreted as part of the date should be preceded by a backslash '\'. '\\' for instance results in "\'. The special characters: '**:, /, -**' and **blank** don't need '\' as prefix.

This setting is valid for the table area and for layer annotations in the node area. This property also can be set on the **General** property page.

Note: Characters which are not to be interpreted as part of the date should be preceded by a backslash '\'. '\\' for instance results in "\'. The special characters: '**:, /, -**' and **blank** don't need '\' as prefix.

This setting is valid for the table area and for layer annotations in the node area. This property also can be set on the **General** property page.

	Data Type	Explanation
Property value	String **Possible Values:**	String that holds the code of the format to be used; if an empty string is passed, the output format of the Gantt object will be used (see VcGantt.DateOutputFormat). Name of the color map

OutputFormatForEndDate

Property of VcInfoWindow

This property lets you set or retrieve the output format of a layer's end date of in information windows of nodes. To compose the date you can use the below codes:

D: first letter of the day of the week (not adjustable)

TD: Day of the Week (adjustable by using the event **OnSupplyTextEntry**)

DD: two-digit figure for the day of the month: 01-31

DDD: first three letters of the day of the week (not adjustable)

M: first letter of the name of the month (not adjustable)

TM: name of the month (adjustable by using the event **OnSupplyTextEntry**)

MM: two-digit figure for the month: 01-12

MMM: first three letters of the name of the month (not adjustable)

YY: two-digit figure for the year

YYYY: four-digit figure for the year

WW: two-digit figure for the number of the calendar week: 01-53

TW: text for "calendar week" (adjustable by using the event **OnSupplyTextEntry**)

Q: one-digit figure for the quarter: 1-4

TQ: name of quarter (adjustable by using the event **OnSupplyTextEntry**)

hh: two-digit figure for the hour in 24 hours format: 00-23

HH: two-digit figure for the hour in 12 hours format: 01-12

Th: Text of "o' clock" (adjustable by using the event **OnSupplyTextEntry**)

TH: "am" or "pm" (adjustable by using the event **OnSupplyTextEntry**)

mm two-digit figure for the minute: 00-59

ss: two-digit figure for the second: 00-59

TS: short date format, as defined in the regional settings of the windows control panel

TL: long date format, as defined in the regional settings of the windows control panel

TT: time format, as defined in the regional settings of the windows control panel

Note: Characters which are not to be interpreted as part of the date should be preceded by a backslash '\'. '\\' for instance results in "\'. The special characters: '**:, /, -**' and **blank** don't need '\' as prefix.

This setting is valid for the table area and for layer annotations in the node area. This property also can be set on the **General** property page.

	Data Type	Explanation
Property value	String **Possible Values:**	String that holds the code of the format to be used; if an empty string is passed, the output format of the Gantt object will be used (see VcGantt.DateOutputFormat). Name of the color map

OutputFormatForStartDate

Property of VcInfoWindow

This property lets you set or retrieve the output format of a layer's start date in information windows of nodes. To compose the date you can use the below codes:

D: first letter of the day of the week (not adjustable)

TD:	Day of the Week (adjustable by using the event **OnSupplyTextEntry**)
DD:	two-digit figure for the day of the month: 01-31
DDD:	first three letters of the day of the week (not adjustable)
M:	first letter of the name of the month (not adjustable)
TM:	name of the month (adjustable by using the event **OnSupplyTextEntry**)
MM:	two-digit figure for the month: 01-12
MMM:	first three letters of the name of the month (not adjustable)
YY:	two-digit figure for the year
YYYY:	four-digit figure for the year
WW:	two-digit figure for the number of the calendar week: 01-53
TW:	text for "calendar week" (adjustable by using the event **OnSupplyTextEntry**)
Q:	one-digit figure for the quarter: 1-4
TQ:	name of quarter (adjustable by using the event **OnSupplyTextEntry**)
hh:	two-digit figure for the hour in 24 hours format: 00-23
HH:	two-digit figure for the hour in 12 hours format: 01-12
Th:	Text of "o' clock" (adjustable by using the event **OnSupplyTextEntry**)
TH:	"am" or "pm" (adjustable by using the event **OnSupplyTextEntry**)
mm	two-digit figure for the minute: 00-59
ss:	two-digit figure for the second: 00-59
TS:	short date format, as defined in the regional settings of the windows control panel
TL:	long date format, as defined in the regional settings of the windows control panel
TT:	time format, as defined in the regional settings of the windows control panel

Note: Characters which are not to be interpreted as part of the date should be preceded by a backslash '\'. '\\' for instance results in "\'. The special characters: '**:, /, -**' and **blank** don't need '\' as prefix.

This setting is valid for the table area and for layer annotations in the node area. This property also can be set on the **General** property page.

	Data Type	Explanation
Property value	String	String that holds the code of the format to be used; if an empty string is passed, the output format of the Gantt object will be used (see VcGantt.DateOutputFormat).
	Possible Values:	Name of the color map

ReferenceDate

Property of VcInfoWindow

This property lets you set or retrieve a reference date. For the information window to actually use the reference date, the property **UseReferenceDate** needs to be set.

	Data Type	Explanation
Property value	Date	Reference date

UseReferenceDate

Property of VcInfoWindow

This property lets you set or retrieve whether the information window uses a reference date.

	Data Type	Explanation
Property value	Boolean	Information Window uses (True) / does not use (False) reference date **Default value:** False
	Possible Values:	Group invisible/visible group nodes are/are not visible

Visible

Property of VcInfoWindow

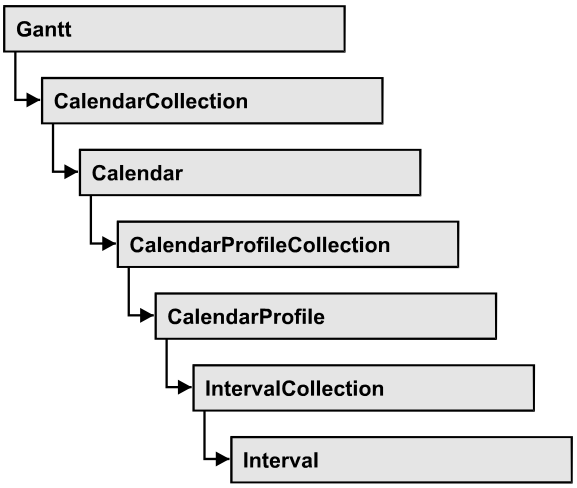
This property lets you set or retrieve whether the information window shoul be visible during node interaction.

	Data Type	Explanation
Property value	Boolean	Information window visible/invisible **Default value:** True
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

Set boxCltn = VcGantt1.BoxCollection
Set box = boxCltn.FirstBox
box.Visible = False
```

7.46 VcInterval



An object of the type **VcInterval** offers the possibility of defining time intervals that are interpreted as working or non-working time. The distinction between the two characteristics is made by the special settings **<WORK>** and **<NONWORK>** of the property **CalendarProfileName**. An interval may refer to other already defined calendar profiles by its property **CalendarProfileName**.

According to the current interval type (**vcCalendarInterval**, **vcDayProfileInterval**, **vcWeekProfileInterval**, **vcYearProfileInterval** oder **vcShiftProfileInterval**) which is not set explicitely but derives from the context of use, only certain properties of the object take effect.

The following table lists the interval types and their corresponding properties:

vcCalendar-Interval	vcYearProfile-Interval	vcWeekProfile-Interval	vcDayProfile-Interval	vcShift-Interval
StartDateTime	StartMonth	StartWeekday	StartTime	Duration
EndDateTime	EndMonth	EndWeekday	EndTime	TimeUnit
	DayInEndMonth			
	DayInStartMonth			

A **CalendarInterval** designates a non-recurring time span within a precisely defined period. Example: 5/5/2010 11:30 to 9/15/2010 5:00.

A **YearProfileInterval** allows to define a yearly recurring day or time span. Example: 5/1 or 12/24 to 12/26.

A **WeekProfileInterval** applies to single or several days in succession of a week. Example: Saturday or Monday to Friday.

A **DayProfileInterval** specifiies certain time spans during a day. Example: 8:00 to 5.00

A **ShiftProfile** designates a time span within the specified unit **vcDay, vcHours, vcMinute** or v**cSeconds** without refererring to a date. Example: 4 hours.

Properties

- BackColorAsARGB
- CalendarProfileName
- DayInEndMonth
- DayInStartMonth
- Duration
- EndDateTime
- EndMonth
- EndTime
- EndWeekday
- LineColor
- LineThickness
- LineType
- Name
- Pattern
- PatternColorAsARGB
- Specification
- StartDateTime
- StartMonth
- StartTime
- StartWeekday
- Text
- TimeUnit
- Type
- UseGraphicalAttributes

Methods

- PutInOrderAfter

Properties

BackColorAsARGB

Property of VcInterval

This property lets you set or retrieve the background color of the interval'x calendar grid. Color values have a transparency or alpha value, followed by a value for a red, a blue and a green partition (ARGB). The values range between 0..255. An alpha value of 0 equals complete transparency, whereas 255 represents a completely solid color. When casting an RGB value on an ARGB value, an alpha value of 255 has to be added.

The background color can also be set in the **Administrate Intervals> dialog.**

	Data Type	Explanation
Property value	**Color**	**ARGB color values** **({0...255},{0...255},{0...255},{0...255})** **Default value: &hFFD8D8D8 (gray)**

CalendarProfileName

Property of VcInterval

This property lets you assign a calendar profile to the interval or retrieve the one currently used. This feature can also be set in the Administrate Intervals dialog.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the calendar profile Name of the color map

DayInEndMonth

Property of VcInterval

This property returns or sets the day in the end month of this interval object (for profiles of the type **vcYearProfile** only). This feature can also be set in the **Administrate Intervals** dialog.

	Data Type	Explanation
Property value	Integer **Possible Values:**	Day of last month Data field index

DayInStartMonth

Property of VcInterval

This property returns or sets the day in the start month of this interval (for profiles of the type **vcYearProfile** only). This feature can also be set in the **Administrate Intervals** dialog.

	Data Type	Explanation
Property value	Integer **Possible Values:**	Day of first month Data field index

Duration

Property of VcInterval

This property lets you set or retrieve the duration for the interval *only for calendar profiles of the type* ***vcShiftProfile***. The duration can also be set in the **Administrate Intervals** dialog.

	Data Type	Explanation
Property value	Long	Duration of interval

EndDateTime

Property of VcInterval

This property returns or sets the end date and time of this interval object (for profiles of the type **vccalendar** only). This feature can also be set in the **Administrate Intervals** dialog.

	Data Type	Explanation
Property value	Date	End date and time of interval

EndMonth

Property of VcInterval

This property returns or sets the end month of this interval object (for profiles of the type **vcYearProfile** only). This feature can also be set in the **Administrate Intervals** dialog.

	Data Type	Explanation
Property value	MonthEnum **Possible Values:** vcApril 4 vcAugust 8 vcDecember 12 vcFebruary 2 vcJanuary 1 vcJuly 7 vcJune 6 vcMarch 3 vcMay 5 vcNovember 11 vcOktober 10 vcSeptember 9	End month of interval **April** **August** **December** **February** **January** **July** **June** **March** **May** **November** **October** **September**

EndTime

Property of VcInterval

This property returns or sets the end time of this interval object (for profiles of the type **vcDayProfile** only). This feature can also be set in the **Administrate Intervals** dialog.

	Data Type	Explanation
Property value	Date	End time of interval

EndWeekday

Property of VcInterval

This property returns or sets the last weekday of this interval object (for profiles of the type **vcWeekProfile** only). This feature can also be set in the **Administrate Intervals** dialog.

	Data Type	Explanation
Property value	WeekdayEnum	Last weekday of interval
	Possible Values:	
	vcFriday 5	Week day **Friday**
	vcMonday 1	Week day **Monday**
	vcSaturday 6	Week day **Saturday**
	vcSunday 7	Week day **Sunday**
	vcThursday 4	Week day **Thursday**
	vcTuesday 2	Week day **Tuesday**
	vcWednesday 3	Week day **Wednesday**

LineColor

Property of VcInterval

This property lets you set or retrieve the line color of an interval's calendar grid lines. The line color can also be set in the **Administrate Intervals** dialog. This feature can also be set in the **Administrate Intervals** dialog.

	Data Type	Explanation
Property value	Color	RGB color values ({0...255},{0...255},{0...255})

LineThickness

Read Only Property of VcInterval

This property lets you set or retrieve the line thickness of the interval's calendar grid lines.

If you set this property to values between 1 and 4, an absolute line thickness is defined in pixels. Irrespective of the zoom factor a line will always show the same line thickness in pixels. When printing though, the line thickness is adapted for the sake of legibility and becomes dependent of the zoom factor:

Value	Points	mm
1	1/2 point	0.09 mm
2	1 point	0.18 mm
3	3/2 points	0.26 mm
4	2 points	0.35 mm

A point equals 1/72 inch and represents the unit of the font size.

If you set this property to values between 5 and 1,000, the line thickness is defined in 1/100 mm, so the lines will be displayed in a true thickness in pixels that depends on the zoom factor.

This property also can be set in the **Administrate intervals** dialog.

	Data Type	Explanation
Property value	Integer	Line thickness LineType {1...4}: line thickness in pixels LineType {5...1000}: line thickness in 1/100 mm **Default value:** As defined in the dialog
	Possible Values:	Data field index

LineType

Property of VcInterval

This property lets you set or retrieve the line type of the interval's calendar grid. The line type property also can be set in the **Administrate Intervals** dialog.

	Data Type	Explanation
Property value	LineTypeEnum	Line type **Default value:** vcSolid
	Possible Values:	
	vcDashed 4	Line dashed
	vcDashedDotted 5	Line dashed-dotted
	vcDotted 3	Line dotted
	vcLineType0 100	Line Type 0
	vcLineType1 101	Line Type 1

vcLineType10 110	Line Type 10
vcLineType11 111	Line Type 11
vcLineType12 112	Line Type 12
vcLineType13 113	Line Type 13
vcLineType14 114	Line Type 14
vcLineType15 115	Line Type 15
vcLineType16 116	Line Type 16
vcLineType17 117	Line Type 17
vcLineType18 118	Line Type 18
vcLineType2 102	Line Type 2
vcLineType3 103	Line Type 3
vcLineType4 104	Line Type 4
vcLineType5 105	Line Type 5
vcLineType6 106	Line Type 6
vcLineType7 107	Line Type 7
vcLineType8 108	Line Type 8
vcLineType9 109	Line Type 9
vcNone 1	No line type
vcNotSet -1	No line type assigned
vcSolid 2	Line solid

Name

Read Only Property of VcInterval

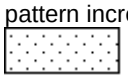
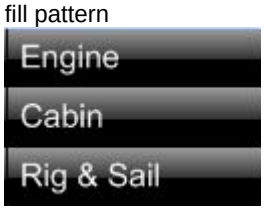
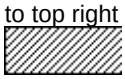
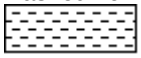
This property lets you retrieve the name of the interval. This feature can also be set in the **Administrate Intervals** dialog.

	Data Type	Explanation
Property value	String	Name of the interval
	Possible Values:	Name of the color map

Pattern

Read Only Property of VcInterval

This property lets you set or retrieve the pattern of the interval's calendar grid. The pattern can also be set in the **Administrate Intervals** dialog.

	Data Type	Explanation
Property value	FillPatternEnum	Pattern type
	Possible Values: vc05PercentPattern... vc90PercentPattern 01 - 11	Dots in foreground color on background color, the density of the foreground pattern increasing with the percentage 
	vcAeroGlassPattern 40	Vertical color gradient in the color of the fill pattern 
	vcBDiagonalPattern 5	Diagonal lines slanting from bottom left to top right 
	vcCrossPattern 6	Cross-hatch pattern 
	vcDarkDownwardDiagonalPattern 2014	Diagonal lines slanting from top left to bottom right; spaced 50% closer than vcFDiagonalPattern and of twice the line width 
	vcDarkHorizontalPattern 2023	Horizontal lines spaced 50% closer than vcHorizontalPattern and of twice the line width 
	vcDarkUpwardDiagonalPattern 2015	Diagonal lines slanting from bottom left to top right, spaced 50% closer than vcBDiagonalPattern and of twice the line width 
	vcDarkVerticalPattern 2022	Vertical lines spaced 50% closer than vcVerticalPattern and of of twice the line width 
	vcDashedDownwardDiagonalPattern 2024	Dashed diagonal lines from top left to bottom right 
	vcDashedHorizontalPattern 2026	Dashed horizontal lines 
	vcDashedUpwardDiagonalPattern 2025	Dashed diagonal lines from bottom left to top right 

vcDashedVerticalPattern 2027	Dashed vertical lines
vcDiagCrossPattern 7	Diagonal cross-hatch pattern, small
vcDiagonalBrickPattern 2032	Diagonal brick pattern
vcDivotPattern 2036	Divot pattern
vcDottedDiamondPattern 2038	Diagonal cross-hatch pattern of dotted lines
vcDottedGridPattern 2037	Cross-hatch pattern of dotted lines
vcFDiagonalPattern 4	Diagonal lines slanting from top left to bottom right
vcHorizontalBrickPattern 2033	Horizontal brick pattern
vcHorizontalGradientPattern 52	Horizontal color gradient
vcHorizontalPattern 3	Horizontal lines
vcLargeCheckerboardPattern 2044	Checkerboard pattern showing squares of twice the size of vcSmallChecker-BoardPattern
vcLargeConfettiPattern 2029	Confetti pattern, large
vcLightDownwardDiagonalPattern 2012	Diagonal lines slanting to from top left to bottom right; spaced 50% closer than vcBDiagonalPattern
vcLightHorizontalPattern 2019	Horizontal lines spaced 50% closer than vcHorizontalPattern
vcLightUpwardDiagonalPattern 2013	Diagonal lines slanting from bottom left to top right, spaced 50% closer than vcBDiagonalPattern
vcLightVerticalPattern 2018	Vertical lines spaced 50% closer than vcVerticalPattern
vcNarrowHorizontalPattern 2021	Horizontal lines spaced 75 % closer than vcHorizontalPattern
vcNarrowVerticalPattern 2020	Vertical lines spaced 75% closer than vcVerticalPattern
vcNoPattern 1276	No fill pattern
vcOutlinedDiamondPattern 2045	Diagonal cross-hatch pattern, large

vcPlaidPattern 2035	Plaid pattern
vcShinglePattern 2039	Diagonal shingle pattern
vcSmallCheckerBoardPattern 2043	Checkerboard pattern
vcSmallConfettiPattern 2028	Confetti pattern
vcSmallGridPattern 2042	Cross-hatch pattern spaced 50% closer than vcCrossPattern
vcSolidDiamondPattern 2046	Checkerboard pattern showing diagonal squares
vcSpherePattern 2041	Checkerboard of spheres
vcTrellisPattern 2040	Trellis pattern
vcVerticalBottomLightedConvexPattern 43	Vertical color gradient from dark to bright
vcVerticalConcavePattern 40	Vertical color gradient from dark to bright to dark
vcVerticalConvexPattern 41	Vertical color gradient from bright to dark to bright
vcVerticalGradientPattern 62	Vertical color gradient
vcVerticalPattern 2	Vertical lines
vcVerticalTopLightedConvexPattern 42	Vertical color gradient from bright to dark
vcWavePattern 2031	Horizontal wave pattern
vcWeavePattern 2034	Interwoven stripe pattern
vcWideDownwardDiagonalPattern 2016	Diagonal lines slanting from top left to bottom right, showing the same spacing but three times the line width of vcF-DiagonalPattern
vcWideUpwardDiagonalPattern 2017	Diagonal lines slanting from bottom left to top right right, showing the same spacing but three times the line width of vcBDiagonalPattern

vcZigZagPattern 2030	Horizontal zig-zag lines

PatternColorAsARGB

Property of VcInterval

This property lets you set or retrieve the pattern color of the interval's calendar grid. Color values have a transparency or alpha value, followed by a value for a red, a blue and a green partition (ARGB). The values range between 0..255. An alpha value of 0 equals complete transparency, whereas 255 represents a completely solid color. When casting an RGB value on an ARGB value, an alpha value of 255 has to be added.

The pattern color can also be set in the **Administrate Intervals** dialog.

	Data Type	Explanation
Property value	Color	RGB color values ({0...255},{0...255},0...255},{0...255})

Specification

Read Only Property of VcInterval

This property lets you retrieve the specification of an interval. A specification is a string that contains legible ASCII characters from 32 to 127 only, so it can be stored smoothly to text files or data bases. This allows for persistency. A specification can be used to create an interval by the method **VcInterval-Collection.AddBySpecification**.

	Data Type	Explanation
Property value	String **Possible Values:**	Specification of the interval Name of the color map

StartDateTime

Property of VcInterval

This property returns or sets the start date and time of this interval object (for profiles of the type **vcCalendar** only). This feature can also be set in the **Administrate Intervals** dialog.

	Data Type	Explanation
Property value	Date	Start date and time of interval

StartMonth

Property of VcInterval

This property returns or sets the start month of this interval object (for profiles of the type **vcYearProfile** only). This feature can also be set in the **Administrate Intervals** dialog.

	Data Type	Explanation
Property value	MonthEnum **Possible Values:** vcApril 4 vcAugust 8 vcDecember 12 vcFebruary 2 vcJanuary 1 vcJuly 7 vcJune 6 vcMarch 3 vcMay 5 vcNovember 11 vcOktober 10 vcSeptember 9	Start month of interval **April** **August** **December** **February** **January** **July** **June** **March** **May** **November** **October** **September**

StartTime

Property of VcInterval

This property returns or sets the start time of this interval object (for profiles of the type **vcDayProfile** only). This feature can also be set in the **Administrate Intervals** dialog.

	Data Type	Explanation
Property value	Date	Start time of interval

StartWeekday

Property of VcInterval

This property returns or sets the first weekday of this interval object (for profiles of the type **vcWeekProfile** only). This feature can also be set in the **Administrate Intervals** dialog.

	Data Type	Explanation
Property value	WeekdayEnum **Possible Values:** vcFriday 5 vcMonday 1 vcSaturday 6 vcSunday 7 vcThursday 4 vcTuesday 2 vcWednesday 3	Start weekday of interval Week day **Friday** Week day **Monday** Week day **Saturday** Week day **Sunday** Week day **Thursday** Week day **Tuesday** Week day **Wednesday**

Text

Property of VcInterval

This property lets you set or retrieve the text of the time ribbon for this interval *only for calendar profiles of the type* ***vcShiftProfile*** The text can also be set in the **Administrate Intervals** dialog.

	Data Type	Explanation
Property value	String **Possible Values:**	Annotation text of the time ribbon Name of the color map

TimeUnit

Property of VcInterval

This property lets you set or retrieve the time unit for the interval *only for calendar profiles of the type* ***vcVariableProfile***. The text can also be set in the **Administrate Intervals** dialog.

	Data Type	Explanation
Property value	TimeUnitEnum **Possible Values:**	Time unit **Default value:** vcDay

vcDay 5	Time unit **day**
vcHour 6	Time unit **hour**
vcMinute 7	Time unit **minute**
vcSecond 8	Time unit **second**

Type

Property of VcInterval

This property lets you enquire the type of the interval. This feature can also be set in the **Administrate Intervals** dialog.

	Data Type	Explanation
Property value	IntervalTypeEnum **Possible Values:** vcCalendarInterval 139 vcDayProfileInterval 4 vcVariableProfileInterval 5 vcWeekProfileInterval 3 vcYearProfileInterval 2	Type of the interval

UseGraphicalAttributes

Property of VcInterval

This property lets you set or retrieve whether the graphical attributes that have been set for this interval shall be used. This feature can be also set in the dialog **Administrate Intervals** (which you reach by clicking ... in the **Administrate Calendar Profiles** dialog). If they are to be used, the property **VcCalendarGrid.UseGraphicalAttributesOfIntervals** needs to have been set to **True**l.

	Data Type	Explanation
Parameter: Rückgabewert	Boolean **Possible Values:**	Graphical attributes of the interval are displayed (True)/are not displayed (False) Group invisible/visible group nodes are/are not visible
Property value	Boolean **Possible Values:**	Graphical attributes are used (True)/are not used (False) **Default value:** True Group invisible/visible

		group nodes are/are not visible

Methods

PutInOrderAfter

Method of VcInterval

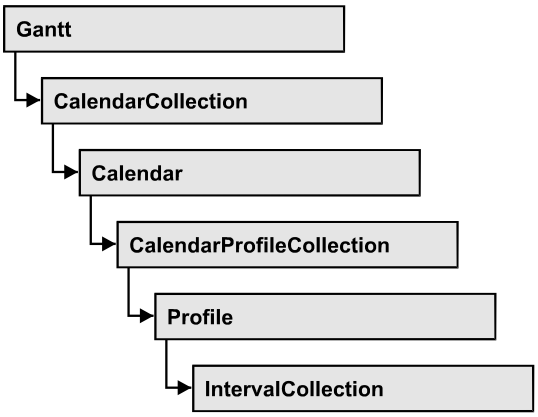
This method lets you set the interval behind an interval specified by name, within the IntervalCollection. If you set the name to "", the interval will be put in the first position. The order of the intervals within the collection determines the order by which they apply to the calendars.

	Data Type	Explanation
Parameter:		
refNameParam	String	Name of the interval behind which the current interval is to be put.
	Possible Values:	Name of the color map
Return value	Void	

Example Code

```
Dim intvlCltn As VcIntervalCollection
Dim intvl1 As VcInterval
Dim intvl2 As VcInterval

intvlCltn = VcGantt1.IntervalCollection()
intvl1 = intvlCltn.Add("intvl1")
intvl2 = intvlCltn.Add("intvl2")
intvl1.PutInOrderAfter("intvl2")
intvlCltn.Update()
```

7.47 VcIntervalCollection



The VcIntervalCollection object contains all intervals available. You can access all objects in an iterative loop by **For Each Interval In BoxFormatCollection** or by the methods **First...** and **Next...**. You can access a single interval by the methods **IntervalByName** and **ntervalByIndex**. The number of intervals in the collection object can be retrieved by the property **Count**. The methods **Add**, **Copy** and **Remove** allow to handle the intervals in the corresponding way.

Properties

- _NewEnum
- Count

Methods

- Add
- AddBySpecification
- Copy
- FirstInterval
- IntervalByIndex
- IntervalByName
- NextInterval
- Remove
- Update

Properties

_NewEnum

Property of VcIntervalCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all interval objects contained. In Visual Basic this property never is displayed, but it can be addressed by the command **For Each** *element* **In** *collection*. In .NET languages the method GetEnumerator is offered instead. Some development environments replace this property by own language constructs.

	Data Type	Explanation
Property value	Object	Reference object

Count

Read Only Property of VcIntervalCollection

This property lets you retrieve the number of intervals in the interval collection.

	Data Type	Explanation
Property value	Long	Number of Interval objects

Methods

Add

Method of VcIntervalCollection

By this method you can create an interval as a member of the IntervalCollection. If the name has not been used before, the new interval object will be returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
Parameter:		
⇨ intervalName	String **Possible Values:**	Interval name Name of the color map
Return value	VcInterval	New interval object

AddBySpecification

Method of VcIntervalCollection

This method lets you create an interval by using an interval specification. This way of creating allows interval objects to become persistent. The specification of an interval can be saved and re-loaded (see VcInterval property **Specification**). In a subsequent the interval can be created again from the specification and is identified by its name.

	Data Type	Explanation
Parameter:		
⇨ Specification	String **Possible Values:**	Interval specification Name of the color map
Return value	VcInterval	New Interval object

Copy

Method of VcIntervalCollection

By this method you can copy an interval. If the interval that is to be copied exists, and if the name for the new interval does not yet exist, the new interval object is returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
Parameter:		
⇨ intervalName	String **Possible Values:**	Name of the interval to be copied Name of the color map
⇨ newIntervalName	String **Possible Values:**	Name of the new interval

		Name of the color map
Return value	VcInterval	Interval object

FirstInterval

Method of VcIntervalCollection

This method can be used to access the initial value, i.e. the first interval of an interval collection, and then to continue in a forward iteration loop by the method **NextInterval** for the intervals following. If there is no interval in the FilterCollection object, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	**Explanation**
Return value	VcInterval	First interval object

IntervalByIndex

Method of VcIntervalCollection

This method lets you access an interval by its index. If no interval of the specified index does exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	**Explanation**
Parameter:		
⇨ Index	Integer **Possible Values:**	Index of the interval Data field index
Return value	VcInterval	Interval object returned

IntervalByName

Method of VcIntervalCollection

By this method you can retrieve an interval by its name. If no interval of the specified name does exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter: ⇨ intervalName	String **Possible Values:**	Name of the interval object Name of the color map
Return value	VcInterval	interval object returned

NextInterval

Method of VcIntervalCollection

This method can be used in a forward iteration loop to retrieve subsequent intervals from an interval collection after initializing the loop by the method **FirstInterval**. If there is no interval left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcInterval	Subsequent interval object

Remove

Method of VcIntervalCollection

This method lets you delete an interval. If the interval is used in another object, it cannot be deleted. Then False will be returned, otherwise True.

	Data Type	Explanation
Parameter: ⇨ intervalName	String **Possible Values:**	Interval name Name of the color map
Return value	Boolean	interval deleted (True)/not deleted (False)

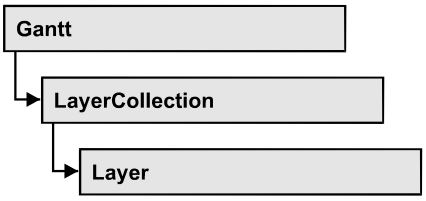
Update

Method of VcIntervalCollection

This method lets you update an interval collection after having modified it.

	Data Type	Explanation
Return value	Boolean	update successful (True)/ not successful (False)

7.48 VcLayer



A layer is the graphical representation of a date (symbol layer) or a set of two dates (rectangle layer) within a node. A layer can be customized by a lot of attributes (shape, color, height, offset, contents of annotation fields, font).

Properties

- BackColorAsARGB
- BackColorDataFieldIndex
- BackColorMapName
- CompletionDataFieldIndex
- DurationDataFieldIndex
- EndDataFieldIndex
- EndSnapTarget
- FilterName
- GraphicsFileName
- GraphicsFileNameDataFieldIndex
- GraphicsFileNameMapName
- Height
- HeightDataFieldIndex
- HeightMapName
- HorizontalOffset
- LabelSizeDependence
- LayerFormat
- LayerShape
- LegendText
- LineColor
- LineColorDataFieldIndex
- LineColorMapName
- LineThickness
- LineType
- MaximumEndDataFieldIndex
- MinimumStartDataFieldIndex
- Moveable

- Name
- NonWorkInterval
- NonWorkIntervalBackColorAsARGB
- NonWorkIntervalBackColorDataFieldIndex
- NonWorkIntervalBackColorMapName
- NonWorkIntervalLineColor
- NonWorkIntervalLineColorDataFieldIndex
- NonWorkIntervalLineColorMapName
- NonWorkIntervalLineThickness
- NonWorkIntervalLineType
- NonWorkIntervalPattern
- NonWorkIntervalPatternColorAsARGB
- NonWorkIntervalPatternColorDataFieldIndex
- NonWorkIntervalPatternColorDataFieldIndex
- NonWorkIntervalPatternDataFieldIndex
- NonWorkIntervalPatternMapName
- NonWorkIntervalShape
- ObjectDrawEventsEnabled
- PatternColorAsARGB
- PatternColorDataFieldIndex
- PatternColorMapName
- Sizeable
- Specification
- StartDataFieldIndex
- StartSnapTarget
- ThreeDEffect
- UsedAsOverlapLayer
- VerticalOffset
- VerticalOffsetDataFieldIndex
- VerticalOffsetMapName
- Visible
- VisibleInLegend

Methods

- CalculateCurrentWidth
- PutInOrderAfter

Properties

BackColorAsARGB

Property of VcLayer

This property lets you set or retrieve the background color of the layer. Color values have a transparency or alpha value, followed by a value for a red, a blue and a green partition (ARGB). The values range between 0..255. An alpha value of 0 equals complete transparency, whereas 255 represents a completely solid color. When casting an RGB value on an ARGB value, an alpha value of 255 has to be added.

Also see **set/getPatternColorAsARGB**.

If by the property **BackColorMapName** a map is specified, the map will set the background colors in dependence on data.

	Data Type	Explanation

Example Code

```
Dim layer As VcLayer

Set layer = VcGantt1.LayerCollection.LayerByIndex(0)
layer.BackColorAsARGB = &h88FF0A06
```

BackColorDataFieldIndex

Property of VcLayer

This property lets you set or retrieve the data field index to be used with the property **BackColorMapName**. If you set this property to **-1**, no map will be used.

	Data Type	Explanation

Example Code

```
Dim dataTable As VcDataTable
Dim dataRecCltn As VcDataRecordCollection
Dim dataRec1 As VcDataRecord
Dim layer As VcLayer
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry
```

```
Set dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
Set dataRecCltn = dataTable.DataRecordCollection
Set dataRec1 = dataRecCltn.Add("1;Node 1;01.01.14;;8;Red")
VcGantt1.EndLoading

Set mapCltn = VcGantt1.MapCollection
Set map = mapCltn.Add("MapColor")

map.Type = vcColorMap
Set mapEntry = map.CreateEntry
mapEntry.DataFieldValue = "Green"
mapEntry.Color = RGB(0, 255, 0)
Set mapEntry = map.CreateEntry
mapEntry.DataFieldValue = "Red"
mapEntry.Color = RGB(255, 0, 0)
mapCltn.Update

Set layer = VcGantt1.LayerCollection.LayerByIndex(0)
layer.LayerShape = vcRectangleLayer
layer.BackColorMapName = "MapColor"
layer.BackColorDataFieldIndex = 5
VcGantt1.LayerCollection.Update
```

BackColorMapName

Property of VcLayer

This property lets you set or retrieve the name of a color map (type vcColorMap). If set to "", no map will be used. If a map name and additionally a data field index is specified by the property **BackColorDataFieldIndex**, the background color will be set by the map. If no data field entry applies, the background color of the layer specified by the property **BackColorAsARGB** will apply.

If the map holds transparent color values (ARGB values), but a property can only use RGB values, XGantt will display the specified color as solid.

	Data Type	Explanation

Example Code

```
Dim dataTable As VcDataTable
Dim dataRecCltn As VcDataRecordCollection
Dim dataRec1 As VcDataRecord
Dim layer As VcLayer
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry

Set dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
Set dataRecCltn = dataTable.DataRecordCollection
Set dataRec1 = dataRecCltn.Add("1;Node 1;01.01.14;;8;Red")
VcGantt1.EndLoading

Set mapCltn = VcGantt1.MapCollection
```

```
Set map = mapCltn.Add("MapColor")

map.Type = vcColorMap
Set mapEntry = map.CreateEntry
mapEntry.DataFieldValue = "Green"
mapEntry.Color = RGB(0, 255, 0)
Set mapEntry = map.CreateEntry
mapEntry.DataFieldValue = "Red"
mapEntry.Color = RGB(255, 0, 0)
mapCltn.Update

Set layer = VcGantt1.LayerCollection.LayerByIndex(0)
layer.LayerShape = vcRectangleLayer
layer.BackColorMapName = "MapColor"
layer.BackColorDataFieldIndex = 5
VcGantt1.LayerCollection.Update
```

CompletionDataFieldIndex

Property of VcLayer

This property lets you set or retrieve the data field that contains the percentage degree of completion of the layer.

The end date visualized by the layer is calculated from the start date field, the end date field or the duration respectively and the percent complete value. The data of the activity will not be changed.

This property is not available for symbol and bitmap layers.

	Data Type	Explanation
Property value	Long	Index of the data field that contains the degree of completion

Example Code

```
Dim layerCltn As VcLayerCollection
Dim layer As VcLayer

Set layerCltn = VcGantt1.LayerCollection

For Each layer In layerCltn
   layer.CompletionDataFieldIndex = 10
Next
```

DurationDataFieldIndex

Property of VcLayer

This property lets you set or retrieve the data field that contains the duration of the layer.

The unit of the duration will be interpreted in dependency on the time unit specified on the **General** property page.

This property is not available for symbol and bitmap layers.

	Data Type	Explanation
Property value	Long	Index of the data field that contains the duration

Example Code

```
Dim layerCltn As VcLayerCollection
Dim layer As VcLayer

Set layerCltn = VcGantt1.LayerCollection

For Each layer In layerCltn
   layer.DurationDataFieldIndex = 4
Next
```

EndDataFieldIndex

Property of VcLayer

This property lets you set or retrieve the data field that contains the end value of the layer, e.g. Early Start, Late Start, Scheduled Start.

To define a rectangle or line layer you need to specify a start and end field or a duration. If both an end field and a duration are specified, the duration entry overrides the end field entry. When an interaction occurs, not only the

duration field will be updated, but also the end field.

This property is not available for symbol and bitmap layers.

	Data Type	Explanation
Property value	Integer	Index of the data field that contains the end value
	Possible Values:	Data field index

Example Code

```
Dim layerCltn As VcLayerCollection
Dim layer As VcLayer

Set layerCltn = VcGantt1.LayerCollection

For Each layer In layerCltn
   layer.DurationDataFieldIndex = 3
Next
```

EndSnapTarget

Property of VcLayer

This property lets you set or retrieve whether the end date of this layer is to define as snap target.

	Data Type	Explanation
Property value	Boolean	End date of this layer is/is not defined as snap target
	Possible Values:	Group invisible/visible group nodes are/are not visible

FilterName

Property of VcLayer

This property lets you specify the name of the filter that defines what activities the layer is to apply to.

	Data Type	Explanation
Property value	String	Filter name
	Possible Values:	Name of the color map

Example Code

```
Dim layerCltn As VcLayerCollection
Dim layer As VcLayer

Set layerCltn = VcGantt1.LayerCollection

For Each layer In layerCltn
    layer.FilterName = "Milestone"
Next
```

GraphicsFileName

Property of VcLayer

This property lets you set or retrieve the name of a graphics file the content of which is displayed in the layer. The graphics file has to be of one of the below formats:

- *.BMP (Microsoft Windows Bitmap)
- *.EMF (Enhanced Metafile oder Enhanced Metafile Plus)

- *.GIF (Graphics Interchange Format)
- *.JPG (Joint Photographic Experts Group)
- *.PNG (Portable Network Graphics)
- *.TIF (Tagged Image File Format)
- *.VMF (Viewer Metafile)
- *.WMF (Microsoft Windows Metafile, probably with EMF included)

EMF, EMF+, VMF and WMF are vector formats that allow to store a file independent of pixel resolution. All other formats are pixel-oriented and confined to a limited resolution.

The VMF format basically has been deprecated, but it will still be supported for some time to maintain compatibility with existing applications.

For the graphics file to be displayed, independent of the format set here, the property **LayerShape** has to be set to **vcBitmapLayer**.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the graphics file Name of the color map

Example Code

```
Dim dataTable As VcDataTable
Dim dataRecCltn As VcDataRecordCollection
Dim dataRec1 As VcDataRecord
Dim layer As VcLayer
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry

Set dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
Set dataRecCltn = dataTable.DataRecordCollection
Set dataRec1 = dataRecCltn.Add("1;Node 1;01.01.14;;8;Pic1.bmp")
VcGantt1.EndLoading

Set mapCltn = VcGantt1.MapCollection
Set map = mapCltn.Add("MapGraphic")

map.Type = vcGraphicsFileMap
Set mapEntry = map.CreateEntry
mapEntry.GraphicsFileName = "c:\Pic1.bmp"
mapCltn.Update

Set layer = VcGantt1.LayerCollection.LayerByIndex(0)
```

```
layer.LayerShape = vcBitmapLayer
layer.GraphicsFileName = "c:\Pic1.bmp"
layer.GraphicsFileNameMapName = "MapGraphic"
layer.GraphicsFileNameDataFieldIndex = 5
VcGantt1.LayerCollection.Update
```

GraphicsFileNameDataFieldIndex

Property of VcLayer

This property lets you set or retrieve the data field index that has to be specified if the property **GraphicsFileNameMapName** is used. If a valid data field index, but no map is specified, the graphics file name will be read from the data field specified.

For the graphics file to be displayed, the property **LayerShape** has to be set to **vcBitmapLayer**.

	Data Type	Explanation
Property value	Integer	Index of the data field
	Possible Values:	Data field index

Example Code

```
Dim dataTable As VcDataTable
Dim dataRecCltn As VcDataRecordCollection
Dim dataRec1 As VcDataRecord
Dim layer As VcLayer
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry

Set dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
Set dataRecCltn = dataTable.DataRecordCollection
Set dataRec1 = dataRecCltn.Add("1;Node 1;01.01.14;;8;Pic1.bmp")
VcGantt1.EndLoading

Set mapCltn = VcGantt1.MapCollection
Set map = mapCltn.Add("MapGraphic")

map.Type = vcGraphicsFileMap
Set mapEntry = map.CreateEntry
mapEntry.GraphicsFileName = "c:\Pic1.bmp"
mapCltn.Update

Set layer = VcGantt1.LayerCollection.LayerByIndex(0)
layer.LayerShape = vcBitmapLayer
layer.GraphicsFileName = "c:\Pic1.bmp"
layer.GraphicsFileNameMapName = "MapGraphic"
layer.GraphicsFileNameDataFieldIndex = 5
VcGantt1.LayerCollection.Update
```

GraphicsFileNameMapName

Property of VcLayer

This property lets you set or retrieve the name of a map of the type **vcGraphicsFileMap** or "". Only if a name and a data field index are specified in the property **GraphicsFileNameDataFieldIndex**, the graphics will be controlled by the map. If no data field entry applies, the graphics specified in the property **GraphicsFileName** will be displayed.

For the graphics file to be displayed, the property **LayerShape** has to be set to **vcBitmapLayer**.

	Data Type	Explanation
Property value	String	Name of the graphics map
	Possible Values:	Name of the color map

Example Code

```
Dim dataTable As VcDataTable
Dim dataRecCltn As VcDataRecordCollection
Dim dataRec1 As VcDataRecord
Dim layer As VcLayer
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry

Set dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
Set dataRecCltn = dataTable.DataRecordCollection
Set dataRec1 = dataRecCltn.Add("1;Node 1;01.01.14;;8;Pic1.bmp")
VcGantt1.EndLoading

Set mapCltn = VcGantt1.MapCollection
Set map = mapCltn.Add("MapGraphic")

map.Type = vcGraphicsFileMap
Set mapEntry = map.CreateEntry
mapEntry.GraphicsFileName = "c:\Pic1.bmp"
mapCltn.Update

Set layer = VcGantt1.LayerCollection.LayerByIndex(0)
layer.LayerShape = vcBitmapLayer
layer.GraphicsFileName = "c:\Pic1.bmp"
layer.GraphicsFileNameMapName = "MapGraphic"
layer.GraphicsFileNameDataFieldIndex = 5
VcGantt1.LayerCollection.Update
```

Height

Property of VcLayer

This property lets you set or retrieve the height of the layer.

	Data Type	Explanation
Property value	Long	Height by 1/100 mm

HeightDataFieldIndex

Property of VcLayer

This property lets you set or retrieve the data field index that has to be specified if the property **HeightMapName** is used. If you set this property to **-1**, no map will be used.

This property will only become effective after the layer collection was updated by the method **VcLayerCollection.Update()**.

	Data Type	Explanation
Property value	Long	Data field index

Example Code

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim layer As VcLayer

Set layer = VcGantt1.LayerCollection.FirstLayer()
Set mapCltn = VcGantt1.MapCollection
mapCltn.SelectMaps (vcMillimeterMap)
Set map = mapCltn.FirstMap
layer.HeightMapName = map.Name
layer.HeightDataFieldIndex = VcGantt1.DetectFieldIndex("Maindata",
"LayerHeight")
VcGantt1.LayerCollection.Update
```

HeightMapName

Property of VcLayer

This property lets you set or retrieve the name of a millimeter map (type vc-MillimeterMap). If set to "", no map will be used. If a map name and additionally a data field index is specified in the property **HeightDataField-Index**, then the height is controlled by the map. If no data field entry applies, the height of the layer that is specified in the property **Height** will be used.

This property will only become effective after the layer collection was updated by the method **VcLayerCollection.Update()**.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the millimetre map Name of the color map

Example Code

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim layer As VcLayer

Set layer = VcGantt1.LayerCollection.FirstLayer()
Set mapCltn = VcGantt1.MapCollection
mapCltn.SelectMaps (vcMillimeterMap)
Set map = mapCltn.FirstMap
layer.HeightMapName = map.Name
layer.HeightDataFieldIndex = VcGantt1.DetectFieldIndex("Maindata",
"LayerHeight")
VcGantt1.LayerCollection.Update
```

HorizontalOffset

Property of VcLayer

This property lets you set or retrieve the horizontal offset of the layer. This is only possible for symbol or bitmap layers. If you set an offset for other layer shapes, this will be without effect.

	Data Type	Explanation
Property value	Integer **Possible Values:**	Horizontal offset in % -50 ... 50 Data field index

LabelSizeDependence

Property of VcLayer

This property lets you set or retrieve, whether and in which way the size of the label is to depend on the size of the layer.

	Data Type	Explanation
Property value	LabelSizeDependenceEnum **Possible Values:** vcFixedToBar 1 vcTextHeightAndWidthIndependent 79 vcTextHeightIndependent 39	Dependence of the label on the layer size restricted by layer siz independent on text height and width independent on text height

vcTextWidthIndependent 40	independent on text width

Example Code

```
Dim layerCltn As VcLayerCollection
Dim layer As VcLayer

Set layerCltn = VcGantt1.LayerCollection
Set layer = layerCltn.LayerByName("Start-End")
layer.LabelSizeDependence = vcFixedToBar
```

LayerFormat

Read Only Property of VcLayer

This property lets you enquire the layer format of this layer.

	Data Type	Explanation
Property value	VcLayerFormat	Layer format

LayerShape

Property of VcLayer

This property lets you set or retrieve the shape of the layer. In the symbols below, black sections can be color-coded (please see **BackColorAsARGB**, **Pattern** und **PatternColor**).).

	Data Type	Explanation
Property value	LayerShapeEnum	Layer shape
	Possible Values:	
	vcAllRoundedRectangleLayer 61441	All corners rounded
	vcBitmapLayer 103007	Layer form bitmap <Bitmap-Layer>
	vcInvisibleSymbolLayer 101000	Layer invisible <unsichtbares Symb
	vcLineLayer 2	Layershape line
	vcNAndSERoundedRectangleLayer 45057	Rounded top left, top right and bottom right corner
	vcNAndSWRoundedRectangleLayer 28673	Rounded top left, top right and bottom left corner
	vcNEAndSERoundedRectangleLayer 40961	Rounded top right and bottom right corner
	vcNEAndSRoundedRectangleLayer 57345	Rounded bottom left, bottom right and top right corner

vcNEAndSWRoundedRectangleLayer 24577	Rounded bottom left and top right corner
vcNERoundedRectangleLayer 8193	Rounded top right corner
vcNRoundedRectangleLayer 12289	Rounded top right and top left corner
vcNWAndSERoundedRectangleLayer 36865	Rounded top left and bottom right corner
vcNWAndSRoundedRectangleLayer 52349	Rounded top left, bottom left and bottom right corner
vcNWAndSWRoundedRectangleLayer 20481	Rounded bottom left and top left corner
vcNWRoundedRectangleLayer 4097	Rounded top left corner
vcRectangleLayer 1	Rectangle layer
vcRectangleTriangleLeft 1938	Rectangle layer, tip pointing left
vcRectangleTriangleLeftRight 1939	Rectangle layer, tip pointing to both sides!!! Datei E:\Manuals\Generierung\xHandbuch\xBilder\div\Layershape_ nicht gefunden!!!
vcRectangleTriangleRight 1939	Rectangle layer, tip pointing right!!! Datei E:\Manuals\Generierung\xHandbuch\xBilder\div\Layershape_ nicht gefunden!!!
vcSERoundedRectangleLayer 32769	Rounded bottom right corner
vcSRoundedRectangleLayer 49153	Rounded bottom right and bottom left corner
vcSummaryBar1 1858	Summary bar
vcSummaryBar2 1859	Summary bar
vcSummaryBar3 1860	Summary bar
vcSummaryBar4 1861	Summary bar
vcSWRoundedRectangleLayer 16385	Rounded bottom left corner
vcSymbolLayer1 101001	Arrow down
vcSymbolLayer10 101010	Square
vcSymbolLayer11 101032	Circle
vcSymbolLayer12 101033	Arrow down in circle
vcSymbolLayer13 101034	Triangle in circle, tip pointing down

vcSymbolLayer14 101035	Pointed right bracket in circle
vcSymbolLayer15 101036	Narrow triangle in circle, tip pointing up
vcSymbolLayer16 101037	Triangle in circle, tip pointing right
vcSymbolLayer17 101038	Triangle in circle, tip pointing left
vcSymbolLayer18 101039	Square in circle, sitting on tip
vcSymbolLayer19 101040	Two narrow triangles in circle, position horizontal, tips pointing
vcSymbolLayer2 101002	Triangle, tip pointing downward
vcSymbolLayer20 101041	Narrow triangle in circle, tip pointing down
vcSymbolLayer21 101042	Square in circle
vcSymbolLayer22 103001	Circle
vcSymbolLayer23 102031	Arrow up
vcSymbolLayer24 102034	Triangle, tip pointing up
vcSymbolLayer25 102016	Pointed bracket, left one
vcSymbolLayer26 102051	Arrow up in circle
vcSymbolLayer27 102054	Triangle in circle, tip pointing up
vcSymbolLayer3 101003	Right pointed bracket
vcSymbolLayer4 101004	Narrow triangle, tip pointing upward
vcSymbolLayer5 101005	Triangle, tip pointing right
vcSymbolLayer6 101006	Triangle, tip pointing left
vcSymbolLayer7 101007	Square sitting on tip
vcSymbolLayer8 101008	Two narrow triangles, position horizontal, tips pointing to cent
vcSymbolLayer9 101009	Narrow triangle, tip pointing down
vcTriangleBottomLeftLayer 1566	Triangle, tip pointing left

vcTriangleBottomRightLayer 1564	Triangle, tip pointing right

LegendText

Property of VcLayer

This property lets you set or retrieve the legend text of a layer. When set to "", the layer name (property **Name**) will be displayed.

	Data Type	Explanation
Property value	String **Possible Values:**	Legend text of the layer **Default value:** " " (content of the property **Name**) Name of the color map

LineColor

Property of VcLayer

This property lets you set or retrieve the color of the (border) line of the layer.

	Data Type	Explanation

LineColorDataFieldIndex

Property of VcLayer

This property lets you set or retrieve the data field index to be used with a map specified by the property **LineColorMapName**. If you set this property to -1, no map will be used.

	Data Type	Explanation

LineColorMapName

Property of VcLayer

This property lets you set or retrieve the name of a map for the line color. If set to "" or if the property **LineColorDataFieldIndex** is set to **-1**, then no map will be used.

	Data Type	Explanation

LineThickness

Property of VcLayer

This property lets you set or retrieve the thickness of the (border) line of the layer.

If you set this property to values between 1 and 4, an absolute line thickness is defined in pixels. Irrespective of the zoom factor a line will always show the same line thickness in pixels. When printing though, the line thickness is adapted for the sake of legibility and becomes dependent of the zoom factor:

Value	Points	mm
1	1/2 point	0.09 mm
2	1 point	0.18 mm
3	3/2 points	0.26 mm
4	2 points	0.35 mm

A point equals 1/72 inch and represents the unit of the font size.

If you set this property to values between 5 and 1,000, the line thickness is defined in 1/100 mm, so the lines will be displayed in a true thickness in pixels that depends on the zoom factor.

	Data Type	Explanation
Property value	Integer	Line thickness LineType {1...4}: line thickness in pixels LineType {5...1000}: line thickness in 1/100 mm **Default value:** As defined in the dialog
	Possible Values:	Data field index

LineType

Property of VcLayer

This property lets you set or retrieve the type of the (border) line of the layer.

	Data Type	Explanation
Property value	LineTypeEnum	Line type ({0...255},{0...255},{0...255})
	Possible Values:	
	vcDashed 4	Line dashed
	vcDashedDotted 5	Line dashed-dotted
	vcDotted 3	Line dotted
	vcLineType0 100	Line Type 0
	vcLineType1 101	Line Type 1
	vcLineType10 110	Line Type 10
	vcLineType11 111	Line Type 11
	vcLineType12 112	Line Type 12
	vcLineType13 113	Line Type 13
	vcLineType14 114	Line Type 14
	vcLineType15 115	Line Type 15
	vcLineType16 116	Line Type 16
	vcLineType17 117	Line Type 17
	vcLineType18 118	Line Type 18
	vcLineType2 102	Line Type 2
	vcLineType3 103	Line Type 3
	vcLineType4 104	Line Type 4
	vcLineType5 105	Line Type 5

vcLineType6 106	Line Type 6
vcLineType7 107	Line Type 7
vcLineType8 108	Line Type 8
vcLineType9 109	Line Type 9
vcNone 1	No line type
vcNotSet -1	No line type assigned
vcSolid 2	Line solid

MaximumEndDataFieldIndex

Property of VcLayer

If this property is set to a valid field index, the date and time of the corresponding field are considered as upper limit for the end time of the layer when a layer or a node is moved interactively .

This property can also be set in the **Edit Layer** dialog.

	Data Type	Explanation

MinimumStartDataFieldIndex

Property of VcLayer

If this property is set to a valid field index, the date and time of the corresponding field are considered as lower limit for the start time of the layer when a layer or a node is moved interactively .

This property can also be set in the **Edit Layer** dialog.

	Data Type	Explanation
Property value	Long	Data field index for earliest start time **Default value:** -1

Moveable

Property of VcLayer

This property lets you set or retrieve whether a layer can be moved interactively.

	Data Type	Explanation
Property value	Boolean	Moveable (True)/ not moveable (False) **Default value:** True
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
Dim layer1 As VcLayer

Set layer1 = VcGantt1.Layer.LayerByName("layer1")
If chkMoveable.Value = vbUnchecked Then
   layer1.Moveable = False
Else
   layer1.Moveable = True
End If
```

Name

Read Only Property of VcLayer

This property lets you retrieve the name of a layer.

	Data Type	Explanation
Property value	String	Name of the layer
	Possible Values:	Name of the color map

Example Code

```
Dim layerCltn As VcLayerCollection
Dim layer As VcLayer

Set layerCltn = VcGantt1.LayerCollection

For Each layer In layerCltn
    List1.AddItem layer.Name
Next layer
```

NonWorkInterval

Property of VcLayer

This property lets you set or retrieve the name of a color map (type vcColorMap). If set to "", no map will be used. Only if a map name and a data field index are specified in the property **PatternColorDataFieldIndex**, the pattern color is controlled by the map. If no data field entry applies, the pattern color of the layer that is specified in the property **PatternColorAs-ARGB** will be used.

If the map holds transparent color values (ARGB values), but a property can only use RGB values, XGantt will display the specified color as solid.

	Data Type	Explanation

NonWorkIntervalBackColorAsARGB

Property of VcLayer

This property lets you set or retrieve the background color of the layer. Color values have a transparency or alpha value, followed by a value for a red, a blue and a green partition (ARGB). The values range between 0..255. An alpha value of 0 equals complete transparency, whereas 255 represents a completely solid color. When casting an RGB value on an ARGB value, an alpha value of 255 has to be added.

Also see **set/getPatternColorAsARGB**.

If by the property **BackColorMapName** a map is specified, the map will set the background colors in dependence on data.

	Data Type	Explanation
Property value	Color	ARGB color values ({0...255},{0...255},0...255},{0...255})

Example Code

```
Dim layer As VcLayer

Set layer = VcGantt1.LayerCollection.LayerByIndex(0)
layer.BackColorAsARGB = &h88FF0A06
```

NonWorkIntervalBackColorDataFieldIndex

Property of VcLayer

This property lets you set or retrieve the data field index to be used with the property **BackColorMapName**. If you set this property to **-1**, no map will be used.

	Data Type	Explanation
Property value	Integer	Data field index
	Possible Values:	Data field index

Example Code

```
Dim dataTable As VcDataTable
Dim dataRecCltn As VcDataRecordCollection
Dim dataRec1 As VcDataRecord
Dim layer As VcLayer
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry

Set dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
Set dataRecCltn = dataTable.DataRecordCollection
Set dataRec1 = dataRecCltn.Add("1;Node 1;01.01.14;;8;Red")
VcGantt1.EndLoading

Set mapCltn = VcGantt1.MapCollection
Set map = mapCltn.Add("MapColor")

map.Type = vcColorMap
Set mapEntry = map.CreateEntry
mapEntry.DataFieldValue = "Green"
mapEntry.Color = RGB(0, 255, 0)
Set mapEntry = map.CreateEntry
mapEntry.DataFieldValue = "Red"
mapEntry.Color = RGB(255, 0, 0)
mapCltn.Update

Set layer = VcGantt1.LayerCollection.LayerByIndex(0)
layer.LayerShape = vcRectangleLayer
layer.BackColorMapName = "MapColor"
layer.BackColorDataFieldIndex = 5
VcGantt1.LayerCollection.Update
```

NonWorkIntervalBackColorMapName

Property of VcLayer

This property lets you set or retrieve the name of a color map (type vcColorMap). If set to "", no map will be used. If a map name and additionally a data field index is specified by the property **BackColorDataFieldIndex**, the background color will be set by the map. If no data field

entry applies, the background color of the layer specified by the property **BackColorAsARGB** will apply.

If the map holds transparent color values (ARGB values), but a property can only use RGB values, XGantt will display the specified color as solid.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the color map Name of the color map

Example Code

```
Dim dataTable As VcDataTable
Dim dataRecCltn As VcDataRecordCollection
Dim dataRec1 As VcDataRecord
Dim layer As VcLayer
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry

Set dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
Set dataRecCltn = dataTable.DataRecordCollection
Set dataRec1 = dataRecCltn.Add("1;Node 1;01.01.14;;8;Red")
VcGantt1.EndLoading

Set mapCltn = VcGantt1.MapCollection
Set map = mapCltn.Add("MapColor")

map.Type = vcColorMap
Set mapEntry = map.CreateEntry
mapEntry.DataFieldValue = "Green"
mapEntry.Color = RGB(0, 255, 0)
Set mapEntry = map.CreateEntry
mapEntry.DataFieldValue = "Red"
mapEntry.Color = RGB(255, 0, 0)
mapCltn.Update

Set layer = VcGantt1.LayerCollection.LayerByIndex(0)
layer.LayerShape = vcRectangleLayer
layer.BackColorMapName = "MapColor"
layer.BackColorDataFieldIndex = 5
VcGantt1.LayerCollection.Update
```

NonWorkIntervalLineColor

Property of VcLayer

This property lets you set or retrieve the color of the (border) line of the layer.

	Data Type	Explanation
Property value	Color	RGB color values ({0...255},{0...255},{0...255})

NonWorkIntervalLineColorDataFieldIndex

Property of VcLayer

This property lets you set or retrieve the data field index to be used with a map specified by the property **LineColorMapName**. If you set this property to -1, no map will be used.

	Data Type	Explanation
Property value	Integer **Possible Values:**	Data field index Data field index

NonWorkIntervalLineColorMapName

Property of VcLayer

This property lets you set or retrieve the name of a map for the line color. If set to "" or if the property **LineColorDataFieldIndex** is set to **-1**, then no map will be used.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the color map Name of the color map

NonWorkIntervalLineThickness

Property of VcLayer

This property lets you set or retrieve the thickness of the (border) line of the layer.

If you set this property to values between 1 and 4, an absolute line thickness is defined in pixels. Irrespective of the zoom factor a line will always show

the same line thickness in pixels. When printing though, the line thickness is adapted for the sake of legibility and becomes dependent of the zoom factor:

Value	Points	mm
1	1/2 point	0.09 mm
2	1 point	0.18 mm
3	3/2 points	0.26 mm
4	2 points	0.35 mm

A point equals 1/72 inch and represents the unit of the font size.

If you set this property to values between 5 and 1,000, the line thickness is defined in 1/100 mm, so the lines will be displayed in a true thickness in pixels that depends on the zoom factor.

	Data Type	Explanation
Property value	Integer	Line thickness LineType {1...4}: line thickness in pixels LineType {5...1000}: line thickness in 1/100 mm **Default value:** As defined in the dialog
	Possible Values:	Data field index

NonWorkIntervalLineType

Property of VcLayer

This property lets you set or retrieve the type of the (border) line of the layer.

	Data Type	Explanation
Property value	LineTypeEnum	Line type ({0...255},{0...255},{0...255})
	Possible Values:	
	vcDashed 4	Line dashed
	vcDashedDotted 5	Line dashed-dotted
	vcDotted 3	Line dotted
	vcLineType0 100	Line Type 0
	vcLineType1 101	Line Type 1
	vcLineType10 110	Line Type 10

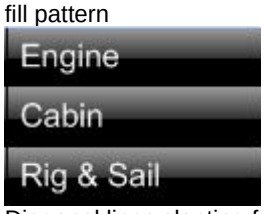
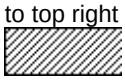
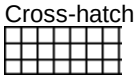
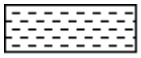
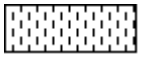
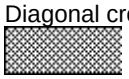
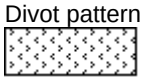
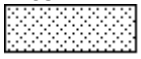
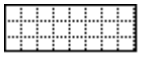
vcLineType11 111	Line Type 11
vcLineType12 112	Line Type 12
vcLineType13 113	Line Type 13
vcLineType14 114	Line Type 14
vcLineType15 115	Line Type 15
vcLineType16 116	Line Type 16
vcLineType17 117	Line Type 17
vcLineType18 118	Line Type 18
vcLineType2 102	Line Type 2
vcLineType3 103	Line Type 3
vcLineType4 104	Line Type 4
vcLineType5 105	Line Type 5
vcLineType6 106	Line Type 6
vcLineType7 107	Line Type 7
vcLineType8 108	Line Type 8
vcLineType9 109	Line Type 9
vcNone 1	No line type
vcNotSet -1	No line type assigned
vcSolid 2	Line solid

NonWorkIntervalPattern

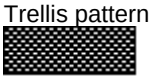
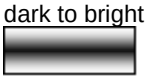
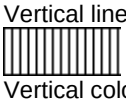
Property of VcLayer

This property lets you set or retrieve the pattern of the layer. If in the property **PatternMapName** a map is specified, this map will control the pattern in dependance on the data.

	Data Type	Explanation
Property value	FillPatternEnum	Pattern type **Default value:** As defined in the dialog
	Possible Values: vc05PercentPattern... vc90PercentPattern 01 - 11	Dots in foreground color on background color, the density of the foreground pattern increasing with the percentage

vcAeroGlassPattern 40	Vertical color gradient in the color of the fill pattern 
vcBDiagonalPattern 5	Diagonal lines slanting from bottom left to top right 
vcCrossPattern 6	Cross-hatch pattern 
vcDarkDownwardDiagonalPattern 2014	Diagonal lines slanting from top left to bottom right; spaced 50% closer than vcFDiagonalPattern and of twice the line width 
vcDarkHorizontalPattern 2023	Horizontal lines spaced 50% closer than vcHorizontalPattern and of twice the line width 
vcDarkUpwardDiagonalPattern 2015	Diagonal lines slanting from bottom left to top right, spaced 50% closer than vcBDiagonalPattern and of twice the line width 
vcDarkVerticalPattern 2022	Vertical lines spaced 50% closer than vcVerticalPattern and of of twice the line width 
vcDashedDownwardDiagonalPattern 2024	Dashed diagonal lines from top left to bottom right 
vcDashedHorizontalPattern 2026	Dashed horizontal lines 
vcDashedUpwardDiagonalPattern 2025	Dashed diagonal lines from bottom left to top right 
vcDashedVerticalPattern 2027	Dashed vertical lines 
vcDiagCrossPattern 7	Diagonal cross-hatch pattern, small 
vcDiagonalBrickPattern 2032	Diagonal brick pattern 
vcDivotPattern 2036	Divot pattern 
vcDottedDiamondPattern 2038	Diagonal cross-hatch pattern of dotted lines 
vcDottedGridPattern 2037	Cross-hatch pattern of dotted lines 

vcFDiagonalPattern 4	Diagonal lines slanting from top left to bottom right
vcHorizontalBrickPattern 2033	Horizontal brick pattern
vcHorizontalGradientPattern 52	Horizontal color gradient
vcHorizontalPattern 3	Horizontal lines
vcLargeCheckerboardPattern 2044	Checkerboard pattern showing squares of twice the size of vcSmallChecker-BoardPattern
vcLargeConfettiPattern 2029	Confetti pattern, large
vcLightDownwardDiagonalPattern 2012	Diagonal lines slanting to from top left to bottom right; spaced 50% closer than vcBDiagonalPattern
vcLightHorizontalPattern 2019	Horizontal lines spaced 50% closer than vcHorizontalPattern
vcLightUpwardDiagonalPattern 2013	Diagonal lines slanting from bottom left to top right, spaced 50% closer than vcBDiagonalPattern
vcLightVerticalPattern 2018	Vertical lines spaced 50% closer than vcVerticalPattern
vcNarrowHorizontalPattern 2021	Horizontal lines spaced 75 % closer than vcHorizontalPattern
vcNarrowVerticalPattern 2020	Vertical lines spaced 75% closer than vcVerticalPattern
vcNoPattern 1276	No fill pattern
vcOutlinedDiamondPattern 2045	Diagonal cross-hatch pattern, large
vcPlaidPattern 2035	Plaid pattern
vcShinglePattern 2039	Diagonal shingle pattern
vcSmallCheckerBoardPattern 2043	Checkerboard pattern
vcSmallConfettiPattern 2028	Confetti pattern
vcSmallGridPattern 2042	Cross-hatch pattern spaced 50% closer than vcCrossPattern
vcSolidDiamondPattern 2046	Checkerboard pattern showing diagonal squares

vcSpherePattern 2041	Checkerboard of spheres 
vcTrellisPattern 2040	Trellis pattern 
vcVerticalBottomLightedConvexPattern 43	Vertical color gradient from dark to bright 
vcVerticalConcavePattern 40	Vertical color gradient from dark to bright to dark 
vcVerticalConvexPattern 41	Vertical color gradient from bright to dark to bright 
vcVerticalGradientPattern 62	Vertical color gradient 
vcVerticalPattern 2	Vertical lines 
vcVerticalTopLightedConvexPattern 42	Vertical color gradient from bright to dark 
vcWavePattern 2031	Horizontal wave pattern 
vcWeavePattern 2034	Interwoven stripe pattern 
vcWideDownwardDiagonalPattern 2016	Diagonal lines slanting from top left to bottom right, showing the same spacing but three times the line width of vcF-DiagonalPattern 
vcWideUpwardDiagonalPattern 2017	Diagonal lines slanting from bottom left to top right right, showing the same spacing but three times the line width of vcBDiagonalPattern 
vcZigZagPattern 2030	Horizontal zig-zag lines 

NonWorkIntervalPatternColorAsARGB

Property of VcLayer

This property lets you set or retrieve the pattern color of the layer. Color values have a transparency or alpha value, followed by a value for a red, a blue and a green partition (ARGB). The values range between 0..255. An alpha value of 0 equals complete transparency, whereas 255 represents a completely solid color. When casting an RGB value on an ARGB value, an alpha value of 255 has to be added.

Also see **set/getBackColorAsARGB**.

If in the property **PatternColorMapName** a map is specified, the map will control the pattern color in dependence of the data.

	Data Type	Explanation
Property value	Color	RGB color values ({0...255},{0...255},0...255},{0...255})

Example Code

```
Dim layer As VcLayer

Set layer = VcGantt1.LayerCollection.LayerByIndex(0)
layer.PatternColorAsARGB = &h88FF0A06
```

NonWorkIntervalPatternColorDataFieldIndex

Property of VcLayer

This property lets you set or retrieve the data field index that has to be specified if the property **PatternColorMapName** is used. If you set this property to **-1**, no map will be used.

	Data Type	Explanation
Property value	Integer **Possible Values:**	Data field index Data field index

NonWorkIntervalPatternColorDataFieldIndex

Property of VcLayer

This property lets you set or retrieve the data field index that has to be specified if the property **PatternColorMapName** is used. If you set this property to **-1**, no map will be used.

	Data Type	Explanation
Property value	Integer **Possible Values:**	Data field index Data field index

NonWorkIntervalPatternDataFieldIndex

Property of VcLayer

This property lets you set or retrieve the data field index to be used together with the property **PatternMapName**. If you set this property to **-1**, no map will be used.

	Data Type	Explanation
Property value	Integer **Possible Values:**	Data field index Data field index

Example Code

```
Dim dataTable As VcDataTable
Dim dataRecCltn As VcDataRecordCollection
Dim dataRec1 As VcDataRecord
Dim layer As VcLayer
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry

Set dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
Set dataRecCltn = dataTable.DataRecordCollection
Set dataRec1 = dataRecCltn.Add("1;Node 1;01.01.14;;8;Horizontal")
VcGantt1.EndLoading

Set mapCltn = VcGantt1.MapCollection
Set map = mapCltn.Add("MapPattern")

map.Type = vcPatternMap
Set mapEntry = map.CreateEntry
mapEntry.DataFieldValue = "Diagonal"
mapEntry.Pattern = vcBDiagonalPattern
Set mapEntry = map.CreateEntry
mapEntry.DataFieldValue = "Horizontal"
mapEntry.Pattern = vcHorizontalPattern
mapCltn.Update

Set layer = VcGantt1.LayerCollection.LayerByIndex(0)
layer.PatternMapName = "MapPattern"
layer.PatternDataFieldIndex = 5
VcGantt1.LayerCollection.Update
```

NonWorkIntervalPatternMapName

Property of VcLayer

This property lets you set or retrieve the name of a pattern map (type vcPatternMap). If set to "", no map will be used. Only if a map name and additionally a data field index are specified in the property **PatternData-FieldIndex**, the pattern is controlled by the map. If no data field entry applies, the pattern of the layer that is specified in the property **Pattern** will be used.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the pattern map Name of the color map

Example Code

```
Dim dataTable As VcDataTable
Dim dataRecCltn As VcDataRecordCollection
Dim dataRec1 As VcDataRecord
Dim layer As VcLayer
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry

Set dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
Set dataRecCltn = dataTable.DataRecordCollection
Set dataRec1 = dataRecCltn.Add("1;Node 1;01.01.14;;8;Horizontal")
VcGantt1.EndLoading

Set mapCltn = VcGantt1.MapCollection
Set map = mapCltn.Add("MapPattern")

map.Type = vcPatternMap
Set mapEntry = map.CreateEntry
mapEntry.DataFieldValue = "Diagonal"
mapEntry.Pattern = vcBDiagonalPattern
Set mapEntry = map.CreateEntry
mapEntry.DataFieldValue = "Horizontal"
mapEntry.Pattern = vcHorizontalPattern
mapCltn.Update

Set layer = VcGantt1.LayerCollection.LayerByIndex(0)
layer.PatternMapName = "MapPattern"
layer.PatternDataFieldIndex = 5
VcGantt1.LayerCollection.Update
```

NonWorkIntervalShape

Property of VcLayer

This property lets you set or retrieve the form of non work intervals in rectangle layers. It can also be set in the **Edit layer** dialog.

	Data Type	Explanation
Property value	NonWorkIntervalShapeEnum **Possible Values:** vcEmptyArea 2 vcLine 1 vcNo 0 vcRectangle 112	Form of non work intervals in rectangle layers work free intervals are displayed as empty area work free intervals are displayed as line work free intervals are not displayed work free intervals are displayed as rectangle

ObjectDrawEventsEnabled

Property of VcLayer

If this property is set to **true**, the events **OnObjectDraw and OnObjectDrawCompleteEx** are enabled for nodes which are drawn with this layer or for annotation ribbons.

	Data Type	Explanation
Property value	Boolean	ObjectDraw events enabled (True) / disabled (False) **Default value:** False
	Possible Values:	Group invisible/visible group nodes are/are not visible

PatternColorAsARGB

Property of VcLayer

This property lets you set or retrieve the pattern color of the layer. Color values have a transparency or alpha value, followed by a value for a red, a blue and a green partition (ARGB). The values range between 0..255. An alpha value of 0 equals complete transparency, whereas 255 represents a completely solid color. When casting an RGB value on an ARGB value, an alpha value of 255 has to be added.

Also see **set/getBackColorAsARGB**.

If in the property **PatternColorMapName** a map is specified, the map will control the pattern color in dependence of the data.

	Data Type	Explanation

Example Code

```
Dim layer As VcLayer

Set layer = VcGantt1.LayerCollection.LayerByIndex(0)
layer.PatternColorAsARGB = &h88FF0A06
```

PatternColorDataFieldIndex

Property of VcLayer

This property lets you set or retrieve the data field index that has to be specified if the property **PatternColorMapName** is used. If you set this property to **-1**, no map will be used.

	Data Type	Explanation

PatternColorMapName

Property of VcLayer

This property lets you set or retrieve the name of a color map (type vcColorMap). If set to "", no map will be used. Only if a map name and a data field index are specified in the property **PatternColorDataFieldIndex**, the pattern color is controlled by the map. If no data field entry applies, the pattern color of the layer that is specified in the property **PatternColorAs-ARGB** will be used.

If the map holds transparent color values (ARGB values), but a property can only use RGB values, XGantt will display the specified color as solid.

	Data Type	Explanation

Sizeable

Property of VcLayer

This property lets you set or retrieve whether the layer size can be changed interactively.

	Data Type	Explanation
Property value	LayerSizeabilityEnum	mode of layer sizeability **Default value:** True

Example Code

```
Dim layer1 As VcLayer

Set layer1 = VcGantt1.Layer.LayerByName("layer1")
```

```
If chkSizeable.Value = vbUnchecked Then
   layer1.Sizeable = vcSizeableNone
Else
   layer1.Sizeable = vcSizeableLeftRight
End If
```

Specification

Read Only Property of VcLayer

This property lets you retrieve the specification of a layer. A specification is a string that contains legible ASCII characters from 32 to 127 only, so it can be stored without problems to text files or data bases. This allows for persistency. A specification can be used to create a layer by the method **VcLayerCollection.AddBySpecification**.

	Data Type	Explanation
Property value	String	Specification of the layer
	Possible Values:	Name of the color map

StartDataFieldIndex

Property of VcLayer

This property lets you set or retrieve the data field that contains the start value of the layer, e.g. Early Start, Late Start, Scheduled Start.

	Data Type	Explanation
Property value	Integer	Index of the data field that contains the start value
	Possible Values:	Data field index

StartSnapTarget

Property of VcLayer

This property lets you set or retrieve whether the start date of this layer is to define as snap target.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	Start date of this layer is/is not defined as snap target Group invisible/visible group nodes are/are not visible

ThreeDEffect

Property of VcLayer

This property lets you set or retrieve whether the layer is highlighted by a 3D effect.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	3D effect switched on (True)/switched off (False) **Default value:** False Group invisible/visible group nodes are/are not visible

UsedAsOverlapLayer

Property of VcLayer

This property lets you set or retrieve whether this layer is to be used as an overlap layer. Overlap layers occur to indicate whether two different nodes overlap. They grow and shrink correspondingly to the size of the overlapping parts and therefore indicate the degree of hiding. (Cf. also **VcGantt.OverlapLayerEnabled** and **VcGantt.OverlapLayerName**).

	Data Type	Explanation
Property value	Boolean **Possible Values:**	**True**: layer is used as an overlap layer; **False**: layer is not used as an overlap layer. **Default value:** False Group invisible/visible group nodes are/are not visible

Example Code

```
Dim layer1 As VcLayer

Set layer1 = VcGantt1.Layer.LayerByName("layer1")
```

```
If chkOverlapLayer.Value = vbUnchecked Then
   layer1.UsedAsOverlapLayer = False
Else
   layer1.UsedAsOverlapLayer = True
End If
```

VerticalOffset

Property of VcLayer

This property lets you set or retrieve the vertical offset of the layer. If in the property **VerticalOffsetMapName** a map is specified, this map will control the vertical offset in dependance on the data.

	Data Type	Explanation
Property value	Long	Vertical offset by 1/100 mm

VerticalOffsetDataFieldIndex

Property of VcLayer

This property lets you set or retrieve the data field index

that has to be specified if the property **VerticalOffsetMapName** is used. If you set this property to **-1**, no map will be used.

	Data Type	Explanation
Property value	Integer	Data field index
	Possible Values:	Data field index

VerticalOffsetMapName

Property of VcLayer

This property lets you set or retrieve the name of a millimeter map (type vcMillimeterMap). If set to "", no map will be used. If a map name and additionally a data field index is specified in the property **VerticalOffset-DataFieldIndex**, then the vertical offset is controlled by the map. If no data field entry applies, the vertical offset of the layer that is specified in the property **VerticalOffset** will be used.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the millimetre map Name of the color map

Visible

Property of VcLayer

This property lets you set or retrieve whether a layer is visible.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	Layer visible/invisible **Default value:** True Group invisible/visible group nodes are/are not visible

Example Code

```
Dim layerCltn As VcLayerCollection
Dim layer As VcLayer

Set layerCltn = VcGantt1.LayerCollection
Set layer = layerCltn.LayerByName("Start-End")
layer.Visible = False
```

VisibleInLegend

Property of VcLayer

This property lets you set or retrieve whether a layer object is to be visible in the legend. This property also can be set by the **Specify Bar Appearance** dialog.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	layer visible in legend (True)/ invisible in legend (False) **Default value:** True Group invisible/visible group nodes are/are not visible

Example Code

```
Dim layerCltn As VcLayerCollection
Dim layer As VcLayer
```

```
Set layerCltn = VcGantt1.layerCollection
Set layer = layerCltn.layerByName("Standard")

layer.VisibleInLegend = False
```

Methods

CalculateCurrentWidth

Method of VcLayer

This method calulates the current width of the layer which belongs to the layer definition of the node specified. The width unit is 1/100 mm. If no layer in the layer definition of the node is visible, for example due to filter conditions, **-1** will be returned.

	Data Type	Explanation
Parameter: ⇨ node	VcNode	Node, in the layer definition of which the layer is looked for.
Return value	Long	Width of the layer in 1/100 mm

PutInOrderAfter

Method of VcLayer

This method lets you set the layer behind a layer specified by name, within the LayerCollection. If you set the name to "", the layer will be put in the first position. The order of the layers determines the order by which they are displayed.

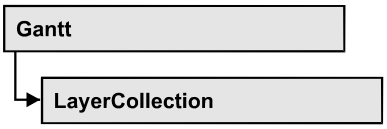
	Data Type	Explanation
Parameter: ⇨ refName	String **Possible Values:**	Name of the layer behind which the current layer is to be put. Name of the color map
Return value	Void	

Example Code

```
Dim layerCltn As VcLayerCollection
Dim layer1 As VcLayer
```

```
Dim layer2 As VcLayer

layerCltn = VcGantt1.LayerCollection()
layer1 = layerCltn.Add("layer1")
layer2 = layerCltn.Add("layer2")
layer1.PutInOrderAfter("layer2")
layerCltn.Update()
```

7.49 VcLayerCollection



The LayerCollection object automatically contains all available layers. You can access all objects in an iterative loop by **For Each layer In Layer-Collection** or by the methods **First...** and **Next...**. You can access a single layer using the methods **LayerByName** and **LayerByIndex**. The number of layers in the collection object can be retrieved by the property **Count**. The methods **Add**, **Copy** and **Remove** allow to handle the layers in the corresponding way.

Properties

- _NewEnum
- Count

Methods

- Add
- AddBySpecification
- Copy
- FirstLayer
- LayerByIndex
- LayerByName
- NextLayer
- Remove
- Update

Properties

_NewEnum

Read Only Property of VcLayerCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all layer objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method

GetEnumerator is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim layer As VcBoxLayer

For Each layer In VcGantt1.LayerCollection
   Debug.Print layer.Name
Next
```

Count

Read Only Property of VcLayerCollection

This property lets you retrieve the number of layers in the layer collection.

	Data Type	Explanation
Property value	Long	Number of layers

Example Code

```
Dim numberOfLayers As Long

numberOfLayers = VcGantt1.LayerCollection.Count
```

Methods

Add

Method of VcLayerCollection

By this method you can create a layer as a member of the LayerCollection. If the name was not used before, the new layer object will be returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
Parameter:		
⇨ LayerName	String **Possible Values:**	Layer name Name of the color map

Return value	VcLayer	New layer object

Example Code

```
Set newLayer = VcGantt1.LayerCollection.Add("test1")
```

AddBySpecification

Method of VcLayerCollection

This method lets you create a layer by using a layer specification. This way of creating allows layer objects to become persistent. The specification of a layer can be saved and re-loaded (see VcLayer property **Specification**). In a subsequent session the layer can be created again from the specification and is identified by its name.

	Data Type	**Explanation**
Parameter:		
⇨ Specification	String **Possible Values:**	Layer specification Name of the color map
Return value	VcLayer	New layer object

Copy

Method of VcLayerCollection

By this method you can copy a layer. If the layer that is to be copied exists, and if the name for the new layer does not yet exist, the new layer object is returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	**Explanation**
Parameter:		
⇨ LayerName	String **Possible Values:**	Name of the layer to be copied Name of the color map
⇨ newLayerName	String **Possible Values:**	Name of the new layer Name of the color map
Return value	VcLayer	Layer object

FirstLayer

Method of VcLayerCollection

This method can be used to access the initial value, i.e. the first layer of a layer collection and then to continue in a forward iteration loop by the method **NextLayer** for the layers following. If there is no layer in the layer collection, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcLayer	First Layer

Example Code

```
Dim layerCltn As VcLayerCollection
Dim layer As VcLayer

Set layerCltn = VcGantt1.LayerCollection
Set layer = layerCltn.FirstLayer
```

LayerByIndex

Method of VcLayerCollection

This method lets you access a layer by its index. If a layer of the specified index does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter:		
⇨ index	Integer	Index of the layer
	Possible Values:	Data field index
Return value	VcLayer	Layer object returned

LayerByName

Method of VcLayerCollection

This method retrieves a layer by its name. If a layer of the specified name does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter:		
⇨ layerName	String	Name of layer

	Possible Values:	Name of the color map
Return value	VcLayer	Layer

Example Code

```
Dim layerCltn As VcLayerCollection
Dim layer As VcLayer

Set layerCltn = VcGantt1.LayerCollection
Set layer = layerCltn.LayerByName("Start-End")
```

NextLayer

Method of VcLayerCollection

This method can be used in a forward iteration loop to retrieve subsequent layers from a layer collection after initializing the loop by the method **FirstLayer**. If there is no layer left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcLayer	Subsequent Layer

Example Code

```
Dim layerCltn As VcLayerCollection
Dim layer As VcLayer

Set layerCltn = VcGantt1.LayerCollection
Set layer = layerCltn.FirstLayer
While Not layer Is Nothing
   list1.AddItem layer.Name
   Set layer = layerCltn.NextLayer
Wend
```

Remove

Method of VcLayerCollection

This method lets you delete a layer. If it is used in another object, it cannot be deleted. Then False will be returned, otherwise True.

	Data Type	Explanation
Parameter:		
⇨ LayerName	String	layer name
	Possible Values:	Name of the color map
Return value	Boolean	layer deleted (True)/not deleted (False)

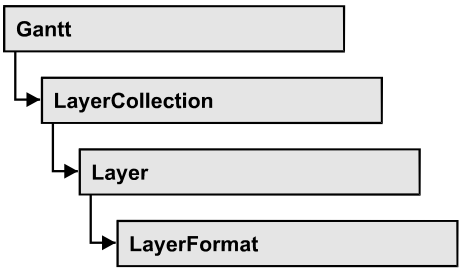
Update

Method of VcLayerCollection

This method lets you update a layer collection after having modified it.

	Data Type	Explanation
Return value	Boolean	update successful (True)/ not successful (False)

7.50 VcLayerFormat



A layer format specifies the annotation of layers.

Properties

- _NewEnum
- FormatField
- FormatFieldCount

Methods

- CopyFormatField
- RemoveFormatField

Properties

_NewEnum

Read Only Property of VcLayerFormat

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all layer format field objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method **GetEnumerator** is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim formatField As VcLayerFormatField

For Each formatField In format
    Debug.Print formatField.Index
```

Next

FormatField

Read Only Property of VcLayerFormat

This property gives access to a VcLayerFormatField object by its index. The index has to be in the range from 0 to FormatFieldCount-1.

Note for users of a version earlier than 3.0: The index does **not** range from 1 to FormatFieldCount as later versions do.

	Data Type	Explanation
Parameter:		
Index	Integer	Index of the layer format field
		0 ... .FormatFieldCount-1
	Possible Values:	Data field index
Property value	VcLayerFormatField	Layer format field

FormatFieldCount

Read Only Property of VcLayerFormat

This property gives access to the number of fields in a layer format.

	Data Type	Explanation
Property value	Integer	Number of fields of the layer format
	Possible Values:	Data field index

Methods

CopyFormatField

Method of VcLayerFormat

This method allows to copy a layer format field. The new VcLayerFormatField object is returned. It is given automatically the next index not used before.

	Data Type	Explanation
Parameter:		
⇨ position	FormatFieldPositionEnum **Possible Values:** vcAbove 1 vcBelow 3 vcLeftOf 0 vcOutsideAbove 9 vcOutsideBelow 11 vcOutsideLeftOf 8 vcOutsideRightOf 12 vcRightOf 4	Position of the new layer format field above below left of outside, above outside, below outside, left of outside, right of right of
⇨ refIndex	Integer **Possible Values:**	Index of the reference layer format field Data field index
Return value	VcLayerFormatField	Layer format field object

RemoveFormatField

Method of VcLayerFormat

This method lets you remove a layer format field by its index. After that, the program will update all layer format field indexes so that they are consecutively numbered again.

	Data Type	Explanation
Parameter:		
⇨ Index	Integer **Possible Values:**	index of the layer format field to be deleted Data field index

7.51 VcLayerFormatField

An object of the type VcLayerFormatField represents a field of a VcLayerFormat-Object. A layer format field does not have a name as many other objects, but it has an index that defines its position in the layer format.

Properties

- Alignment
- BottomMargin
- BottomMargin
- ConstantText
- FormatName
- Index
- LeftMargin
- LeftMargin
- MinimumWidth
- Priority
- RightMargin
- RightMargin
- SuppressTruncatedText
- TextDataFieldIndex
- TextFont
- TextFontColor
- TextFontColorDataFieldIndex
- TextFontColorMapName
- TextFontDataFieldIndex
- TextFontMapName
- TextLineCount
- TextLineCountDataFieldIndex
- TextLineCountMapName
- TopMargin
- TopMargin

Methods

- CalculateLineCount

Properties

Alignment

Property of VcLayerFormatField

This property lets you set or retrieve the alignment of the content of the layer format field.

	Data Type	Explanation
Property value	FormatFieldAlignmentEnum **Possible Values:** vcFFABottom 28 vcFFABottomLeft 27 vcFFABottomRight 29 vcFFACenter 25 vcFFALeft 24 vcFFARight 26 vcFFATop 22 vcFFATopLeft 21 vcFFATopRight 23	Alignment of the field content bottom bottom left bottom right center left right top top left top right

BottomMargin

Property of VcLayerFormatField

This property lets you set or retrieve the width of the bottom margin of the layer format field.

	Data Type	Explanation
Parameter: ⇨ Rückgabewert	Integer **Possible Values:**	Width of the bottom margin of the layer format field 0…9 Data field index
Property value	Integer **Possible Values:**	Width of the bottom margin of the layer format field 0…9 Data field index

BottomMargin

Property of VcLayerFormatField

This property lets you set or retrieve the width (in mm) of the bottom margin of the layer format field. It can also be set in the **Edit Layer Format** dialog box.

	Data Type	Explanation
Parameter: ⇨ Rückgabewert	Integer **Possible Values:**	Width of the bottom margin of the layer format field 0…9 Data field index
Property value	Integer **Possible Values:**	Width of the bottom margin of the layer format field 0…9 Data field index

ConstantText

Property of VcLayerFormatField

This property allows the layer format field to display a constant text, if the property **TextDataFieldIndex** was set to **-1**.

	Data Type	Explanation
Property value	String **Possible Values:**	Constant text Name of the color map

FormatName

Read Only Property of VcLayerFormatField

This property lets you retrieve the name of the layer format to which this layer format field belongs.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the layer format Name of the color map

Index

Read Only Property of VcLayerFormatField

This property lets you enquire the index of the layer format field in the corresponding layer format.

	Data Type	Explanation
Property value	Integer **Possible Values:**	Index of the layer format field Data field index

LeftMargin

Property of VcLayerFormatField

This property lets you set or retrieve the width of the left margin of the layer format field.

	Data Type	Explanation
Parameter: ⇨ Rückgabewert	Integer **Possible Values:**	Width of the left margin of the layer format field 0…9 Data field index
Property value	Integer **Possible Values:**	Width of the left margin of the layer format field 0…9 Data field index

LeftMargin

Property of VcLayerFormatField

This property lets you set or retrieve the width (in mm) of the left margin of the layer format field. It can also be set in the **Edit Layer Format** dialog box.

	Data Type	Explanation
Parameter: ⇨ Rückgabewert	Integer **Possible Values:**	Width of the left margin of the layer format field 0…9 Data field index
Property value	Integer **Possible Values:**	Width of the left margin of the layer format field 0…9 Data field index

MinimumWidth

Property of VcLayerFormatField

This property lets you enquire or set the minimum width of the layer format field in mm if the label size dependence allows it.

	Data Type	Explanation
Property value	Integer **Possible Values:**	Minimum width of the layer format field in mm 0 ... 99 Data field index

Priority

Property of VcLayerFormatField

This property lets you specify or enquire the priority of the layer format field. By the priority you can influence the allocation of the available space in the field. The higher the priority, the greater the chance to get the space necessary.

	Data Type	Explanation
Property value	Integer **Possible Values:**	Priority of the layer format field Data field index

RightMargin

Property of VcLayerFormatField

This property lets you set or retrieve the width of the right margin of the layer format field.

	Data Type	Explanation
Parameter: ⇨ Rückgabewert	Integer **Possible Values:**	Width of the right margin of the layer format field 0…9 Data field index
Property value	Integer **Possible Values:**	Width of the right margin of the layer format field 0…9 Data field index

RightMargin

Property of VcLayerFormatField

This property lets you set or retrieve the width (in mm) of the right margin of the layer format field. It can also be set in the **Edit Layer Format** dialog box.

	Data Type	Explanation
Parameter: ⇨ Rückgabewert	Integer **Possible Values:**	Width of the right margin of the layer format field 0…9 Data field index

Property value	Integer **Possible Values:**	Width of the right margin of the layer format field 0…9 Data field index

SuppressTruncatedText

Property of VcLayerFormatField

This property lets you set or retrieve whether text which doesn't fit in the layer format field exactly is to be suppressed or cut.

	Data Type	**Explanation**
Property value	Boolean **Possible Values:**	Property active (True)/ not active (False) Group invisible/visible group nodes are/are not visible

TextDataFieldIndex

Property of VcLayerFormatField

This property lets you set or retrieve the index of the data field, the content of which is to be displayed in the layer format field. If the value of the index equals **-1**, the content of the property **ConstantText** will be returned instead.

	Data Type	**Explanation**
Property value	Integer **Possible Values:**	Index of the data field Data field index

TextFont

Property of VcLayerFormatField

This property lets you set or retrieve the font of the layer format field. If in the property **TextFontMapName** a map is specified, this map will control the text font color dependent on the data.

	Data Type	Explanation
Property value	StdFont	Font type of the layer format

TextFontColor

Property of VcLayerFormatField

This property lets you set or retrieve the font color of the layer format field, if it is of the type **vcFFTText**. If a map was set by the property **TextFontMap-Name**, the map will control the text font color in dependence of the data.

	Data Type	Explanation
Property value	OLE_COLOR	Font color of the layer format **Default value:** -1

TextFontColorDataFieldIndex

Property of VcLayerFormatField

This property lets you set or retrieve the data field index to be used with a font color map specified by the property **TextFontColorMapName**. If you set this property to **-1**, no map will be used.

	Data Type	Explanation
Property value	Integer **Possible Values:**	Data field index Data field index

TextFontColorMapName

Property of VcLayerFormatField

This property lets you set or retrieve the name of a color map (type vcColorMap) for the font color. If the name of the color map is set to "", no map will be used. If a map name and a data field index are specified by the property **TextFontColorDataFieldIndex**, the font color will be controlled by the map. If no map entry applies, the font color specified in the property **TextFontColor** will be used.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the font color map Name of the color map

TextFontDataFieldIndex

Property of VcLayerFormatField

This property lets you set or retrieve the data field index to be used together with a font map specified by the property **TextFontMapName**. If you set this property to **-1**, no map will be used.

	Data Type	Explanation
Property value	Integer **Possible Values:**	Data field index Data field index

TextFontMapName

Property of VcLayerFormatField

This property lets you set or retrieve the name of a font map (type vcFontMap). If set to "", no map will be used. If a map name and additionally a data field index is specified in the property **TextFontDataFieldIndex**, then the font is controlled by the map. If no data field entry applies, the font that is specified in the property **TextFont** will be used.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the font map Name of the color map

TextLineCount

Property of VcLayerFormatField

This property lets you enquire or set the line count, if the label size dependence allows it

	Data Type	Explanation
Property value	Integer	Number of lines
	Possible Values:	Data field index

TextLineCountDataFieldIndex

Property of VcLayerFormatField

This property lets you set or retrieve the data field index to be used together with a numeric map specified by the property **TextLineCountMapName**. If you set this property to **-1**, no map will be used.

	Data Type	Explanation
Property value	Integer	Data field index
	Possible Values:	Data field index

TextLineCountMapName

Property of VcLayerFormatField

This property lets you set or retrieve the name of a numeric map for the number of text lines. If the name of the map is set to "", no map will be used. If a map name and a data field index are specified by the property **TextLineCountDataFieldIndex**, the number of lines will be controlled by the map. If no map entry applies, the number of lines specified by the property **TextLineCount** will be used.

	Data Type	Explanation
Property value	String	Name of the numeric map
	Possible Values:	Name of the color map

TopMargin

Property of VcLayerFormatField

This property lets you set or retrieve the width of the top margin of the layer format field.

	Data Type	Explanation
Parameter: ⇨ Rückgabewert	Integer **Possible Values:**	Width of the top margin of the layer format field 0…9 Data field index
Property value	Integer **Possible Values:**	Width of the top margin of the layer format field 0…9 Data field index

TopMargin

Property of VcLayerFormatField

This property lets you set or retrieve the width (in mm) of the top margin of the layer format field. It can also be set in the **Edit Layer Format** dialog box.

	Data Type	Explanation
Parameter: ⇨ Rückgabewert	Integer **Possible Values:**	Width of the top margin of the layer format field 0…9 Data field index
Property value	Integer **Possible Values:**	Width of the top margin of the layer format field 0…9 Data field index

Methods

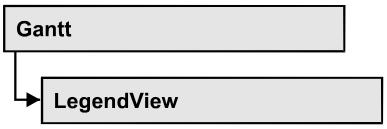
CalculateLineCount

Method of VcLayerFormatField

For external fields of a layer only: This method calculates the number of text lines in the layer format field of the designated node, considering the current sizes of the layer and of the font. If internal fields are passed, -1 will be returned. The result of the method can be stored to a data field of the node to control the number of lines displayed (See dialog **Edit layer format -> Line count**).

	Data Type	Explanation
Parameter: ⇨ node	VcNode	Node
Return value	Long	Number of text lines

7.52 VcLegendView



An object of the type **VcWorldView** designates the legend view window.

Properties

- Border
- BorderColor
- Height
- HeightActualValue
- Left
- LeftActualValue
- ParentHWnd
- ScrollBarMode
- Top
- TopActualValue
- Visible
- Width
- WidthActualValue
- WindowMode

Methods

- Update

Properties

Border

Property of VcLegendView

This property lets you set or retrieve whether the legend view has a frame (not in **vcPopupWindow** mode). he color of the frame is **Color.Black**. This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	Legend view with a border line (True)/without border line (False) **Default value:** True Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.LegendView.Mode = vcNotFixed
VcGantt1.LegendView.Border = True
```

BorderColor

Property of VcLegendView

This property lets you set/retrieve the color of the frame that may be visible.

	Data Type	Explanation
Property value	Color RGB ({0...255},{0...255},{0...255})	RGB color values ({0...255},{0...255},{0...255}) **Default value:** 0,0,0

Height

Property of VcLegendView

This property lets you retrieve the vertical extent of the legend view. In the modes **vcFixedAtTop**, **vcFixedAtBottom**, **vcNotFixed** and **vcPopupWindow** of the property **Mode** it can also be set.

Please note that the pixel coordinates are system coordinates, i. e. in Visual Basic you have to perform a conversion from/to Twips by the properties **App.TwipsPerPixelX** and **App.TwipsPerPixelY**.

This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	Long	Height of the legend view {0, ...} **Default value:** 100

Example Code

```
VcGantt1.LegendView.Height = 100
```

HeightActualValue

Read Only Property of VcLegendView

This property lets you retrieve the vertical extent of the legend view which actually is displayed. In the modes b!vcLVFixedAtBottom, **vcLVFixedAtLeft**, **vcLVFixedAtRight**, **vcLVFixedAtTop** the actual value may differ from the one that was set because in these modes either the height or the width is preset.

Please note that the pixel coordinates are system coordinates, i. e. in Visual Basic you have to perform a conversion from/in Twips by the properties **App.TwipsPerPixelX** and **App.TwipsPerPixelY**.

	Data Type	Explanation
Property value	Long	Actual height of the legend view {0, ...} **Default value:** 100

Example Code

```
VcGantt1.LegendView.HeightActualValue = 300
```

Left

Property of VcLegendView

This property lets you retrieve the left position of the legend view. In the modes **vcLVNotFixed** and **vcLVPopupWindow** of the property **Mode** it can also be set.

Please note that the pixel coordinates are system coordinates, i. e. in Visual Basic you have to perform a conversion from/to Twips by the properties **App.TwipsPerPixelX** and **App.TwipsPerPixelY**.

This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	Long	Left position of the legend view **Default value:** 0

Example Code

```
VcGantt1.LegendView.Left = 200
```

LeftActualValue

Read Only Property of VcLegendView

This property lets you retrieve the left position of the legend view which actually is displayed. In the modes b!vcLVFixedAtBottom, **vcLVFixedAtLeft**, **vcLVFixedAtRight**, **vcLVFixedAtTop** the actual value may differ from the one that was set because in these modes either height or width is preset.

Please note that the pixel coordinates are system coordinates, i. e. in Visual Basic you have to perform a conversion from/to Twips by the properties **App.TwipsPerPixelX** and **App.TwipsPerPixelY**.

	Data Type	Explanation
Property value	Long	Actual left position of the legend view **Default value:** 0

Example Code

```
VcGantt1.LegendView.LeftActualValue = 150
```

ParentHWnd

Property of VcLegendView

In the **vcLVNotFixed** mode, this property lets you set the HWnd handle of the parent window, for example, if the legend view is to appear in a frame window implemented by your own. By default, the frame window is positioned on the HWnd handle of the parent window of the VARCHART ActiveX main window. This property can be used only at run time.

	Data Type	Explanation
Property value	OLE_HANDLE	Handle

Example Code

```
MsgBox (VcGantt1.legendview.ParentHWnd)
```

ScrollBarMode

Property of VcLegendView

This property lets you set or retrieve the scroll bar mode of the legend view. This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	LegendViewScrollBarModeEnum	Scrollbarmode **Default value:** NoScrollBar
	Possible Values:	
	vcAutomaticScrollBar 3	Display of a horizontal or vertical scrollbar if required.
	vcHorizontalScrollBar 1	Display of a horizontal scrollbar if required.
	vcNoScrollBar 0	The complete chart is displayed without scrollbars.
	vcVerticalScrollBar 2	Display of a vertical scrollbar if required.

Example Code

```
VcGantt1.LegendView.ScrollBarMode = vcAutomaticScrollBar
```

Top

Property of VcLegendView

This property lets you retrieve the top position of the legend view. In the modes **vcNotFixed** und **vcPopupWindow** of the property **Mode** it also can be set.

Please note that the pixel coordinates are system coordinates, i. e. in Visual Basic you have to perform a conversion from/to Twips by the properties **App.TwipsPerPixelX** and **App.TwipsPerPixelY**.

This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	Long	Top position of the legend view **Default value:** 0

Example Code

```
VcGantt1.LegendView.Top = 20
```

TopActualValue

Read Only Property of VcLegendView

This property lets you retrieve the top position of the legend view which actually is displayed. In the modes b!vcLVFixedAtBottom, **vcLVFixedAtLeft**, **vcLVFixedAtRight**, **vcLVFixedAtTop** the actual value may differ from the one that was set because in these modes either the height or the width is preset.

Please note that the pixel coordinates are system coordinates, i. e. in Visual Basic you have to perform a conversion from/to Twips by the properties **App.TwipsPerPixelX** and **App.TwipsPerPixelY**.

	Data Type	Explanation
Property value	Long	Actual top position of the legend view **Default value:** 0

Example Code

```
VcGantt1.LegendView.TopActualValue = 40
```

Visible

Property of VcLegendView

This property lets you enquire/set whether the legend view is visible or not. This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	Legend view visible (True)/not visible (False) **Default value:** False Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.LegendView.Visible = True
```

Width

Property of VcLegendView

This property lets you retrieve the horizontal extent of the legend view. In the modes **vcFixedAtLeft**, **vcFixedAtRight**, **vcNotFixed** and **vcPopupWindow** of the property **Mode** it also can be set.

Please note that the pixel coordinates are system coordinates, i. e. in Visual Basic you have to perform a conversion from/to Twips by the properties **App.TwipsPerPixelX** and **App.TwipsPerPixelY**.

This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	Long	Horizontal extension of the legend view {0, ...} **Default value:** 100

Example Code

```
VcGantt1.LegendView.Width = 200
```

WidthActualValue

Read Only Property of VcLegendView

This property lets you retrieve the horizontal extent of the legend view which actually is displayed. In the mode b!vcLVFixedAtBottom, **vcLVFixedAtLeft**, **vcLVFixedAtRight**, **vcLVFixedAtTop** the actual value may differ from the one that was set because in these modes either the height or the width is preset. Please note that the pixel coordinates are system coordinates, i. e. in Visual Basic you have to perform a conversion from/to Twips by the properties **App.TwipsPerPixelX** and **App.TwipsPerPixelY**.

	Data Type	Explanation
Property value	Long	Actual horizontal extension of the legend view {0, ...} **Default value:** 100

Example Code

```
VcGantt1.LegendView.WidthActualValue = 600
```

WindowMode

Property of VcLegendView

This property lets you enquire/set the legend view mode. This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	LegendViewWindowModeEnum	Mode of the legend view **Default value:** vcPopupWindow
	Possible Values:	
	vcFixedAtBottom 4	The legend view is displayed on the bottom of the VARCHART ActiveX control window. Then the height can be specified, whereas the position and the width are fixed.
	vcFixedAtLeft 1	The legend view is displayed on the left side of the VARCHART ActiveX control window. Then the width can be specified, whereas the position and the height are fixed.
	vcFixedAtRight 2	The legend view is displayed on the right side of the VARCHART ActiveX control window. Then the width can be specified, whereas the position and the height are fixed.
	vcFixedAtTop 3	The legend view is displayed on the top of the VARCHART ActiveX control window. Then the height can be specified, whereas the position and the width are fixed.
	vcNotFixed 5	The legend view is a child window of the current parent window of the VARCHART ActiveX. It can be positioned at any position with any extension. The parent window can be modified via the property **VcWorldView.ParentHWnd**.
	vcPopupWindow 6	The legend view is a popup window with its own frame. The user can modify its position and extension, open it via the default context menu, and close it via the **Close** button in the frame.

Example Code

```
VcGantt1.LegendView.WindowMode = vcNotFixed
```

Methods

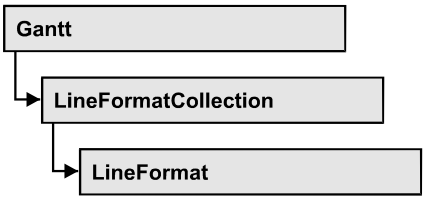
Update

Method of VcLegendView

This method lets you update the legend.

	Data Type	Explanation

7.53 VcLineFormat



An object of the type VcLineFormat defines the content and the appearance of lines, for example in a date line grid.

Properties

- _NewEnum
- FormatField
- FormatFieldCount
- Name
- Specification

Methods

- CopyFormatField
- RemoveFormatField

Properties

_NewEnum

Read Only Property of VcLineFormat

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all line format field objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method **GetEnumerator** is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation

Example Code

```
Dim formatField As VcLineFormatField
```

```
For Each formatField In format
    Debug.Print formatField.Index
Next
```

FormatField

Read Only Property of VcLineFormat

This property lets you retrieve a VcLineFormatField object by an index. The index has to be in the range from 0 to FormatFieldCount-1.

Note to users of versions previous to 3.0: The index does **not** count in the range from 1 to FormatFieldCount as in the versions up to 3.0.

	Data Type	Explanation
Parameter:		
index	Integer	Index of the line format field
		0 ... .FormatFieldCount-1
	Possible Values:	Data field index
Property value	VcNodeFormatField	Node format field

FormatFieldCount

Read Only Property of VcLineFormat

This property lets you retrieve the number of format fields of this line format.

	Data Type	Explanation
Property value	Integer	Number of fields of the line format
	Possible Values:	Data field index

Example Code

```
Dim format As VcLineFormat
Dim numberOfColumns As Integer

Set format = VcGantt1.Line.LineFormatCollection.FormatByName("StandardList")
numberOfColumns = FormatFieldCount
```

Name

Property of VcLineFormat

This property lets you set / retrieve the name of the line format.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the line format Name of the color map

Example Code

```
Dim format As VcLineFormat
Dim formatName As String

Set format = VcGantt1.Line.LineFormatCollection.FirstFormat
formatName = format.Name
```

Specification

Read Only Property of VcLineFormat

This property lets you retrieve the specification of a line format. A specification is a string that contains legible ASCII characters from 32 to 127 only, so it can be stored smoothly to text files or data bases. This allows for persistency. A specification can be used to create a node format by the method **VcNodeFormatCollection.AddBySpecification**.

	Data Type	Explanation
Property value	String **Possible Values:**	Specification of the line format Name of the color map

Methods

CopyFormatField

Method of VcLineFormat

This method copies a line format field, returning the new VcLineFormatField object. It contains the next consecutive unused index.

	Data Type	Explanation
Parameter:		
⇨ position	FormatFieldPositionEnum **Possible Values:** vcAbove 1 vcBelow 3 vcLeftOf 0 vcOutsideAbove 9 vcOutsideBelow 11 vcOutsideLeftOf 8 vcOutsideRightOf 12 vcRightOf 4	Position of the new line format field above below left of outside, above outside, below outside, left of outside, right of right of
⇨ refIndex	Integer **Possible Values:**	Index of the reference line format field Data field index
Return value	VcLineFormatField	Line format field object

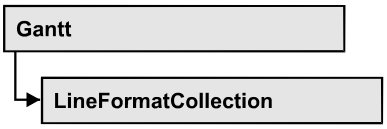
RemoveFormatField

Method of VcLineFormat

This method lets you remove a layer format field by its index. After that, the program will update all layer format field indexes so that they are consecutively numbered again.

	Data Type	Explanation
Parameter:		
⇨ index	Integer **Possible Values:**	Index of the line format field to be deleted Data field index
Return value	Void	

7.54 VcLineFormatCollection



An object of the type VcLineFormatCollection automatically contains all line formats available to lines. You can access all objects in an iterative loop by **For Each lineFormat In LineFormatCollection** or by the methods **First...** and **Next...**. You can access a single line format by the methods **FormatBy-Name** and **FormatByIndex**. The number of lines in the collection object can be retrieved by the property **Count**. The methods **Add**, **Copy** and **Remove** allow to handle the line formats in the corresponding way.

Properties

- _NewEnum
- Count

Methods

- Add
- AddBySpecification
- Copy
- FirstFormat
- FormatByIndex
- FormatByName
- NextFormat
- Remove

Properties

_NewEnum

Read Only Property of VcLineFormatCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all line format objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method **GetEnumerator** is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation

Example Code

```
Dim format As VcLineFormat

For Each format In VcGantt1.LineFormatCollection
   Debug.Print format.Name
Next
```

Count

Read Only Property of VcLineFormatCollection

This property lets you retrieve the number of line formats in the line format collection.

	Data Type	Explanation
Property value	Long	Number of line formats

Example Code

```
Dim lineFormatCltn As VcLineFormatCollection
Dim numberOfLineformats As Long

Set lineFormatCltn = VcGantt1.LineFormatCollection
Dim numberOfLineformats = lineFormatCltn.Count
```

Methods

Add

Method of VcLineFormatCollection

By this method you can create a line format as a member of the LineFormat-Collection. If the name was not used before, the new box object will be returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
Parameter:		
⇨ FormatName	String	Name of the line format
	Possible Values:	Name of the color map
Return value	VcLineFormat	New line format object

Example Code

```
Set newLineFormat = VcGantt1.LineFormatCollection.Add("boxFormat1")
```

AddBySpecification

Method of VcLineFormatCollection

This method lets you create a line format by using a box format specification. This way of creating allows line format objects to become persistent. The specification of a line format can be saved and re-loaded (see VcLineFormat property **Specification**). In a subsequent session the line format can be created again from the specification and is identified by its name.

	Data Type	Explanation
Parameter:		
⇨ formatSpecification	String **Possible Values:**	Line format specification Name of the color map
Return value	VcLineFormat	New line format object

Copy

Method of VcLineFormatCollection

By this method you can copy a line format. If the line format that is to be copied exists, and if the name for the new line format does not yet exist, the new line format object is returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
Parameter:		
⇨ FormatName	String **Possible Values:**	Name of the line format to be copied Name of the color map
⇨ newFormatName	String **Possible Values:**	Name of the new line format Name of the color map
Return value	VcLineFormat	Line format object

Example Code

```
Dim lineFormatCltn As VcLineFormatCollection
```

```
Dim lineFormat As VcLineFormat

Set lineFormatCltn = VcGantt1.LineFormatCollection
Set lineFormat = lineFormatCltn.Copy("CurrentLineFormat", "NewLineFormat")
```

FirstFormat

Method of VcLineFormatCollection

This method can be used to access the initial value, i.e. the first line format of a line format collection and then to continue in a forward iteration loop by the method **NextFormat** for the line formats following. If there is no line format in the line format collection, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcLineFormat	First line format

Example Code

```
Dim format As VcLineFormat

Set format = VcGantt1.LineFormatCollection.FirstFormat
```

FormatByIndex

Method of VcLineFormatCollection

This method lets you access a line format by its index. If a line format of the specified index does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter: ⇨ index	Integer **Possible Values:**	Index of the linde format Data field index
Return value	VcLineFormat	Line format object returned

Example Code

```
Dim lineFormatCltn As VcLineFormatCollection
Dim format As VcLineFormat

Set lineFormatCltn = VcGantt1.LineFormatCollection
Set format = lineFormatCltn.FormatByIndex(2)
```

FormatByName

Method of VcLineFormatCollection

By this method you can retrieve a line format by its name. If a line format of the specified name does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter:		
⇨ formatName	String	Name of the line format
	Possible Values:	Name of the color map
Return value	VcLineFormat	Line format

Example Code

```
Dim formatCltn As VcLineFormatCollection
Dim format As VcLineFormat

Set formatCltn = VcGantt1.LineFormatCollection
Set format = formatCltn.FormatByName("Standard")
```

NextFormat

Method of VcLineFormatCollection

This method can be used in a forward iteration loop to retrieve subsequent line formats from a line format collection after initializing the loop by the method **FirstFormat**. If there is no format left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcLineFormat	Subsequent line format

Example Code

```
Dim formatCltn As VcLineFormatCollection
Dim format As VcLineFormat

Set formatCltn = VcGantt1.LineFormatCollection
Set format = formatCltn.FirstFormat

While Not format Is Nothing
     List1.AddItem format.Name
     Set format = formatCltn.NextFormat
Wend
```

Remove

Method of VcLineFormatCollection

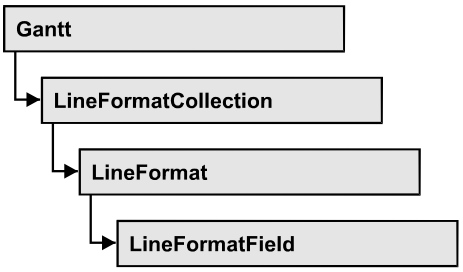
This method lets you delete a line format. If the line format is still used by another object, it cannot be deleted. Then **False** will be returned, otherwise **True**.

	Data Type	Explanation
Parameter:		
⇨ FormatName	String	Line format name
	Possible Values:	Name of the color map
Return value	Boolean	Line format deleted (True) / not deleted (False)

Example Code

```
Dim lineFormatCltn As VcLineFormatCollection
Dim lineFormat As VcLineFormat

Set lineFormatCltn = VcGantt1.LineFormatCollection
Set lineFormat = lineFormatCltn.FormatByIndex(1)
lineFormatCltn.Remove (lineFormat.Name)
```

7.55 VcLineFormatField



An object of the type **VcLineFormat** represents a field of a VcLineFormat object. A line format field does not have a name as many other objects do, but it has an index that defines its position in the line format.

Properties

- Alignment
- ConstantText
- DateOutputFormat
- FormatName
- Index
- PatternBackgroundColorAsARGB
- PatternBackgroundColorDataFieldIndex
- PatternBackgroundColorMapName
- PatternColorAsARGB
- PatternColorDataFieldIndex
- PatternColorMapName
- PatternEx
- PatternExDataFieldIndex
- PatternExMapName
- TextDataFieldIndex
- TextFont
- TextFontColor
- TextFontColorDataFieldIndex
- TextFontColorMapName
- TextFontDataFieldIndex
- TextFontMapName
- TextLineCount

Properties

Alignment

Property of VcLineFormatField

This property lets you set or retrieve the alignment of the content of the line format field.

	Data Type	Explanation
Property value	FormatFieldAlignmentEnum	Alignment of the field content
	Possible Values:	
	vcFFABottom 28	bottom
	vcFFABottomLeft 27	bottom left
	vcFFABottomRight 29	bottom right
	vcFFACenter 25	center
	vcFFALeft 24	left
	vcFFARight 26	right
	vcFFATop 22	top
	vcFFATopLeft 21	top left
	vcFFATopRight 23	top right

ConstantText

Property of VcLineFormatField

This property allows the line format field to display a constant text, if the line format field is of the type **vcFFTText** and if the property **TextDataField-Index** was set to -1.

	Data Type	Explanation
Property value	String	Constant text
	Possible Values:	Name of the color map

DateOutputFormat

Property of VcLineFormatField

This property lets you specify or enquire the date output format. To compose the date you can use the below codes:

D: first letter of the day of the week (not adjustable)

TD:	Day of the Week (adjustable by using the event **OnSupplyTextEntry**)
DD:	two-digit figure for the day of the month: 01-31
DDD:	three initial characters of the day of the week (not adjustable)
M:	first character of the name of the month (not adjustable)
TM:	name of the month (adjustable by using the event **OnSupplyTextEntry**)
MM:	two-digit figure for the month: 01-12
MMM:	three initial characters of the name of the month (not adjustable)
YY:	two-digit figure for the year
YYYY:	four-digit figure for the year
WW:	two-digit figure for the number of the calendar week: 01-53
TW:	text for "calendar week" (adjustable by using the event **OnSupplyTextEntry**)
Q:	one-digit figure for the quarter: 1-4
TQ:	name of quarter (adjustable by using the event **OnSupplyTextEntry**)
hh:	two-digit figure for the hour in 24 hours format: 00-23
HH:	two-digit figure for the hour in 12 hours format: 01-12
Th:	Text of "o' clock" (adjustable by using the event **OnSupplyTextEntry**)
TH:	"am" or "pm" (adjustable by using the event **OnSupplyTextEntry**)
mm	two-digit figure for the minute: 00-59
ss:	two-digit figure for the second: 00-59
TS:	short date format, as defined in the regional settings of the windows control panel
TL:	long date format, as defined in the regional settings of the windows control panel
TT:	time format, as defined in the regional settings of the windows control panel

Note: Characters which are not to be interpreted as part of the date should be preceded by a backslash '\'. '\\' for instance results in "\'. The special characters: '**:, /, -**' and **blank** don't need '\' as prefix.

	Data Type	Explanation
Property value	String	Date
		{DMYhms:;/}
	Possible Values:	Name of the color map

Example Code

```
VcLineFormatField.DateOutputFormat = "DD.MM.YY"
```

FormatName

Read Only Property of VcLineFormatField

This property lets you retrieve the name of the line format to which this line format field belongs.

	Data Type	Explanation
Property value	String	Name of the line format object
	Possible Values:	Name of the color map

Index

Read Only Property of VcLineFormatField

This property lets you retrieve the index of the line format field in the corresponding line format.

	Data Type	Explanation
Property value	Integer	Index of the table format field
	Possible Values:	Data field index

PatternBackgroundColorAsARGB

Property of VcLineFormatField

This property lets you set or retrieve the background color of the line format field. Color values have a transparency or alpha value, followed by a value for a red, a blue and a green partition (ARGB). The values range between 0..255. An alpha value of 0 equals complete transparency, whereas 255 represents a completely solid color. When casting an RGB value on an ARGB value, an alpha value of 255 has to be added.

If the line format field shall have the color of the line format, select the value -1.

If by the property **BackColorMapName** a map is specified, the map will set the background color of the line format field in dependence of data.

	Data Type	Explanation
Property value	Long	Background color of the table format **Default value:** -1

PatternBackgroundColorDataFieldIndex

Property of VcLineFormatField

This property lets you set or retrieve the data field index to be used with a color map specified by the property **PatternBackgroundColorMapName**. If you set this property to -1, no map will be used.

	Data Type	Explanation
Property value	Long	Data field index

PatternBackgroundColorMapName

Property of VcLineFormatField

This property lets you set or retrieve the name of a color map (type vcColor-Map). If set to "", no map will be used. If a map name and additionally a data field index is specified in the property **PatternBackgroundColorDataField-Index**, then the background color is controlled by the map. If no data field entry applies, the background color that is specified in the property **Back-Color** will be used.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the color map Name of the color map

PatternColorAsARGB

Property of VcLineFormatField

This property lets you set or retrieve the pattern color of the line format field. Color values have a transparency or alpha value, followed by a value for a red, a blue and a green partition (ARGB). The values range between 0..255. An alpha value of 0 equals complete transparency, whereas 255 represents a completely solid color. When casting an RGB value on an ARGB value, an alpha value of 255 has to be added.

If the box format field shall have the background color of the box format, select the value **-1**.

	Data Type	Explanation
Property value	Long	Pattern color of the line format field

Example Code

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

Set boxFormatCltn = VcGantt1.BoxFormatCollection
Set boxFormatField = boxFormatCltn.FirstFormat.formatField(0)
boxFormatField.PatternColor = RGB(0, 255, 0)
```

PatternColorDataFieldIndex

Read Only Property of VcLineFormatField

This property lets you set or retrieve the data field index that has to be specified if the property **PatternColorMapName** is used. If you set this property to **-1**, no map will be used.

	Data Type	Explanation
Property value	Integer **Possible Values:**	Data field index Data field index

PatternColorMapName

Property of VcLineFormatField

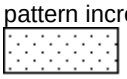
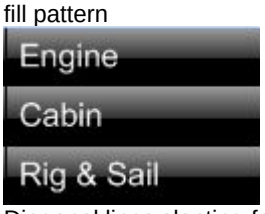
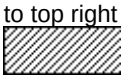
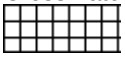
This property lets you set or retrieve the name of a color map (type vcColorMap). If set to "", no map will be used. Only if a map name and a data field index are specified in the property **PatternColorDataFieldIndex**, the pattern color is controlled by the map. If no data field entry applies, the pattern color of the calendar grid that is specified in the property **PatternColor** will be used.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the color map Name of the color map

PatternEx

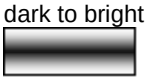
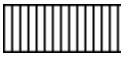
Property of VcLineFormatField

This property lets you set or retrieve the pattern of the field background of the line format field.

	Data Type	Explanation
Property value	FillPatternEnum	Pattern type **Default value:** As defined in the dialog
	Possible Values: vc05PercentPattern... vc90PercentPattern 01 - 11	Dots in foreground color on background color, the density of the foreground pattern increasing with the percentage 
	vcAeroGlassPattern 40	Vertical color gradient in the color of the fill pattern 
	vcBDiagonalPattern 5	Diagonal lines slanting from bottom left to top right 
	vcCrossPattern 6	Cross-hatch pattern 

vcDarkDownwardDiagonalPattern 2014	Diagonal lines slanting from top left to bottom right; spaced 50% closer than vcFDiagonalPattern and of twice the line width
vcDarkHorizontalPattern 2023	Horizontal lines spaced 50% closer than vcHorizontalPattern and of twice the line width
vcDarkUpwardDiagonalPattern 2015	Diagonal lines slanting from bottom left to top right, spaced 50% closer than vcBDiagonalPattern and of twice the line width
vcDarkVerticalPattern 2022	Vertical lines spaced 50% closer than vcVerticalPattern and of of twice the line width
vcDashedDownwardDiagonalPattern 2024	Dashed diagonal lines from top left to bottom right
vcDashedHorizontalPattern 2026	Dashed horizontal lines
vcDashedUpwardDiagonalPattern 2025	Dashed diagonal lines from bottom left to top right
vcDashedVerticalPattern 2027	Dashed vertical lines
vcDiagCrossPattern 7	Diagonal cross-hatch pattern, small
vcDiagonalBrickPattern 2032	Diagonal brick pattern
vcDivotPattern 2036	Divot pattern
vcDottedDiamondPattern 2038	Diagonal cross-hatch pattern of dotted lines
vcDottedGridPattern 2037	Cross-hatch pattern of dotted lines
vcFDiagonalPattern 4	Diagonal lines slanting from top left to bottom right
vcHorizontalBrickPattern 2033	Horizontal brick pattern
vcHorizontalGradientPattern 52	Horizontal color gradient
vcHorizontalPattern 3	Horizontal lines

vcLargeCheckerboardPattern 2044	Checkerboard pattern showing squares of twice the size of vcSmallChecker-BoardPattern
vcLargeConfettiPattern 2029	Confetti pattern, large
vcLightDownwardDiagonalPattern 2012	Diagonal lines slanting to from top left to bottom right; spaced 50% closer than vcBDiagonalPattern
vcLightHorizontalPattern 2019	Horizontal lines spaced 50% closer than vcHorizontalPattern
vcLightUpwardDiagonalPattern 2013	Diagonal lines slanting from bottom left to top right, spaced 50% closer than vcBDiagonalPattern
vcLightVerticalPattern 2018	Vertical lines spaced 50% closer than vcVerticalPattern
vcNarrowHorizontalPattern 2021	Horizontal lines spaced 75 % closer than vcHorizontalPattern
vcNarrowVerticalPattern 2020	Vertical lines spaced 75% closer than vcVerticalPattern
vcNoPattern 1276	No fill pattern
vcOutlinedDiamondPattern 2045	Diagonal cross-hatch pattern, large
vcPlaidPattern 2035	Plaid pattern
vcShinglePattern 2039	Diagonal shingle pattern
vcSmallCheckerBoardPattern 2043	Checkerboard pattern
vcSmallConfettiPattern 2028	Confetti pattern
vcSmallGridPattern 2042	Cross-hatch pattern spaced 50% closer than vcCrossPattern
vcSolidDiamondPattern 2046	Checkerboard pattern showing diagonal squares
vcSpherePattern 2041	Checkerboard of spheres
vcTrellisPattern 2040	Trellis pattern
vcVerticalBottomLightedConvexPattern 43	Vertical color gradient from dark to bright

vcVerticalConcavePattern 40	Vertical color gradient from dark to bright to dark 
vcVerticalConvexPattern 41	Vertical color gradient from bright to dark to bright 
vcVerticalGradientPattern 62	Vertical color gradient 
vcVerticalPattern 2	Vertical lines 
vcVerticalTopLightedConvexPattern 42	Vertical color gradient from bright to dark 
vcWavePattern 2031	Horizontal wave pattern 
vcWeavePattern 2034	Interwoven stripe pattern 
vcWideDownwardDiagonalPattern 2016	Diagonal lines slanting from top left to bottom right, showing the same spacing but three times the line width of vcF-DiagonalPattern 
vcWideUpwardDiagonalPattern 2017	Diagonal lines slanting from bottom left to top right right, showing the same spacing but three times the line width of vcBDiagonalPattern 
vcZigZagPattern 2030	Horizontal zig-zag lines 

PatternExDataFieldIndex

Property of VcLineFormatField

This property lets you set or retrieve the data field index to be used together with the property **PatternExMapName**. If you set this property to **-1**, no map will be used.

	Data Type	Explanation
Property value	Long	Data field index

PatternExMapName

Property of VcLineFormatField

This property lets you set or retrieve the name of a font map (type vcPatternMap). If set to "", no map will be used. If a map name and additionally a data field index is specified in the property **PatternExDataFieldIndex**, then the pattern is controlled by the map. If no data field entry applies, the pattern that is specified in the property **PatternEx** will be used.

	Data Type	Explanation
Property value	String	Name of the pattern map
	Possible Values:	Name of the color map

TextDataFieldIndex

Property of VcLineFormatField

only for the type ***vcFFTText***: This property lets you set or retrieve the index of the data field, the content of which is to be displayed in the line format field. If its value equals -1, the content of the property **ConstantText** will be returned.

	Data Type	Explanation
Property value	Integer	Index of the data field
	Possible Values:	Data field index

TextFont

Property of VcLineFormatField

This property lets you set or retrieve the font color of the line format field, if it is of the type **vcFFTText**. If a map was set by the property **TextFontMap-Name**, the map will control the text font in dependence of the data.

	Data Type	Explanation
Property value	StdFont	Font type of the table format

TextFontColor

Property of VcLineFormatField

This property lets you set or retrieve the font color of the line format field, if it is of the type **vcFFTText**. If a map was set by the property **TextFontMapName**, the map will control the text font color in dependence of the data.

	Data Type	Explanation
Property value	OLE_COLOR	Font color of the table format **Default value:** -1

TextFontColorDataFieldIndex

Property of VcLineFormatField

This property lets you set or retrieve the data field index to be used with a font color map specified by the property **TextFontColorMapName**. If you set this property to -1, no map will be used.

	Data Type	Explanation
Property value	Integer **Possible Values:**	Data field index Data field index

TextFontColorMapName

Property of VcLineFormatField

This property lets you set or retrieve the name of a color map (type vcColorMap) for the font color, if the format field is of the type **vcFFTText**. If the name of the color map is set to "", no map will be used. If a map name and a data field index are specified by the property **TextFontColorDataFieldIndex**, the font color will be controlled by the map. If no map entry applies, the font color specified in the property **TextFontColor** will be used.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the font color map Name of the color map

TextFontDataFieldIndex

Property of VcLineFormatField

This property lets you set or retrieve the data field index to be used with a font map specified by the property **TextFontMapName**. If you set this property to -1, no map will be used.

	Data Type	Explanation
Property value	Integer	Data field index
	Possible Values:	Data field index

TextFontMapName

Property of VcLineFormatField

This property lets you set or retrieve the name of a font map (type vcFontMap). If set to "", no map will be used. If a map name and additionally a data field index is specified in the property **TextFontDataFieldIndex**, then the font is controlled by the map. If no data field entry applies, the font that is specified in the property **TextFont** will be used.

	Data Type	Explanation
Property value	String	Name of the font map
	Possible Values:	Name of the color map

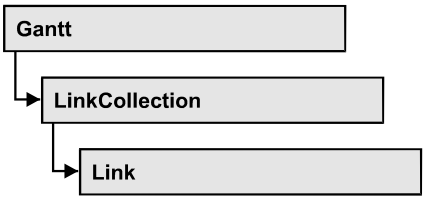
TextLineCount

Property of VcLineFormatField

This property lets you set or retrieve the number of lines, if the size of the annotation field allows for more than one line.

	Data Type	Explanation
Property value	Integer	Number of lines
	Possible Values:	Data field index

7.56 VcLink



A VcLink object represents the logical and graphical link between two nodes. On the **Link** property page you can specify via a tick box **Show links** whether links should be displayed. Even if they are not displayed, they will be used for scheduling.

Properties

- AllData
- DataField
- ID
- PredecessorNode
- SuccessorNode

Methods

- DataRecord
- DeleteLink
- RelatedDataRecord
- UpdateLink

Properties

AllData

Property of VcLink

This property lets you set or retrieve all data fields of a link. When setting the data, you can specify a CSV string (using semicolons as separators) or a data field. When retrieving the data, a character string will be returned. (See also **InsertLinkRecord**.)

	Data Type	Explanation
Property value	data field/string	All data of the link

Example Code

```
Dim linkCltn As VcLinkCollection
Dim link As VcLink
Dim allDataOfLink As String

Set linkCltn = VcGantt1.LinkCollection
Set link = linkCltn.FirstLink

allDataOfLink = link.AllData
```

DataField

Property of VcLink

This property lets you set or retrieve a specific data field of a link. The values which identify the predecessor and the successor nodes must not be changed.

	Data Type	Explanation
Parameter:		
⇨ index	Integer	Index of the data field
	Possible Values:	Data field index
Property value	Variant	Content of data field

Example Code

```
Dim linkCltn As VcLinkCollection
Dim link As VcLink
Dim message As String

Set linkCltn = VcGantt1.LinkCollection
For Each link in linkCltn
         message = "Delete link from " & link.DataField(1)
                         & " to " &  link.DataField(2) & " ?"
         If MsgBox(message, vbOKCancel, "Delete Link") = vbOK Then
              link.DeleteLink
         End If
Next link
```

ID

Read Only Property of VcLink

By this property you can retrieve the ID of a link.

	Data Type	Explanation
Property value	String	Link ID
	Possible Values:	Name of the color map

PredecessorNode

Read Only Property of VcLink

This method lets you identify the predecessor node of a link.

	Data Type	Explanation
Property value	VcNode	Predecessor node

Example Code

```
Dim linkCltn As VcLinkCollection
Dim link As VcLink
Dim node As VcNode
Dim nodeName As String

Set linkCltn = VcGantt1.LinkCollection
Set link = linkCltn.FirstLink
Set node = link.PredecessorNode
nodeName = node.DataField(1)
```

SuccessorNode

Read Only Property of VcLink

This method lets you identify the successor node of a link.

	Data Type	Explanation
Property value	VcNode	Successor node

Example Code

```
Dim linkCltn As VcLinkCollection
Dim link As VcLink
Dim node As VcNode
Dim nodeName As String

Set linkCltn = VcGantt1.LinkCollection
Set link = linkCltn.FirstLink
Set node = link.SuccessorNode
nodeName = node.DataField(1)
```

Methods

DataRecord

Method of VcLink

This property lets you retrieve the link as a data record object. The properties of the data record object give access to the corresponding data table and the data table collection.

	Data Type	Explanation
Return value	VcDataRecord	Data record returned

DeleteLink

Method of VcLink

By this method you can delete a link.

	Data Type	Explanation
Return value	Boolean	Link was (True) / was not (False) successfully deleted

Example Code

```
Private Sub VcGantt1_OnLinkRClick(ByVal link As VcGanttLib.VcLink, _
                                  ByVal x As Long, ByVal y As Long, _
                                  returnStatus As Variant)

   Dim message As String

   message = "Delete Link: " & link.AllData

   If MsgBox(message, vbOKCancel, "Delete Link") = vbOK Then
       link.DeleteLink
   End If

   returnStatus = vcRetStatNoPopup

End Sub
```

RelatedDataRecord

Method of VcLink

This property lets you retrieve a data record from a data table that is related to the link data table. The index passed by the parameter denotes the field in the data record that holds the key of the related data record.

	Data Type	Explanation
Parameter:		
⇨ index	Integer **Possible Values:**	Index of data field that holds the key Data field index
Return value	VcDataRecord	Related data record returned

UpdateLink

Method of VcLink

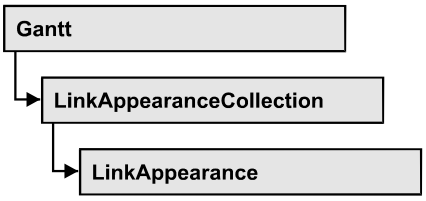
When a data field of a link was edited by the **DataField** property, you can update the diagram by the **UpdateLink** method.

	Data Type	Explanation
Return value	Boolean	Link was (True) / was not (False) updated successfully

Example Code

```
Dim linkCltn As VcLinkCollection
Dim link As VcLink

Set linkCltn = VcGantt1.LinkCollection
Set link = linkCltn.FirstLink

link.DataField(2) = "10"
link.UpdateLink
```

7.57 VcLinkAppearance



A VcLinkAppearance object defines the appearance of a link, if the node data comply with the conditions defined by the filters assigned. Different link appearances can be set on the **Link** property page in the table.

Properties

- FilterName
- LineColor
- LineThickness
- LineType
- Name
- PredecessorLayerName
- PrePortSymbol
- RoutingType
- Specification
- SuccessorLayerName
- SuccPortSymbol
- Visible

Methods

- PutInOrderAfter

Properties

FilterName

Read Only Property of VcLinkAppearance

This property lets you enquire the filter that is used for a specific link appearance. This property also can be set on the **Link** property page.

	Data Type	Explanation
Property value	VcFilter	Filter object

Example Code

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance
Dim filterOfLinkApp As String

Set linkAppearanceCltn = VcGantt1.LinkAppearanceCollection
Set linkAppearance = linkAppearanceCltn.LinkAppearanceByName("Blue")
filterOfLinkApp = linkAppearance.Filter
```

LineColor

Property of VcLinkAppearance

This property lets you set or retrieve the line color of a LinkAppearance object. This property can also be set on the **Link** property page in the **Line attributes** dialog.

	Data Type	Explanation
Property value	Color	RGB color values ({0...255},{0...255},{0...255})

Example Code

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance

Set linkAppearanceCltn = VcGantt1.LinkAppearanceCollection
Set linkAppearance = linkAppearanceCltn.LinkAppearanceByName("Blue")

linkAppearance.LineColor = RGB(0, 0, 255)
```

LineThickness

Property of VcLinkAppearance

This property lets you set or retrieve the line thickness of a LinkAppearance object.

If you set this property to values between 1 and 4, an absolute line thickness is defined in pixels. Irrespective of the zoom factor a line will always show the same line thickness in pixels. When printing though, the line thickness is adapted for the sake of legibility and becomes dependent of the zoom factor:

Value	Points	mm
1	1/2 point	0.09 mm

Value	Points	mm
2	1 point	0.18 mm
3	3/2 points	0.26 mm
4	2 points	0.35 mm

A point equals 1/72 inch and represents the unit of the font size.

If you set this property to values between 5 and 1,000, the line thickness is defined in 1/100 mm, so the lines will be displayed in a true thickness in pixels that depends on the zoom factor.

	Data Type	Explanation
Property value	Long	Line thickness LineType {1...4}: line thickness in pixels LineType {5...1000}: line thickness in 1/100 mm **Default value:** As defined on property page

Example Code

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance

Set linkAppearanceCltn = VcGantt1.LinkAppearanceCollection
Set linkAppearance = linkAppearanceCltn.LinkAppearanceByName("Standard")

linkAppearance.LineThickness = 4
```

LineType

Property of VcLinkAppearance

This property lets you set or retrieve the line type of a LinkAppearance object. This property can also be set in the **Line Attributes** dialog box that can be invoked by the **Link** property page.

	Data Type	Explanation
Property value	LineTypeEnum	Line type **Default value:** vcSolid
	Possible Values:	
	vcDashed 4	Line dashed
	vcDashedDotted 5	Line dashed-dotted
	vcDotted 3	Line dotted
	vcLineType0 100	Line Type 0
	vcLineType1 101	Line Type 1

vcLineType10 110	Line Type 10
vcLineType11 111	Line Type 11
vcLineType12 112	Line Type 12
vcLineType13 113	Line Type 13
vcLineType14 114	Line Type 14
vcLineType15 115	Line Type 15
vcLineType16 116	Line Type 16
vcLineType17 117	Line Type 17
vcLineType18 118	Line Type 18
vcLineType2 102	Line Type 2
vcLineType3 103	Line Type 3
vcLineType4 104	Line Type 4
vcLineType5 105	Line Type 5
vcLineType6 106	Line Type 6
vcLineType7 107	Line Type 7
vcLineType8 108	Line Type 8
vcLineType9 109	Line Type 9
vcNone 1	No line type
vcNotSet -1	No line type assigned
vcSolid 2	Line solid

Example Code

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance

Set linkAppearanceCltn = VcGantt1.LinkAppearanceCollection
Set linkAppearance = linkAppearanceCltn.LinkAppearanceByName("Blue")

linkAppearance.LineType = 5
```

Name

Read Only Property of VcLinkAppearance

This property lets you retrieve the name of a LinkAppearance object.

	Data Type	Explanation
Property value	String **Possible Values:**	Name

	Name of the color map

Example Code

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance
Dim nameLinkApp As String

Set linkAppearanceCltn = VcGantt1.LinkAppearanceCollection
Set linkAppearance = linkAppearanceCltn.FirstLinkAppearance

nameLinkApp  = linkAppearance.name
```

PredecessorLayerName

Property of VcLinkAppearance

This property lets you specify or retrieve to which layer of the predecessor node a link is to be drawn. If you enter "" (default), the link will be drawn to the first visible layer of this node.

This property can also be set on the **Links** property page.

	Data Type	Explanation
Property value	String **Possible Values:**	Character string that passes the layer name Name of the color map

PrePortSymbol

Property of VcLinkAppearance

This property lets you assign/retrieve a port symbol to/from a link, that visually accentuates the junction of the link and the predecessor node.

This property can also be set on the **Links** property page.

	Data Type	Explanation
Property value	LinkPredecessorSymbolEnum **Possible Values:** vcLPSArrow 64 vcLPSDoubleArrow 65	Symbol on the predecessor node **Default value:** vcLPSNone Predecessor port symbol **arrow** Predecessor port symbol **double arrow**

vcLPSDoubleSemiCircle 97	Predecessor port symbol **double semi-circle**
vcLPSFilledArrow 72	Predecessor port symbol **filled Arrow**
vcLPSFilledDoubleArrow 88	Predecessor port symbol **filled double arrow**
vcLPSFilledDoubleSemiCircle 120	Predecessor port symbol **filled double semi-circle**
vcLPSFilledSemiCircle 104	Predecessor port symbol **filled semi-circle**
vcLPSNone 0	Predecessor port symbol **none**
vcLPSSemiCircle 96	Predecessor port symbol **semi-circle**

Example Code

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance
Dim nameLinkApp As String

Set linkAppearanceCltn = VcGantt1.LinkAppearanceCollection
Set linkAppearance = linkAppearanceCltn.FirstLinkAppearance

linkAppearance.PrePortSymbol = vcLPSFilledDoubleSemiCircle
```

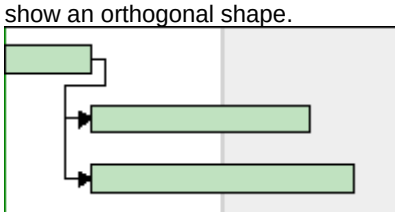
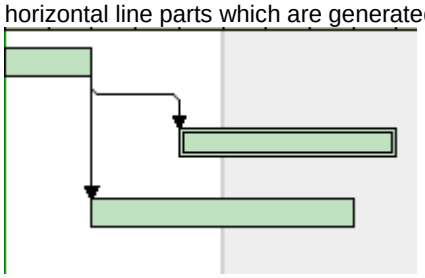
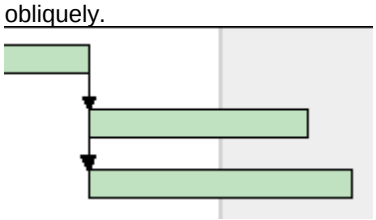
RoutingType

Property of VcLinkAppearance

This property lets you set or retrieve, whether the links of the diagram should be drawn horizontally and vertically only (and therefore show orthogonal shapes), or if they are allowed to lead directly to their aim, probably on an oblique route, allowing to cut through objects.

This property can also be set on the **Links** property page.

	Data Type	Explanation
Property value	LinkRoutingTypeEnum	Routing type **Default value:** vcLRTOrthogonal
	Possible Values: vcLRTNotSet -1	A routing type is used which is further up the list of the LinkAppearance objects.

vcLRTOrthogonal 1	Links run horizontally and vertically only and show an orthogonal shape. 
vcLRTOrthogonalDistinguishable 2	Links run horizontally and vertically only and show an orthogonal shape. By displaying bends with appropriate slants, the links will be better distinguished and their direction more easily tracked. The height of the chart will be increased according to the number of horizontal line parts which are generated. 
vcLRTStraightLined 4	Links have a straight shape and may run obliquely. 

Specification

Read Only Property of VcLinkAppearance

This property lets you retrieve the specification of a link appearance. A specification is a string that contains legible ASCII characters from 32 to 127 only, so it can be stored without problems to text files or data bases. This allows for persistency. A specification can be used to create a link appearance by the method **VcLinkAppearanceCollection.AddBySpecification**.

	Data Type	Explanation
Property value	String **Possible Values:**	Specification of the link appearance Name of the color map

SuccessorLayerName

Property of VcLinkAppearance

This property lets you specify or retrieve to which one of the layers of the successor node a link is to be drawn. If you set "" (default), the link will be drawn to the first visible layer of the node.

This property can also be set on the **Links** property page.

	Data Type	Explanation
Property value	String **Possible Values:**	Character string that passes the layer name Name of the color map

SuccPortSymbol

Property of VcLinkAppearance

This property lets you assign/retrieve a port symbol to a link, that visually accentuates the intersection of the link and the successor node.

This property can also be set on the **Links** property page.

	Data Type	Explanation
Property value	LinkSuccessorSymbolEnum	Symbol on the succesor node **Default value:** vcLSSNone
	Possible Values:	
	vcLSSArrow 32	Successor port symbol **arrow** →
	vcLSSDoubleArrow 33	Successor port symbol **double arrow** ↠
	vcLSSFilledArrow 40	Successor port symbol **filled arrow** ➔
	vcLSSFilledDoubleArrow 56	Successor port symbol **filled double arrow** ➠
	vcLSSNone 0	Successor port symbol **none**

Example Code

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance
Dim nameLinkApp As String

Set linkAppearanceCltn = VcGantt1.LinkAppearanceCollection
Set linkAppearance = linkAppearanceCltn.FirstLinkAppearance

linkAppearance.SuccPortSymbol = vcLSSFilledDoubleArrow
```

Visible

Property of VcLinkAppearance

This property lets you set or retrieve whether the link is to be visible or not, taking no effect, however, on the phantom lines for links while dragging.

This property can also be set on the **Links** property page, but here also applying to the phantom lines.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	Property active/not active **Default value:** True Group invisible/visible group nodes are/are not visible

Example Code

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance
Dim nameLinkApp As String

Set linkAppearanceCltn = VcGantt1.LinkAppearanceCollection
Set linkAppearance = linkAppearanceCltn.FirstLinkAppearance

linkAppearance.Visible = False
```

Methods

PutInOrderAfter

Method of VcLinkAppearance

This method lets you set the link appearance behind a link appearance specified by name, within the LinkAppearanceCollection. If you set the name to "", the link appearence will be put in the first position. The order of the link appearances within the collection determines the order by which they apply to the links.

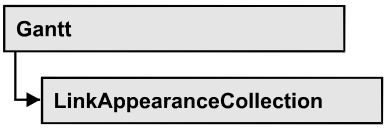
	Data Type	Explanation
Parameter:		
refLinkAppearanceName	String **Possible Values:**	Name of the link appearance behind which the current link appearance is to be put. Name of the color map

Return value	Void	

Example Code

```
Dim linkAppCltn As VcLinkAppearanceCollection
Dim linkApp1 As VcLinkAppearance
Dim linkApp2 As VcLinkAppearance

linkAppCltn = VcGantt1.LinkAppearanceCollection()
linkApp1 = linkAppCltn.Add("linkApp1")
linkApp2 = linkAppCltn.Add("linkApp2")
linkApp1.PutInOrderAfter("linkApp2")
linkAppCltn.Update()
```

7.58 VcLinkAppearanceCollection



An object of the type VcLinkAppearanceCollection automatically contains all available link appearances. You can access all objects in an iterative loop by **For Each linkAppearance In LinkAppearanceCollection** or by the methods **First...** and **Next...**. You can access a single line format by the methods **LinkAppearanceByName** and **LinkAppearandeByIndex**. The number of link appearances in the collection object can be retrieved by the property **Count**.

Properties

- _NewEnum
- Count

Methods

- Add
- AddBySpecification
- Copy
- FirstLinkAppearance
- LinkAppearanceByIndex
- LinkAppearanceByName
- NextLinkAppearance
- Remove
- Update

Properties

_NewEnum

Read Only Property of VcLinkAppearanceCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all link appearance objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method

GetEnumerator is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim linkApp As VcLinkAppearance

For Each linkApp In VcGantt1.LinkAppearanceCollection
   Debug.Print linkApp.Name
Next
```

Count

Read Only Property of VcLinkAppearanceCollection

This property lets you retrieve the number of link appearances in the LinkAppearanceCollection object.

	Data Type	Explanation
Property value	Long	Number of link appearance objects

Example Code

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance
Dim numberOfLinkAppearances As Integer

Set linkAppearanceCltn = VcGantt1.LinkAppearanceCollection

numberOfLinkAppearances = linkAppearanceCltn.Count
```

Methods

Add

Method of VcLinkAppearanceCollection

By this method you can create a new link appearance as a member of the LinkAppearanceCollection. If the name was not used before, the new linke appearance object will be returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned. All attributes of the new links appearance by default are set to transparent .

	Data Type	Explanation
Parameter:		
⇨ newName	String **Possible Values:**	Name of the link appearance Name of the color map
Return value	VcLinkAppearance	New linke appearance object

Example Code

```
Set newLinkAppearance = VcGantt1.LinkAppearanceCollection.Add("linkapp1")
```

AddBySpecification

Method of VcLinkAppearanceCollection

This method lets you create a link appearance by using a link appearance specification. This way of creating allows link appearance objects to become persistent. The specification of a link appearance can be saved and re-loaded (see VcLinkAppearance property **Specification**). In a later session the link appearance can be created again from the specification and is identified by its name.

	Data Type	Explanation
Parameter:		
⇨ linkAppearanceSpecification	String **Possible Values:**	Link appearance specification Name of the color map
Return value	VcLinkAppearance	New link appearance object

Copy

Method of VcLinkAppearanceCollection

By this method you can copy a link appearance. When the link appearance has come into existence and if the name for the new link appearance did not yet exist, the new link appearance object will be returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
Parameter:		
⇨ fromName	String **Possible Values:**	Name of the link appearance to be copied

		Name of the color map
⇨ newName	String **Possible Values:**	Name of the new link appearance Name of the color map
Return value	VcLinkAppearance	Link appearance object

FirstLinkAppearance

Method of VcLinkAppearanceCollection

This method can be used to access the initial value, i.e. the first link appearance of a link appearance collection and then to continue in a forward iteration loop by the method **NextLinkAppearance** for the link appearances following. If there is no link appearance in the link appearance collection, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcLinkAppearance	First linkAppearance object

Example Code

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance

Set linkAppearanceCltn = VcGantt1.LinkAppearanceCollection
Set linkAppearance = linkAppearanceCltn.FirstLinkAppearance
```

LinkAppearanceByIndex

Method of VcLinkAppearanceCollection

This method lets you access a link appearance object by its index. If a link appearance object of the specified index does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	Integer	Index of the link appearance object

LinkAppearanceByName

Method of VcLinkAppearanceCollection

This method retrieves a link appearance object by its name. If a link appearance object of the specified name does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter:		
⇨ linkAppearanceName	String	Name of the link appearance object
	Possible Values:	Name of the color map
Return value	VcLinkAppearance	LinkAppearance object

Example Code

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance

Set linkAppearanceCltn = VcGantt1.LinkAppearanceCollection
Set linkAppearance = linkAppearanceCltn.LinkAppearanceByName("Standard")
```

NextLinkAppearance

Method of VcLinkAppearanceCollection

This method can be used in a forward iteration loop to retrieve subsequent link appearances from a link appearance collection after initializing the loop by the method **FirstLinkAppearance**. If there is no link appearance left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcLinkAppearance	Subsequent linkAppearance object

Example Code

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance

Set linkAppearanceCltn = VcGantt1.LinkAppearanceCollection
Set linkAppearance = linkAppearanceCltn.FirstLinkAppearance

While Not linkAppearance Is Nothing
    linkAppearance.Visible = False
    Listbox.AddItem ("Name:" & linkAppearance.name)
    Set linkAppearance = linkAppearanceCltn.NextLinkAppearance
Wend
```

Remove

Method of VcLinkAppearanceCollection

This method lets you delete a link appearance. If the link appearance is used by a different object, it cannot be deleted. In the latter case **False** will be returned, otherwise **True**.

	Data Type	Explanation
Parameter:		
⇨ name	String	Name of the link appearance
	Possible Values:	Name of the color map
Return value	Boolean	Link appearance deleted (True)/not deleted (False)

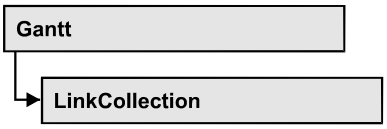
Update

Method of VcLinkAppearanceCollection

This method lets you update a collection of link appearances after having modified it.

	Data Type	Explanation
Return value	Boolean	Update successful (True) / not successful (False)

7.59 VcLinkCollection



An object of the type VcLinkCollection contains all available links. You can access all objects in an iterative loop by **For Each link In LinkCollection** or by the methods **First...** and **Next...**. The number of links in the collection object can be retrieved by the property **Count**.

Properties

- _NewEnum
- Count

Methods

- FirstLink
- NextLink
- SelectLinks

Properties

_NewEnum

Read Only Property of VcLinkCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all link objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method **GetEnumerator** is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim link As VcLink

For Each link In VcGantt1.LinkCollection
   Debug.Print link.Name
Next
```

Count

Read Only Property of VcLinkCollection

This property lets you retrieve the number of links in the link collection.

	Data Type	Explanation
Property value	Long	Number of links

Example Code

```
Dim linkCltn As VcLinkCollection
Dim numberLinks  As Integer

Set linkCltn = VcGantt1.LinkCollection
numberLinks = linkCltn.Count
```

Methods

FirstLink

Method of VcLinkCollection

This method can be used to access the initial value, i.e. the first link of a link collection, and to continue in a forward iteration loop by the method **NextLink** for the links following. If there is no link in the link collection, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcLink	First link

Example Code

```
Dim linkCltn As VcLinkCollection
Dim link As VcLink

Set linkCltn = VcGantt1.LinkCollection
Set link = linkCltn.FirstLink
```

NextLink

Method of VcLinkCollection

This method can be used in a forward iteration loop to retrieve subsequent links from a link collection after initializing the loop by the method **FirstLink**. If there is no link left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcLink	Subsequent link

Example Code

```
Dim linkCltn As VcLinkCollection
Dim link As VcLink
Dim stringLink As String

Set linkCltn = VcGantt1.LinkCollection
Set link = linkCltn.FirstLink

While Not link Is Nothing
     stringLink = link.AllData
     Listbox.AddItem (stringLink)
     Set link = linkCltn.NextLink
Wend
```

SelectLinks

Method of VcLinkCollection

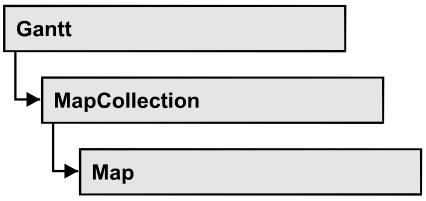
This method lets you specify the links that the link collection is to contain.

	Data Type	Explanation
Parameter:		
⇨ selectionType	SelectionTypeEnum	Links to be selected
	Possible Values:	
	vcAll 0	All objects in the diagram will be selected
	vcAllLinksCausingCycles 7	If this selection type is chosen, the link collection will contain all links that cause the existence of cycles. If these links are deleted, cycles will cede to exist in this chart.
	vcAllLinksInCycles 6	If this selection type is chosen, the link collection will contain all links that participate in forming cycles. Cycles are chains of nodes and links of which the beginning and end join.
	vcAllVisible 1	All visible objects will be selected
	vcSelected 2	All marked objects will be selected
Return value	Long	Number of links selected

Example Code

```
Dim linkCollection As VcLinkCollection

Set linkCollection = VcGantt1.LinkCollection
linkCollection.SelectGroups (vcAllMarked)
```

7.60 VcMap



Maps define certain properties of nodes by data field entries, for example their background color which is based on the data of the node record.

In a map you can specify 150 map entries at maximum. By the call **For Each mapEntry In Map** you can retrieve all data field entries in an iterative loop.

Properties

- _NewEnum
- ConsiderFilterEntries
- Count
- Name
- Specification
- Type

Methods

- CreateEntry
- DeleteEntry
- FirstMapEntry
- GetMapEntry
- NextMapEntry

Properties

_NewEnum

Read Only Property of VcMap

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all map objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method **GetEnumerator** is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim map As VcMap

For Each map in VcGantt1.Map
   Debug.Print.map.Name
Next
```

ConsiderFilterEntries

Read Only Property of VcMap

This property lets you set/retrieve whether filters are considered when a map is assigned to data field entries so that ranges of values can also be specified as keys.

	Data Type	Explanation

Count

Read Only Property of VcMap

This property lets you retrieve the number of map entries in a map.

	Data Type	Explanation
Property value	Long	Number of map entries

Example Code

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim numberOfEntries As Long

Set mapCltn = VcGantt1.MapCollection
mapCltn.SelectMaps vcAnyMap
Set map = mapCltn.MapByName("Map1")
numberOfEntries = map.count
```

Name

Read Only Property of VcMap

This property lets you retrieve the name of a map.

	Data Type	Explanation
Property value	String **Possible Values:**	Name Name of the color map

Example Code

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapName As String

Set mapCltn = VcGantt1.MapCollection
mapCltn.SelectMaps (vcAnyMap)
Set map = mapCltn.FirstMap
mapName = map.Name
```

Specification

Read Only Property of VcMap

This property lets you retrieve the specification of a map. A specification is a string that contains legible ASCII characters from 32 to 127 only, so it can be stored without problems to text files or data bases. This allows for persistency. A specification can be used to create a map by the method **VcMapCollection.AddBySpecification**.

	Data Type	Explanation
Property value	String **Possible Values:**	Specification of the map Name of the color map

Type

Property of VcMap

This property lets you enquire/set the map type.

	Data Type	Explanation
Property value	MapTypeEnum **Possible Values:** vcAnyMap 0 vcColorMap 1 vcFontMap 8 vcGraphicsFileMap 7 vcMillimeterMap 9 vcNumberMap 10 vcPatternMap 3	map type **any** (used only for selecting) **Colors** **Fonts** **Graphics file** **Millimeters** **Numbers** **Pattern**

	vcTextMap 6	**Text**

Example Code

```
Dim mapCollection As VcMapCollection
Dim map As VcMap

Set mapCollection = VcGantt1.MapCollection
mapCollection.SelectMaps (vcAnyMap)
Set map = mapCollection.MapByName("Map1")
map.Type = vcPatternMap
```

Methods

CreateEntry

Method of VcMap

This method lets you create a new entry (a new row) for a map. To make the entry work, the method **MapCollection.Update()** should be invoked after creating.

	Data Type	Explanation
Return value	VcMapEntry	Map entry

Example Code

```
Set mapCltn = VcGantt1.MapCollection
Set map = mapCltn.Add("MapColor")

map.Type = vcColorMap
Set mapEntry = map.CreateEntry
mapEntry.DataFieldValue = "Green"
mapEntry.Color = RGB(0, 255, 0)
Set mapEntry = map.CreateEntry
mapEntry.DataFieldValue = "Red"
mapEntry.Color = RGB(255, 0, 0)
mapCltn.Update
```

DeleteEntry

Method of VcMap

This method lets you delete an entry (a row) of the map. To make the deletion work, the method **MapCollection.Update()** should be invoked after deleting.

	Data Type	Explanation
Parameter:		
⇨ mapEntry	VcMapEntry	Map entry
Return value	Boolean	Map entry was (True) / was not (False) deleted successfully

Example Code

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry

Set mapCltn = VcGantt1.MapCollection
mapCltn.SelectMaps vcAnyMap
Set map = mapCltn.MapByName("Map1")
Set mapEntry = map.FirstMapEntry

map.DeleteEntry mapEntry
mapCltn.Update
```

FirstMapEntry

Method of VcMap

This method can be used to access the initial value, i.e. the first entry of a map object and then to continue in a forward iteration loop by the method **NextMapEntry** for the entries following. If there is no entry in the map, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcMapEntry	First map entry

Example Code

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry

Set mapCltn = VcGantt1.MapCollection
mapCltn.SelectMaps (vcAnyMap)

Set map = mapCltn.FirstMap
Set mapEntry = map.FirstMapEntry
```

GetMapEntry

Method of VcMap

This method returns the corresponding map entry for the given data field value.

	Data Type	Explanation
Return value	VcMapEntry	Map entry according to field value

NextMapEntry

Method of VcMap

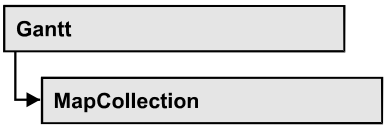
This method can be used in a forward iteration loop to retrieve subsequent entries (rows) from a map object after initializing the loop by the method **FirstMapEntry**. If there is no map entry left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcMapEntry	Subsequent map entry

Example Code

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry

Set mapCltn = VcGantt1.MapCollection
mapCltn.SelectMaps (vcAnyMap)

Set map = mapCltn.FirstMap
Set mapEntry = map.FirstMapEntry

While Not mapEntry Is Nothing
   List1.AddItem (mapEntry.Legend)
   Set mapEntry = map.NextMapEntry
Wend
```

7.61 VcMapCollection



An object of the type VcMapCollection contain the maps, which were assigned to the collection by the method **SelectMaps**. You can access all objects in an iterative loop by **For Each map In MapCollection** or by the methods **First...** and **Next...**. You can access a single map using the methods **MapByName** and **MapByIndex**. The number of maps in the collection object can be retrieved by the property **Count**. The methods **Add**, **Copy** and **Remove** allow to handle the maps in the corresponding way.

Properties

- _NewEnum
- Count

Methods

- Add
- AddBySpecification
- Copy
- FirstMap
- MapByIndex
- MapByName
- NextMap
- Remove
- SelectMaps
- Update

Properties

_NewEnum

Read Only Property of VcMapCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all map objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method

GetEnumerator is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim map As VcMap

For Each map In VcGantt1.MapCollection
   Debug.Print map.Count
Next
```

Count

Read Only Property of VcMapCollection

This property lets you retrieve the number of maps in the MapCollection object.

	Data Type	Explanation
Property value	Long	Number of maps

Example Code

```
Dim mapCltn As VcMapCollection
Dim numberOfMaps As Long

Set mapCltn = VcGantt1.MapCollection
mapCltn.SelectMaps vcAnyMap
numberOfMaps = mapCltn.Count
```

Methods

Add

Method of VcMapCollection

By this method you can create a map as a member of the MapCollection. If the name was not used before, the new map object will be returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
Parameter:		
⇨ mapName	String	Map name

	Possible Values:	Name of the color map
Return value	VcMap	New map object

Example Code

```
Set newMap = VcGantt1.MapCollection.Add("map1")
```

AddBySpecification

Method of VcMapCollection

This method lets you create a map by using a map specification. This way of creating allows map objects to become persistent. The specification of a map can be saved and re-loaded (see VcMap property **Specification**). In a subsequent session the map can be created again from the specification and is identified by its name.

	Data Type	**Explanation**
Parameter:		
⇨ Specification	String	Map specification
	Possible Values:	Name of the color map
Return value	VcMap	New map object

Copy

Method of VcMapCollection

By this method you can copy a map. If the map that is to be copied exists, and if the name for the new map does not yet exist, the new map object is returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	**Explanation**
Parameter:		
⇨ mapName	String	Name of the map to be copied
	Possible Values:	Name of the color map
⇨ newMapName	String	Name of the new map
	Possible Values:	Name of the color map

Return value	VcMap	Map object

FirstMap

Method of VcMapCollection

This method can be used to access the initial value, i.e. the first map of a map collection and then to continue in a forward iteration loop by the method **NextMap** for the maps following. If there is no map in the MapCollection, a **none** object will be returned (**Nothing** in Visual Basic). Before using this method, a selection of maps needs to have been defined by the method **VcMapCollection.SelectMaps**.

	Data Type	Explanation
Return value	VcMap	First map

Example Code

```
Dim mapCltn As VcMapCollection
Dim map As VcMap

Set mapCltn = VcGantt1.MapCollection
mapCltn.SelectMaps (vcAnyMap)
Set map = mapCltn.FirstMap
```

MapByIndex

Method of VcMapCollection

This method lets you access a map by its index. If a map of the specified index does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter:		
⇨ index	Integer **Possible Values:**	Index of the map Data field index
Return value	VcMap	Map object returned

MapByName

Method of VcMapCollection

By this method you can get a map by its name. Beforehand, a set of maps needs to be selected by the method **SelectMaps**. If a map of the specified name does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter:		
⇨ mapName	String	Name of the map
	Possible Values:	Name of the color map
Return value	VcMap	Map

Example Code

```
Dim mapCltn As VcMapCollection
Dim map As VcMap

Set mapCltn = VcGantt1.MapCollection
mapCltn.SelectMaps (vcAnyMap)
Set map = mapCltn.MapByName("Map_1")
```

NextMap

Method of VcMapCollection

This method can be used in a forward iteration loop to retrieve subsequent maps from a map collection after initializing the loop by the method **FirstMap**. If there is no map left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcMap	Subsequent map

Example Code

```
Dim mapCltn As VcMapCollection
Dim map As VcMap

Set mapCltn = VcGantt1.MapCollection
mapCltn.SelectMaps (vcAnyMap)
Set map = mapCltn.FirstMap

While Not map Is Nothing
   List1.AddItem map.Name
   Set map = mapCltn.NextMap
Wend
```

Remove

Method of VcMapCollection

This method lets you delete a map. If the map is used in another object, it cannot be deleted. Then False will be returned, otherwise True.

	Data Type	Explanation
Parameter:		
⇨ mapName	String **Possible Values:**	Map name Name of the color map
Return value	Boolean	Map deleted (True)/not deleted (False)

SelectMaps

Method of VcMapCollection

This method lets you specify the map types that your map collection is allowed to contain.

	Data Type	Explanation
Parameter:		
⇨ selectionType	MapTypeEnum **Possible Values:** vcAnyMap 0 vcColorMap 1 vcFontMap 8 vcGraphicsFileMap 7 vcMillimeterMap 9 vcNumberMap 10 vcPatternMap 3 vcTextMap 6	Map type to be selected **any** (used only for selecting) **Colors** **Fonts** **Graphics file** **Millimeters** **Numbers** **Pattern** **Text**
Return value	Long	Number of maps selected

Example Code

```
Dim mapCltn As VcMapCollection
Dim map As VcMap

Set mapCltn = VcGantt1.MapCollection
mapCltn.SelectMaps vcAnyMap
```

Update

Method of VcMapCollection

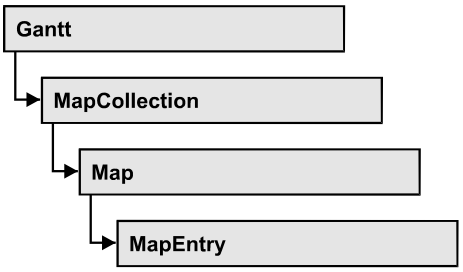
This method has to be used when map modifications have been made. The method **UpdateMaps** updates all objects that are concerned by the maps you have edited. You should call this method at the end of the code that defines the maps and the map collection. Otherwise the update will be processed before all map definitions are processed.

	Data Type	Explanation
Return value	Boolean	update successful (True)/ not successful (False)

Example Code

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry

Set mapCltn = VcGantt1.MapCollection
mapCltn.SelectMaps vcAnyMap
Set map = mapCltn.MapByName("Map1")
Set mapEntry = map.FirstMapEntry

While Not mapEntry.DataFieldValue = "A"
   Set mapEntry = map.NextMapEntry
Wend

mapEntry.Color = RGB(0, 0, 0)

mapCltn.Update
```

7.62 VcMapEntry



An object of the type VcMapEntry is a map entry and therefore an element of a map. A map entry is defined by the combination of a data field content of the node´s record, a color or graphics file and a legend text.

In each map you can specify up to a maximum of 150 map entries. If you need further map entries, please specify a new map, e. g. as a copy of the current one.

Properties

- ColorAsARGB
- DataFieldValue
- FontBody
- FontName
- FontSize
- GraphicsFileName
- Legend
- Millimeter
- Number
- Pattern

Properties

ColorAsARGB

Property of VcMapEntry

for Color Maps: This property lets you set or retrieve the color value of a map entry. Color values have a transparency or alpha value, followed by a value for a red, a blue and a green partition (ARGB). The values range between 0..255. An alpha value of 0 equals complete transparency, whereas

255 represents a completely solid color. When casting an RGB value on an ARGB value, an alpha value of 255 has to be added.

	Data Type	Explanation
Property value	Color	ARGB color values ({0...255},{0...255},{0...255},{0...255})

Example Code

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry
Dim colorOfMapEntry As OLE_COLOR

Set mapCltn = VcGantt1.MapCollection
mapCltn.SelectMaps (vcColorMap)
Set map = mapCltn.MapByName("Map1")
Set mapEntry = map.FirstMapEntry

colorOfMapEntry = mapEntry.Color
```

DataFieldValue

Property of VcMapEntry

This property lets you set or retrieve the content of a data of each map entry.

	Data Type	Explanation
Property value	String **Possible Values:**	Content of the data field Name of the color map

Example Code

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry
Dim dataFieldValue As String

Set mapCltn = VcGantt1.MapCollection
mapCltn.SelectMaps (vcAnyMap)
Set map = mapCltn.MapByName("Map1")
Set mapEntry = map.FirstMapEntry

dataFieldValue = mapEntry.DataFieldValue
```

FontBody

Property of VcMapEntry

for font maps: This property lets you set or retrieve the font body of the map entry.

	Data Type	Explanation
Property value	FontBodyEnum **Possible Values:** vcBold 2 vcBoldItalic 4 vcItalic 3 vcRegular 1	Font body bold bold and italic italic regular

Example Code

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry
Dim FontBodyOfMapEntry As FontBodyEnum

Set mapCltn = VcGantt1.MapCollection
mapCltn.SelectMaps (vcFontMap)
Set map = mapCltn.MapByName("Map1")
Set mapEntry = map.FirstMapEntry

FontBodyOfMapEntry = vcBold
```

FontName

Property of VcMapEntry

for font maps: This property lets you set or retrieve the font name of the map entry.

	Data Type	Explanation
Property value	String **Possible Values:**	Font type Name of the color map

Example Code

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry
Dim FontNameOfMapEntry As String

Set mapCltn = VcGantt1.MapCollection
mapCltn.SelectMaps (vcFontMap)
Set map = mapCltn.MapByName("Map1")
Set mapEntry = map.FirstMapEntry

FontNameOfMapEntry = "Arial"
```

FontSize

Property of VcMapEntry

for font maps: This property lets you set or retrieve the font name of he map entry.

	Data Type	Explanation
Property value	Long	Font size

Example Code

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry
Dim FontSizeOfMapEntry As Long

Set mapCltn = VcGantt1.MapCollection
mapCltn.SelectMaps (vcFontMap)
Set map = mapCltn.MapByName("Map1")
Set mapEntry = map.FirstMapEntry

FontSizeOfMapEntry = 12
```

GraphicsFileName

Property of VcMapEntry

For graphics file maps: This property lets you set or retrieve the graphics file name of a map entry. *Available formats:*

- *.BMP (Microsoft Windows Bitmap)
- *.EMF (Enhanced Metafile or Enhanced Metafile Plus)
- *.GIF (Graphics Interchange Format)
- *.JPG (Joint Photographic Experts Group)
- *.PNG (Portable Network Graphics)
- *.TIF (Tagged Image File Format)
- *.VMF (Viewer Metafile)
- *.WMF (Microsoft Windows Metafile, probably with EMF included)

EMF, EMF+, VMF and WMF are vector formats that allow to store a file independent of pixel resolution. All other formats are pixel-oriented and confined to a limited resolution.

The VMF format basically has been deprecated, but it will still be supported for some time to maintain compatibility with existing applications.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the graphics file Name of the color map

Example Code

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry

Set mapCltn = VcGantt1.MapCollection
mapCltn.SelectMaps (vcGraphicsFileMap)
Set map = mapCltn.MapByName("Map1")
Set mapEntry = map.FirstMapEntry

mapEntry.GraphicsFileName = AppPath & "\picture1.bmp"
```

Legend

Property of VcMapEntry

This property lets you set or retrieve the legend text of a map entry.

	Data Type	Explanation
Property value	String **Possible Values:**	Legend text Name of the color map

Example Code

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry
Dim LegendOfMapEntry As String

Set mapCltn = VcGantt1.MapCollection
mapCltn.SelectMaps (vcFontMap)
Set map = mapCltn.MapByName("Map1")
Set mapEntry = map.FirstMapEntry

LegendOfMapEntry = "1. activity"
```

Millimeter

Property of VcMapEntry

for millimeter maps: This property lets you set or retrieve the millimetre value of a map entry.

	Data Type	Explanation
Property value	Long	1/100 units

Example Code

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry
Dim MillimeterOfMapEntry As Long

Set mapCltn = VcGantt1.MapCollection
mapCltn.SelectMaps (vcMillimeterMap)
Set map = mapCltn.MapByName("Map1")
Set mapEntry = map.FirstMapEntry

MillimeterOfMapEntry = 3
```

Number

Property of VcMapEntry

for numeric maps: This property lets you set / retrieve a numeric value of a map.

	Data Type	Explanation
Property value	Long	Numeric value

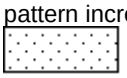
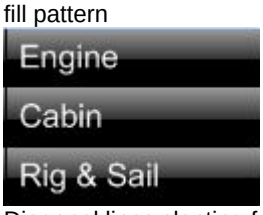
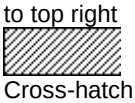
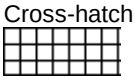
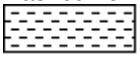
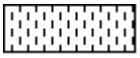
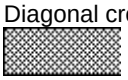
Example Code

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry
Dim MillimeterOfMapEntry As Long

Set mapCltn = VcGantt1.MapCollection
mapCltn.SelectMaps (vcMillimeterMap)
Set map = mapCltn.MapByName("Map1")
Set mapEntry = map.FirstMapEntry

MillimeterOfMapEntry = 3
```

Pattern

Property of VcMapEntry

for pattern maps (vcPatternMap): This property lets you set or retrieve the pattern of a map entry.

	Data Type	Explanation
Property value	FillPatternEnum	Pattern type
	Possible Values:	
	vc05PercentPattern... vc90PercentPattern 01 - 11	Dots in foreground color on background color, the density of the foreground pattern increasing with the percentage 
	vcAeroGlassPattern 40	Vertical color gradient in the color of the fill pattern 
	vcBDiagonalPattern 5	Diagonal lines slanting from bottom left to top right 
	vcCrossPattern 6	Cross-hatch pattern 
	vcDarkDownwardDiagonalPattern 2014	Diagonal lines slanting from top left to bottom right; spaced 50% closer than vcFDiagonalPattern and of twice the line width 
	vcDarkHorizontalPattern 2023	Horizontal lines spaced 50% closer than vcHorizontalPattern and of twice the line width 
	vcDarkUpwardDiagonalPattern 2015	Diagonal lines slanting from bottom left to top right, spaced 50% closer than vcBDiagonalPattern and of twice the line width 
	vcDarkVerticalPattern 2022	Vertical lines spaced 50% closer than vcVerticalPattern and of of twice the line width 
	vcDashedDownwardDiagonalPattern 2024	Dashed diagonal lines from top left to bottom right 
	vcDashedHorizontalPattern 2026	Dashed horizontal lines 
	vcDashedUpwardDiagonalPattern 2025	Dashed diagonal lines from bottom left to top right 
	vcDashedVerticalPattern 2027	Dashed vertical lines 
	vcDiagCrossPattern 7	Diagonal cross-hatch pattern, small 
	vcDiagonalBrickPattern 2032	Diagonal brick pattern 

vcDivotPattern 2036	Divot pattern
vcDottedDiamondPattern 2038	Diagonal cross-hatch pattern of dotted lines
vcDottedGridPattern 2037	Cross-hatch pattern of dotted lines
vcFDiagonalPattern 4	Diagonal lines slanting from top left to bottom right
vcHorizontalBrickPattern 2033	Horizontal brick pattern
vcHorizontalGradientPattern 52	Horizontal color gradient
vcHorizontalPattern 3	Horizontal lines
vcLargeCheckerboardPattern 2044	Checkerboard pattern showing squares of twice the size of vcSmallChecker-BoardPattern
vcLargeConfettiPattern 2029	Confetti pattern, large
vcLightDownwardDiagonalPattern 2012	Diagonal lines slanting to from top left to bottom right; spaced 50% closer than vcBDiagonalPattern
vcLightHorizontalPattern 2019	Horizontal lines spaced 50% closer than vcHorizontalPattern
vcLightUpwardDiagonalPattern 2013	Diagonal lines slanting from bottom left to top right, spaced 50% closer than vcBDiagonalPattern
vcLightVerticalPattern 2018	Vertical lines spaced 50% closer than vcVerticalPattern
vcNarrowHorizontalPattern 2021	Horizontal lines spaced 75 % closer than vcHorizontalPattern
vcNarrowVerticalPattern 2020	Vertical lines spaced 75% closer than vcVerticalPattern
vcNoPattern 1276	No fill pattern
vcOutlinedDiamondPattern 2045	Diagonal cross-hatch pattern, large
vcPlaidPattern 2035	Plaid pattern
vcShinglePattern 2039	Diagonal shingle pattern
vcSmallCheckerBoardPattern 2043	Checkerboard pattern

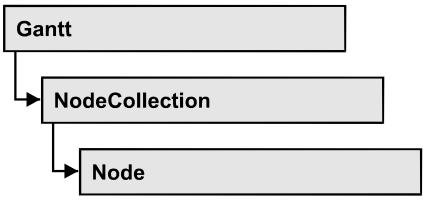
vcSmallConfettiPattern 2028	Confetti pattern
vcSmallGridPattern 2042	Cross-hatch pattern spaced 50% closer than vcCrossPattern
vcSolidDiamondPattern 2046	Checkerboard pattern showing diagonal squares
vcSpherePattern 2041	Checkerboard of spheres
vcTrellisPattern 2040	Trellis pattern
vcVerticalBottomLightedConvexPattern 43	Vertical color gradient from dark to bright
vcVerticalConcavePattern 40	Vertical color gradient from dark to bright to dark
vcVerticalConvexPattern 41	Vertical color gradient from bright to dark to bright
vcVerticalGradientPattern 62	Vertical color gradient
vcVerticalPattern 2	Vertical lines
vcVerticalTopLightedConvexPattern 42	Vertical color gradient from bright to dark
vcWavePattern 2031	Horizontal wave pattern
vcWeavePattern 2034	Interwoven stripe pattern
vcWideDownwardDiagonalPattern 2016	Diagonal lines slanting from top left to bottom right, showing the same spacing but three times the line width of vcF-DiagonalPattern
vcWideUpwardDiagonalPattern 2017	Diagonal lines slanting from bottom left to top right right, showing the same spacing but three times the line width of vcBDiagonalPattern
vcZigZagPattern 2030	Horizontal zig-zag lines

Example Code

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry
Dim pattern As FillPatternEnum
```

```
Set mapCltn = VcGantt1.mapCollection
mapCltn.SelectMaps (vcPatternMap)
Set map = mapCltn.MapByName("Map1")
Set mapEntry = map.FirstMapEntry

pattern = vcBDiagonalPattern
```

7.63 VcNode



A node is a basic element of a Gantt diagram. Nodes can be linked to form a structure. What a node looks like is determined by layers, the filters of which are matching the nodes. Nodes can be inserted either interactively or by the VcGantt methods **InsertNodeRecord** or **Open**.

Properties

- AllData
- DataField
- ID
- IncomingLinks
- MarkNode
- MoveMode
- OutgoingLinks
- SnapTargetMode
- SuperGroup
- UpdateBehaviorName

Methods

- DataRecord
- DeleteNode
- GetPositionInView
- GetPositionInViewAsVariant
- NodeRowInView
- OutlineIndent
- OutlineOutdent
- RelatedDataRecord
- SetPositionInView
- UpdateNode

Properties

AllData

Property of VcNode

This record lets you set or retrieve all data of a node at once. When setting the property, a CSV string (using semicolons as separators) or a variant that contains all data fields of the node in an array are allowed. When retrieving the property, a string will be returned. (See also **InsertNodeRecord**.)

	Data Type	Explanation
Property value	String/data field	All data of the data set

Example Code

```
Private Sub VcGantt1_OnNodeModify(ByVal node As VcGanttLib.VcNode,  _
                                  ByVal modificationType As _
                                  VcGanttLib.ModificationTypeEnum, _
                                  returnStatus As Variant)

    Dim allDataOfNode As String

    returnStatus = vcRetStatFalse

    allDataOfNode = node.AllData
    MsgBox allDataOfNode

End Sub
```

DataField

Property of VcNode

This property lets you assign/retrieve data to/from the data field of a node. If the data field was modified by the **DataField** property, the diagram needs to be updated by the **UpdateNode** method.

	Data Type	Explanation
Parameter: ⇨ index	Integer **Possible Values:**	Index of data field Data field index
Property value	Variant	Content of the data field

Example Code

```
Private Sub VcGantt1_OnNodeRClick(ByVal node As VcGanttLib.VcNode, _
                                  ByVal location As VcGanttLib.LocationEnum, _
```

```
                                ByVal x As Long, ByVal y As Long, _
                                returnStatus As Variant)

   If MsgBox("Delete Node: " & node.dataField(0), vbYesNo, "Delete Node") = _
                                vbYes Then node.DeleteNode

   returnStatus = vcRetStatNoPopup

End Sub
```

ID

Read Only Property of VcNode

By this property you can retrieve the ID of a node.

	Data Type	Explanation
Property value	String	Node ID
	Possible Values:	Name of the color map

IncomingLinks

Read Only Property of VcNode

This property gives access to all incoming links of a node.

	Data Type	Explanation
Property value	VcLinkCollection	Link collection

Example Code

```
Private Sub VcGantt1_OnNodeRClick(ByVal node As VcGanttLib.VcNode, _
                                ByVal location As VcGanttLib.LocationEnum, _
                                ByVal x As Long, ByVal y As Long, _
                                returnStatus As Variant)

    Dim incomingLinks As VcLinkCollection
    Dim link As VcLink
    Dim predecessorNode As VcNode

    Set incomingLinks = node.IncomingLinks

    For Each link In incomingLinks
        Set predecessorNode = link.PredecessorNode
        predecessorNode.MarkNode = True
    Next link

    returnStatus = vcRetStatNoPopup

End Sub
```

MarkNode

Property of VcNode

By this property you can set or retrieve whether a node is marked. The marking assigned will be visible only if on the **Nodes** property page the marking type **No Mark** was not selected.

	Data Type	Explanation
Property value	Boolean	Node marked/not marked
	Possible Values:	
		Group invisible/visible group nodes are/are not visible

Example Code

```
Dim nodeCltn As VcNodeCollection
Dim node As VcNode
Dim predecessor As VcNode
Dim linkCltn As VcLinkCollection
Dim link As VcLink

Set nodeCltn = VcGantt1.nodeCollection
nodeCltn.SelectNodes (vcAll)

For Each node In nodeCltn
    Set linkCltn = node.IncomingLinks
    For Each link In linkCltn
        Set predecessor = link.predecessorNode
        predecessor.MarkNode = True
    Next link
Next node
```

MoveMode

Property of VcNode

This property lets you set or retrieve the direction(s) that a node interactively can be moved to.

	Data Type	Explanation
Property value	NodeMoveModeEnum	Mode of moving a node
	Possible Values:	
	vcNodeMoveModeAutomaticXOrY 4	Move mode either **in x** or in **y direction**. The direction of the dragging automatically determines whether the x or the y direction is selected.
	vcNodeMoveModeNoMove 0	**No move** mode
	vcNodeMoveModeX 1	Move mode **in x direction**
	vcNodeMoveModeXY 3	Move mode **in x and y direction**
	vcNodeMoveModeY 2	Move mode **in y direction**

Example Code

```
Dim nodeCltn As VcNodeCollection
Dim node As VcNode

Set nodeCltn = VcGantt1.NodeCollection
nodeCltn.SelectNodes (vcAll)

For Each node In nodeCltn
    If node.DataField(2) < Now Then node.MoveMode = vcNodeMoveModeNoMove
Next node
```

OutgoingLinks

Read Only Property of VcNode

This property gives access to the set of links that leave a node.

	Data Type	Explanation
Property value	VcLinkCollection	Link collection

Example Code

```
Private Sub VcGantt1_OnNodeRClick(ByVal node As VcGanttLib.VcNode, _
                                  ByVal location As VcGanttLib.LocationEnum, _
                                  ByVal x As Long, ByVal y As Long, _
                                  returnStatus As Variant)

    Dim outgoingLinks As VcLinkCollection
    Dim link As VcLink
    Dim successorNode As VcNode

    Set outgoingLinks = node.outgoingLinks

    For Each link In outgoingLinks
        Set successorNode = link.successorNode
        successorNode.MarkNode = True
    Next link

    returnStatus = vcRetStatNoPopup

End Sub
```

SnapTargetMode

Property of VcNode

This property lets you set or retrieve whether this node is to be selected as possible snap target manually or automatically.

	Data Type	Explanation
Property value	NodeSnapTargetModeEnum **Possible Values:**	This node's selection mode for moving with snap targets switched on **Default value:** vcNSTMAutomatically

vcNSTMAutomatically 1	Node is automatically selected as snap target when **VcGantt. SnapTargetNodesSelectionMode** has been set to **vcAutomatically**. Does not affect **vcUserSelection**. Only the selected nodes will be checked on the property's value.
vcNSTMYesOnUserSelection 2	Node is selected as snap target when **VcGantt. SnapTargetNodesSelectionMode** has been set to **vcUserSelection**. Does not affect **vcAutomatically**. ALL nodes will be checked on the property's value.
vcNSTMNo 0	Node is not selected as snap target.

SuperGroup

Read Only Property of VcNode

This property lets you enquire the group that this node belongs to.

	Data Type	Explanation
Property value	VcGroup	Group that the node belongs to

Example Code

```
Private Sub VcGantt1_OnNodeRClick(ByVal node As VcGanttLib.VcNode, _
                                  ByVal location As VcGanttLib.LocationEnum, _
                                  ByVal x As Long, ByVal y As Long, _
                                  returnStatus As Variant)

    returnStatus = vcRetStatNoPopup
    Dim group As VcGroup
    Set group = node.SuperGroup
    Label1.Caption = "Group: " & group.Name

End Sub
```

UpdateBehaviorName

Property of VcNode

This property lets you set or retrieve the name of the UpdateBehavior.

	Data Type	Explanation
Property value	String	Name of the UpdateBehavior
	Possible Values:	Name of the color map

Methods

DataRecord

Method of VcNode

This property lets you retrieve the node as a data record object. The properties of the data record object give access to the corresponding data table and the data table collection.

	Data Type	Explanation
Return value	VcDataRecord	Data record returned

DeleteNode

Method of VcNode

This method lets you delete a node.

	Data Type	Explanation
Return value	Boolean	Node was (true) / was not (false) deleted successfully

Example Code

```
Private Sub VcGantt1_OnNodeRClick(ByVal node As VcGanttLib.VcNode, _
                                  ByVal location As _
                                  VcGanttLib.LocationEnum, ByVal x As Long, _
                                  ByVal y As Long, returnStatus As Variant)

    If MsgBox("Delete Node: " & node.DataField(0), vbYesNo, "Delete Node") = _
              vbYes Then node.DeleteNode
    returnStatus = vcRetStatNoPopup

End Sub
```

GetPositionInView

Method of VcNode

This method lets you enquire the position of a node in the visible area of the diagram.

Note: If you use VBScript, you can only use the analogous method **GetPositionInViewAsVariant** because of the parameters by Reference.

	Data Type	Explanation
Parameter:		
ViewReferencePoint	ViewReferencePointEnum	Reference point (of the diagram)
	Possible Values: vcVRPBottomCenter 28 vcVRPBottomLeft 27 vcVRPBottomRight 29 vcVRPCenterCenter 25 vcVRPCenterLeft 24 vcVRPCenterRight 26 vcVRPTopCenter 22 vcVRPTopLeft 21 vcVRPTopRight 23	 bottom center bottom left bottom right center center center left center right top center top left top right
NodeReferencePoint	NodeReferencePointEnum	Node reference point
	Possible Values: vcNRPBottomCenter 28 vcNRPBottomLeft 27 vcNRPBottomRight 29 vcNRPCenterCenter 25 vcNRPCenterLeft 24 vcNRPCenterRight 26 vcNRPTopCenter 22 vcNRPTopLeft 21 vcNRPTopRight 23	 bottom center bottom left bottom right center center center left center right top center top left top right
⇦ XOffset	Long	X value of the offset (= distance of the node reference point and the reference point) (unit: pixels)
⇦ YOffset	Long	Y value of the offset (unit: pixels)
Return value	Void	

GetPositionInViewAsVariant

Method of VcNode

This method is identical with the method **GetPositionInView** except for the parameters. It was necessary to implement this event because some languages (e.g. VBScript) can use parameters by Reference (indicated by ⇦) only if the type of these parameters is VARIANT.

NodeRowInView

Method of VcNode

This method lets you enquire whether (True) or not (False) the row that this node is in is displayed in the visible section of the diagram.

	Data Type	Explanation
Return value	Boolean	Row is/is not in the visible section of the diagram

Example Code

```
Dim node As VcNode

Set node = VcGantt1.GetNodeByID("15")
If Not node.NodeRowInView Then _
        Call VcGantt1.ScrollToNodeLine(node, vcVerCenterAligned)
```

OutlineIndent

Method of VcNode

This method allows to demote a node in a diagram hierarchy, the node being indented, i.e. moved towards the right within the table while remaining in its row. This method corresponds to the **Outline indent** item in the node context menu.

The return value indicates whether the method could be performed successfully. For example, nodes on the lowest level cannot be demoted.

	Data Type	Explanation
Return value	Boolean	method successful (True)/ not successful (False)

Example Code

```
Private Sub CommandOutlineIndent_Click()
    Dim anode As VcNode
    Set anode = getSelectedNode   'Function returns selected node
    MsgBox anode.OutlineIndent
    Set anode = Nothing
End Sub
```

OutlineOutdent

Method of VcNode

This method allows to promote a node in a diagram hierarchy, the node being outdented, i.e. moved to the left within the table and remaining in its row. This method corresponds to the **Outline outdent** item in the context menu for nodes.

The return value indicates whether the method could be performed successfully. For example, nodes on the highest level cannot be promoted.

	Data Type	Explanation
Return value	Boolean	Method successful (True)/ not successful (False)

Example Code

```
Private Sub CommandOutlineOutdent_Click()
    Dim anode As VcNode
    Set anode = getSelectedNode   'Function returns selected node
    MsgBox anode.OutlineOutdent
    Set anode = Nothing
End Sub
```

RelatedDataRecord

Method of VcNode

This property lets you retrieve a data record from a data table that is related to the node data table. The index passed by the parameter denotes the field in the data record that holds the key of the related data record.

	Data Type	Explanation
Parameter:		
⇨ index	Integer **Possible Values:**	Index of data field that holds the key Data field index
Return value	VcDataRecord	Related data record returned

SetPositionInView

Method of VcNode

This method lets you set that the node will be displayed in a certain position in the visible area of the diagram. This position is specified by a distance vector (x,y) between a special node reference point and a special (diagram) reference point.

	Data Type	Explanation
Parameter:		
ViewReferencePoint	ViewReferencePointEnum	Reference point (of the diagram)
	Possible Values:	
	vcVRPBottomCenter 28	bottom center
	vcVRPBottomLeft 27	bottom left
	vcVRPBottomRight 29	bottom right
	vcVRPCenterCenter 25	center center
	vcVRPCenterLeft 24	center left
	vcVRPCenterRight 26	center right
	vcVRPTopCenter 22	top center

	vcVRPTopLeft 21 vcVRPTopRight 23	top left top right
NodeReferencePoint	NodeReferencePointEnum	Node reference point
	Possible Values: vcNRPBottomCenter 28 vcNRPBottomLeft 27 vcNRPBottomRight 29 vcNRPCenterCenter 25 vcNRPCenterLeft 24 vcNRPCenterRight 26 vcNRPTopCenter 22 vcNRPTopLeft 21 vcNRPTopRight 23	 bottom center bottom left bottom right center center center left center right top center top left top right
⇦ XOffset	Long	X-value of the offset (= distance of the node reference point and the reference point) (unit: pixels)
⇦ YOffset	Long	Y-value of the offset (unit: pixels)
Return value	Void	

Example Code

```
' scroll the diagram so that the vector between the bottom right corner of the
node and the bottom right corner of the diagram is (-10, -10)

mySelNode.SetPositionInView vcVRPBottomRight, vcNRPBottomRight, -10, -10
```

UpdateNode

Method of VcNode

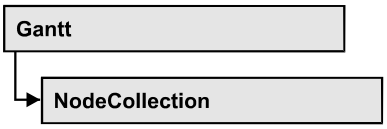
If data fields of a node have been modified by the **DataField** property, the diagram needs to be updated by the **UpdateNode** method.

	Data Type	Explanation
Return value	Boolean	Node was (true) / was not (false) updated successfully

Example Code

```
Dim nodeCltn As VcNodeCollection
Dim node As VcNode

Set nodeCltn = VcGantt1.NodeCollection
Set node = nodeCltn.FirstNode

node.DataField(12) = "Group A"
node.UpdateNode
```

7.64 VcNodeCollection



An object of the type VcNodeCollection contains all nodes available in the diagram. You can select a part of them by using the method **SelectNodes**. You can access all objects in an iterative loop by **For Each node In Node-Collection** or by the methods **First...** and **Next...**. The number of nodes in the collection object can be retrieved by the property **Count**.

Properties

- _NewEnum
- Count

Methods

- FirstNode
- NextNode
- SelectNodes

Properties

_NewEnum

Read Only Property of VcNodeCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all node objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method **GetEnumerator** is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim node As VcNode

For Each node In VcGantt1.NodeCollection
```

```
   Debug.Print node.Name
Next
```

Count

Read Only Property of VcNodeCollection

This property lets you retrieve the number of nodes in the NodeCollection object.

	Data Type	Explanation
Property value	Long	Number of Nodes in the node collection

Example Code

```
Dim nodeCltn As VcNodeCollection

Set nodeCltn = VcGantt1.NodeCollection
MsgBox "Number of nodes: " & nodeCltn.Count
```

Methods

FirstNode

Method of VcNodeCollection

This method can be used to access the initial value, i.e. the first node of a NodeCollection, and then to continue in a forward iteration loop by the method **NextNode** for the nodes following. If there is no node in the Node-Collection, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcNode	First node

Example Code

```
Dim nodeCltn As VcNodeCollection
Dim node As VcNode

Set nodeCltn = VcGantt1.NodeCollection
Set node = nodeCltn.FirstNode
```

NextNode

Method of VcNodeCollection

This method can be used in a forward iteration loop to retrieve subsequent nodes from a node collection after initializing the loop by the method **FirstNode**. If there is no node left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcNode	Subsequent node

Example Code

```
Dim nodeCltn As VcNodeCollection
Dim node As VcNode

Set nodeCltn = VcGantt1.NodeCollection
Set node = nodeCltn.FirstNode

While Not node Is Nothing
     node.MarkNode = False
     Set node = nodeCltn.NextNode
Wend
```

SelectNodes

Method of VcNodeCollection

This method lets you specify the nodes to be collected by the NodeCollection object.

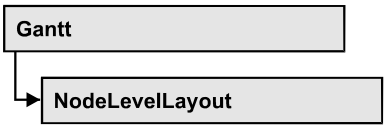
	Data Type	Explanation
Parameter:		
⇨ selType	SelectionTypeEnum	Nodes to be selected
	Possible Values:	
	vcAll 0	All objects in the diagram will be selected
	vcAllLinksCausingCycles 7	If this selection type is chosen, the link collection will contain all links that cause the existence of cycles. If these links are deleted, cycles will cede to exist in this chart.
	vcAllLinksInCycles 6	If this selection type is chosen, the link collection will contain all links that participate in forming cycles. Cycles are chains of nodes and links of which the beginning and end join.
	vcAllVisible 1	All visible objects will be selected
	vcSelected 2	All marked objects will be selected
Return value	Long	Number of nodes selected

Example Code

```
Dim nodeCltn As VcNodeCollection
Dim node As VcNode
```

```
Set nodeCltn = VcGantt1.NodeCollection
nodeCltn.SelectNodes vcSelected
```

7.65 VcNodeLevelLayout



An object of the type VcNodeLevelLayout defines the sorting of nodes as well as the appearance of node rows.

Properties

- CalendarGridName
- DateLineName
- RowBackColorAsARGB
- RowBackColorDataFieldIndex
- RowBackColorMapName
- RowPattern
- RowPatternColorAsARGB
- RowPatternColorDataFieldIndex
- RowPatternColorMapName
- RowPatternDataFieldIndex
- RowPatternMapName
- SeparationLineColor
- SeparationLineInterval
- SeparationLineThickness
- SeparationLineType
- ShowCalendarGrids
- ShowDateLines
- ShowSeparationLines
- ShowSeparationLinesAtTop
- SortDataFieldIndex
- SortOrder

Properties

CalendarGridName

Read Only Property of VcNodeLevelLayout

This property lets you set or retrieve the name of the calendar grid. You can also set this property in the **Nodes** section of the **Grouping** dialog.

	Data Type	Explanation

DateLineName

Property of VcNodeLevelLayout

This property lets you set or retrieve the name of the date line for this node level layout. You can also set this property in the **Grouping** dialog.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the date line Name of the color map

RowBackColorAsARGB

Property of VcNodeLevelLayout

This property lets you set or retrieve the background color of the rows. The default color is white.

	Data Type	Explanation
Property value	Color	ARGB color values ({0...255},{0...255},{0...255},{0...255})

RowBackColorDataFieldIndex

Property of VcNodeLevelLayout

This property lets you set or retrieve the data field index to be used with a color map specified by the property **RowBackColorMapName**. If you set this property to **-1**, no map will be used.

	Data Type	Explanation
Property value	Integer **Possible Values:**	Data field index Data field index

RowBackColorMapName

Property of VcNodeLevelLayout

This property lets you set or retrieve the name of a color map (type vcColorMap). If set to "", no map will be used. If a map name and additionally a data field index is specified in the property **RowBackColorDataFieldIndex**, then the background color is controlled by the map. If no data field entry applies, the background color that is specified in the property **RowBackColor** will be used.

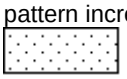
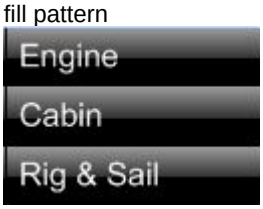
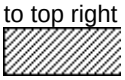
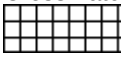
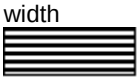
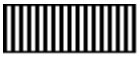
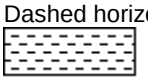
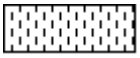
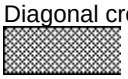
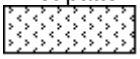
	Data Type	Explanation
Property value	String **Possible Values:**	Name of the color map Name of the color map

RowPattern

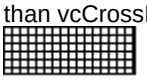
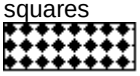
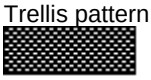
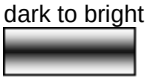
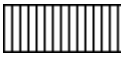
Read Only Property of VcNodeLevelLayout

This property lets you set or retrieve the background pattern of the node rows of this group level.

	Data Type	Explanation
Property value	FillPatternEnum **Possible Values:**	Pattern type

vc05PercentPattern... vc90PercentPattern 01 - 11	Dots in foreground color on background color, the density of the foreground pattern increasing with the percentage 
vcAeroGlassPattern 40	Vertical color gradient in the color of the fill pattern 
vcBDiagonalPattern 5	Diagonal lines slanting from bottom left to top right 
vcCrossPattern 6	Cross-hatch pattern 
vcDarkDownwardDiagonalPattern 2014	Diagonal lines slanting from top left to bottom right; spaced 50% closer than vcFDiagonalPattern and of twice the line width 
vcDarkHorizontalPattern 2023	Horizontal lines spaced 50% closer than vcHorizontalPattern and of twice the line width 
vcDarkUpwardDiagonalPattern 2015	Diagonal lines slanting from bottom left to top right, spaced 50% closer than vcBDiagonalPattern and of twice the line width 
vcDarkVerticalPattern 2022	Vertical lines spaced 50% closer than vcVerticalPattern and of of twice the line width 
vcDashedDownwardDiagonalPattern 2024	Dashed diagonal lines from top left to bottom right 
vcDashedHorizontalPattern 2026	Dashed horizontal lines 
vcDashedUpwardDiagonalPattern 2025	Dashed diagonal lines from bottom left to top right 
vcDashedVerticalPattern 2027	Dashed vertical lines 
vcDiagCrossPattern 7	Diagonal cross-hatch pattern, small 
vcDiagonalBrickPattern 2032	Diagonal brick pattern 
vcDivotPattern 2036	Divot pattern 

vcDottedDiamondPattern 2038	Diagonal cross-hatch pattern of dotted lines
vcDottedGridPattern 2037	Cross-hatch pattern of dotted lines
vcFDiagonalPattern 4	Diagonal lines slanting from top left to bottom right
vcHorizontalBrickPattern 2033	Horizontal brick pattern
vcHorizontalGradientPattern 52	Horizontal color gradient
vcHorizontalPattern 3	Horizontal lines
vcLargeCheckerboardPattern 2044	Checkerboard pattern showing squares of twice the size of vcSmallChecker-BoardPattern
vcLargeConfettiPattern 2029	Confetti pattern, large
vcLightDownwardDiagonalPattern 2012	Diagonal lines slanting to from top left to bottom right; spaced 50% closer than vcBDiagonalPattern
vcLightHorizontalPattern 2019	Horizontal lines spaced 50% closer than vcHorizontalPattern
vcLightUpwardDiagonalPattern 2013	Diagonal lines slanting from bottom left to top right, spaced 50% closer than vcBDiagonalPattern
vcLightVerticalPattern 2018	Vertical lines spaced 50% closer than vcVerticalPattern
vcNarrowHorizontalPattern 2021	Horizontal lines spaced 75 % closer than vcHorizontalPattern
vcNarrowVerticalPattern 2020	Vertical lines spaced 75% closer than vcVerticalPattern
vcNoPattern 1276	No fill pattern
vcOutlinedDiamondPattern 2045	Diagonal cross-hatch pattern, large
vcPlaidPattern 2035	Plaid pattern
vcShinglePattern 2039	Diagonal shingle pattern
vcSmallCheckerBoardPattern 2043	Checkerboard pattern
vcSmallConfettiPattern 2028	Confetti pattern

vcSmallGridPattern 2042	Cross-hatch pattern spaced 50% closer than vcCrossPattern 
vcSolidDiamondPattern 2046	Checkerboard pattern showing diagonal squares 
vcSpherePattern 2041	Checkerboard of spheres 
vcTrellisPattern 2040	Trellis pattern 
vcVerticalBottomLightedConvexPattern 43	Vertical color gradient from dark to bright 
vcVerticalConcavePattern 40	Vertical color gradient from dark to bright to dark 
vcVerticalConvexPattern 41	Vertical color gradient from bright to dark to bright 
vcVerticalGradientPattern 62	Vertical color gradient 
vcVerticalPattern 2	Vertical lines 
vcVerticalTopLightedConvexPattern 42	Vertical color gradient from bright to dark 
vcWavePattern 2031	Horizontal wave pattern 
vcWeavePattern 2034	Interwoven stripe pattern 
vcWideDownwardDiagonalPattern 2016	Diagonal lines slanting from top left to bottom right, showing the same spacing but three times the line width of vcF-DiagonalPattern 
vcWideUpwardDiagonalPattern 2017	Diagonal lines slanting from bottom left to top right right, showing the same spacing but three times the line width of vcBDiagonalPattern 
vcZigZagPattern 2030	Horizontal zig-zag lines 

RowPatternColorAsARGB

Property of VcNodeLevelLayout

This property lets you set or retrieve the pattern color of the node rows of this group level. Color values have a transparency or alpha value, followed by a value for a red, a blue and a green partition (ARGB). The values range between 0..255. An alpha value of 0 equals complete transparency, whereas 255 represents a completely solid color. When casting an RGB value on an ARGB value, an alpha value of 255 has to be added.

Also see **set/getRowBackColorAsARGB**.

If in the property **RowPatternColorMapName** a map is specified, the map will control the pattern color in dependence of the data.

	Data Type	Explanation
Property value	Color	ARGB color values ({0...255},{0...255},{0...255},{0...255})

RowPatternColorDataFieldIndex

Property of VcNodeLevelLayout

This property lets you set or retrieve the data field index that has to be specified if the property **RowPatternColorMapName** is used. If you set this property to **-1**, no map will be used.

	Data Type	Explanation
Property value	Integer **Possible Values:**	Data field index Data field index

RowPatternColorMapName

Property of VcNodeLevelLayout

This property lets you set or retrieve the name of a color map (type vcColorMap). If set to "", no map will be used. Only if a map name and a data field index are specified in the property **RowPatternColorDataFieldIndex**, the pattern color is controlled by the

map. If no data field entry applies, the pattern color of the group title row that is specified in the property **RowPatternColor** will be used.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the color map Name of the color map

RowPatternDataFieldIndex

Property of VcNodeLevelLayout

This property lets you set or retrieve the data field index to be used together with the property **RowPatternMapName**. If you set this property to **-1**, no map will be used.

	Data Type	Explanation
Property value	Integer **Possible Values:**	Data field index Data field index

RowPatternMapName

Property of VcNodeLevelLayout

This property lets you set or retrieve the name of a pattern map (type vcPatternMap). If set to "", no map will be used. Only if a map name and additionally a data field index are specified in the property **RowPatternDataFieldIndex**, the pattern is controlled by the map. If no data field entry applies, the pattern of the layer that is specified in the property **RowPattern** will be used.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the pattern map Name of the color map

SeparationLineColor

Property of VcNodeLevelLayout

This property lets you set or retrieve the color of the separation lines of the the grouping levels.

This property also can be set in the **Grouping** dialog, section **Nodes**, field **Separation Line**.

	Data Type	Explanation
Property value	Color	Color value ({0...255},{0...255},{0...255})

SeparationLineInterval

Read Only Property of VcNodeLevelLayout

This property lets you set or retrieve after how many activities a separating line is drawn.

	Data Type	Explanation

SeparationLineThickness

Read Only Property of VcNodeLevelLayout

This property lets you set or retrieve the line thickness of a separation line between nodes.

If you set this property to values between 1 and 4, an absolute line thickness is defined in pixels. Irrespective of the zoom factor a line will always show the same line thickness in pixels. When printing though, the line thickness is adapted for the sake of legibility and becomes dependent of the zoom factor:

Value	Points	mm
1	1/2 point	0.09 mm
2	1 point	0.18 mm
3	3/2 points	0.26 mm
4	2 points	0.35 mm

A point equals 1/72 inch and represents the unit of the font size.

If you set this property to values between 5 and 1,000, the line thickness is defined in 1/100 mm, so the lines will be displayed in a true thickness in pixels that depends on the zoom factor.

	Data Type	Explanation
Property value	Long	Line thickness LineType {1...4}: line thickness in pixels LineType {5...1000}: line thickness in 1/100 mm

SeparationLineType

Read Only Property of VcNodeLevelLayout

This property lets you set or retrieve the line type of a date line.

This property also can be set in the **Grouping** dialog, section **Nodes**, field **Separation Line**.

	Data Type	Explanation
Property value	LineTypeEnum	Type of separation lines of hierarchy levels
	Possible Values:	
	vcDashed 4	Line dashed
	vcDashedDotted 5	Line dashed-dotted
	vcDotted 3	Line dotted
	vcLineType0 100	Line Type 0
	vcLineType1 101	Line Type 1
	vcLineType10 110	Line Type 10
	vcLineType11 111	Line Type 11
	vcLineType12 112	Line Type 12
	vcLineType13 113	Line Type 13
	vcLineType14 114	Line Type 14
	vcLineType15 115	Line Type 15
	vcLineType16 116	Line Type 16
	vcLineType17 117	Line Type 17

vcLineType18 118	Line Type 18
vcLineType2 102	Line Type 2
vcLineType3 103	Line Type 3
vcLineType4 104	Line Type 4
vcLineType5 105	Line Type 5
vcLineType6 106	Line Type 6
vcLineType7 107	Line Type 7
vcLineType8 108	Line Type 8
vcLineType9 109	Line Type 9
vcNone 1	No line type
vcNotSet -1	No line type assigned
vcSolid 2	Line solid

ShowCalendarGrids

Read Only Property of VcNodeLevelLayout

This property lets you set or retrieve whether calendar grids are to be displayed.

This property also can be set in the **Nodes** section of the **Grouping** dialog.

	Data Type	Explanation
Property value	Boolean	Workfree periods are/are not accentuated
	Possible Values:	Group invisible/visible group nodes are/are not visible

ShowDateLines

Property of VcNodeLevelLayout

This property lets you set or retrieve whether date lines are to be displayed. This property also can be set in the **Grouping** dialog.

	Data Type	Explanation
Property value	Boolean	Date lines are/are not displayed.
	Possible Values:	Group invisible/visible

		group nodes are/are not visible

ShowSeparationLines

Read Only Property of VcNodeLevelLayout

This property lets you set or retrieve whether separation lines are to be displayed between the activities.

This property also can be set in the **Nodes** section of the **Grouping** dialog.

	Data Type	Explanation

ShowSeparationLinesAtTop

Read Only Property of VcNodeLevelLayout

This property lets you set or retrieve whether separation lines are to be displayed above activities (or below).

This property also can be set in the **Nodes** section of the **Grouping** dialog.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	Separation lines at top are displayed/not displayed Group invisible/visible group nodes are/are not visible

SortDataFieldIndex

Property of VcNodeLevelLayout

This property lets you set/retrieve the data field index used for sorting the nodes of this VcGroupLevelLayout object

	Data Type	Explanation
Parameter:		
⇨ sortlevel	Integer **Possible Values:**	Sorting level

		Data field index
Property value	Long	sorting field **Default value:** vcAscending

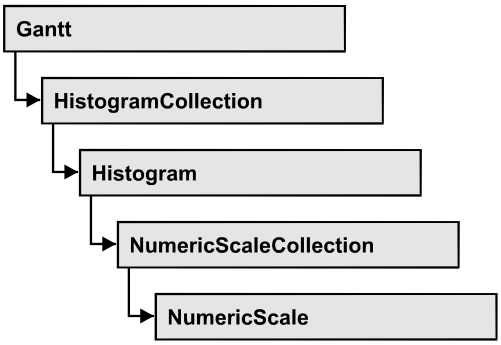
SortOrder

Property of VcNodeLevelLayout

This property lets you specify the sorting order of activities (ascending or descending). The property **SortDataFieldIndex** lets you specify the field the activties are sorted by. This property also can be set in the **Grouping** dialog.

	Data Type	**Explanation**
Parameter: ⇨ sortLevel	Integer **Possible Values:**	Sorting level Data field index
Property value	SortOrderEnum **Possible Values:** vcAscending 1 vcDescending 2	Ascending or descending order **Default value:** vcAscending ascending order descending order

7.66 VcNumericScale



An object of the type VcNumericScale is the scale of the vertical axis of a histogram.

Properties

- DoubleOutputFormat
- Font
- FontColor
- Histogram
- LineColor
- MajorTicks
- MajorTicksEx
- MinorTicks
- MinorTicksEx
- Name
- PatternBackgroundColorAsARGB
- PatternColorAsARGB
- PatternEx
- ThreeDEffect
- TickColor
- Title
- Unit
- UnitEx
- UnitLabel
- UnitWidth
- UpdateBehaviorName

Properties

DoubleOutputFormat

Property of VcNumericScale

This property lets you set or retrieve the output format of numbers as a double value in the numeric scale. The format is presented by the below characters:

- Text
- I
- D

plus the separators **comma** and **period**. **Text** represents a character string; **I** represents the figures before the decimal separator and **D** represents the figures after the decimal separator. The overall sequence is **Text I D Text**, where a comma and a period can be inserted in the places desired. As an example be the number -284901,3458. By the format **I,DDDD ppm** it will be output as **-284901,3458 ppm**. By the format **$I,III.DD** it will be output as **$-284,901.35**.

	Data Type	Explanation
Property value	String	Character string which describes the double format, for example "$I,III.DD".
	Possible Values:	Name of the color map

Example Code

```
VcGantt1.DoubleOutputFormat = "I,DDDD ppm"
```

Font

Property of VcNumericScale

This property lets you set or retrieve the font attributes of the numeric scale.

	Data Type	Explanation
Property value	StdFont	Font attributes of the numeric scale

Example Code

```
Dim font As StdFont
font = VcGantt1.NumericScaleCollection.Active.Font
```

FontColor

Property of VcNumericScale

This property lets you set or retrieve the font color of the numeric scale.

	Data Type	Explanation
Property value	Color	RGB color values **Default value:** RGB (0,0,0)

Example Code

```
Dim histogram As VcHistogram
Dim numericScale As VcNumericScale

Set histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
Set numericScale = histogram.NumericScaleCollection.Active

numericScale.FontColor = RGB(10, 10, 180)
```

Histogram

Read Only Property of VcNumericScale

This property lets you retrieve the histogram to which the numeric scale belongs.

	Data Type	Explanation
Property value	VcHistogram	Histogram object

Example Code

```
Private Sub VcGantt1_OnNumericScaleLClick(ByVal numericScale As
VcGanttLib.VcNumericScale, ByVal x As Long, ByVal y As Long, returnStatus As
Variant)

    MsgBox "Clicked on numeric scale of the histogram " &
numericScale.histogram.Name

End Sub
```

LineColor

Property of VcNumericScale

This property lets you set or retrieve the color of all border lines of the numeric scales of histograms.

If you set the color, it will be changed for the border lines of **all** numeric scales, retrieving will deliver the border line color of the **first** numeric scale .

	Data Type	Explanation
Property value	Color	RGB color values ({0...255},{0...255},{0...255})

MajorTicks

Property of VcNumericScale

This property lets you set or retrieve after how many units a major tick is drawn that has an annotation. You can also set the number of units in the **Edit Histogram** dialog. Also see **set/getMinorTicks** and **set/getMajorTicks**.

	Data Type	Explanation
Property value	Integer **Possible Values:**	Number of units between two major ticks {1...32767} Data field index

Example Code

```
Dim numericScaleCltn As VcNumericScaleCollection
Dim numericScale As VcNumericScale

Set numericScaleCltn = VcGantt1.NumericScaleCollection
Set numericScale = numericScaleCltn.Active
numericScale.MajorTicks = 4
```

MajorTicksEx

Property of VcNumericScale

This property lets you set or retrieve after how many units a major tick is drawn that has an annotation. Compared to the property **MajorTicks**, this property can be used to set floating point values. You can also set the number of units in the **Edit Histogram** dialog.

Also see **set/getMinorTicks**.

	Data Type	Explanation
Property value	Double	Number of units between two major ticks

Example Code

```
Dim numericScaleCltn As VcNumericScaleCollection
Dim numericScale As VcNumericScale

Set numericScaleCltn = VcGantt1.NumericScaleCollection
Set numericScale = numericScaleCltn.Active
numericScale.MajorTicksEx = 100 000
```

MinorTicks

Property of VcNumericScale

This property lets you set or retrieve after how many time units a minor tick without annotation is drawn. You can also set the number of the units in the **Edit Histogram** dialog. Also see **set/getMinorTicksEx** and **set/getMajorTicks**.

	Data Type	Explanation
Property value	Integer	Number of units between two minor ticks {1...32767}
	Possible Values:	Data field index

Example Code

```
Dim numericScaleCltn As VcNumericScaleCollection
Dim numericScale As VcNumericScale

Set numericScaleCltn = VcGantt1.NumericScaleCollection
Set numericScale = numericScaleCltn.Active
numericScale.MinorTicks = 2
```

MinorTicksEx

Property of VcNumericScale

This property lets you set or retrieve after how many time units a minor tick without annotation is drawn. Compared to the property **MinorTicks**, this property can be used to set floating point values. You can also set the number of the units in the **Edit Histogram** dialog. Also see **set/getMajorTicks**.

	Data Type	Explanation
Property value	Double	Number of units between two minor ticks {32 767...999 999}

Example Code

```
Dim numericScaleCltn As VcNumericScaleCollection
Dim numericScale As VcNumericScale

Set numericScaleCltn = VcGantt1.NumericScaleCollection
Set numericScale = numericScaleCltn.Active
```

```
numericScale.MinorTicks = 2
```

Name

Read Only Property of VcNumericScale

This property lets you retrieve the name of a numeric scale in a histogram. The name can be set in the **Edit Histogram** dialog.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the numeric scale Name of the color map

Example Code

```
Dim histogram As VcHistogram
Dim numericScale As VcNumericScale

Set histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
Set numericScale = histogram.NumericScaleCollection.Active

MsgBox "Active num. Scale: " & numericScale.Name
```

PatternBackgroundColorAsARGB

Property of VcNumericScale

This property lets you set or retrieve the background color of the numeric scale. Color values have a transparency or alpha value, followed by a value for a red, a blue and a green partition (ARGB). The values range between 0..255. An alpha value of 0 equals complete transparency, whereas 255 represents a completely solid color. When casting an RGB value on an ARGB value, an alpha value of 255 has to be added.

	Data Type	Explanation
Property value	Long	Background color of the box format **Default value:** -1

PatternColorAsARGB

Property of VcNumericScale

This property lets you set or retrieve the pattern color of the numeric scale. Color values have a transparency or alpha value, followed by a value for a red, a blue and a green partition (ARGB). The values range between 0..255.

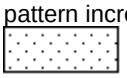
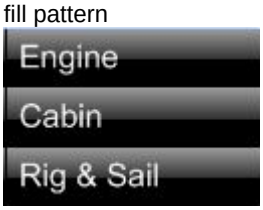
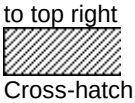
An alpha value of 0 equals complete transparency, whereas 255 represents a completely solid color. When casting an RGB value on an ARGB value, an alpha value of 255 has to be added.

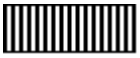
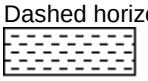
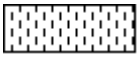
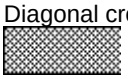
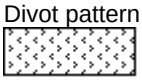
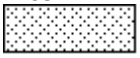
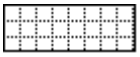
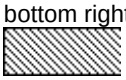
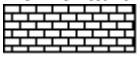
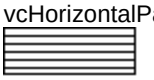
	Data Type	Explanation
Property value	Long	Pattern color of the line format field

PatternEx

Property of VcNumericScale

This property lets you set or retrieve the background pattern of the numeric scale.

	Data Type	Explanation
Property value	FillPatternEnum	Pattern type
	Possible Values: vc05PercentPattern... vc90PercentPattern 01 - 11	Dots in foreground color on background color, the density of the foreground pattern increasing with the percentage 
	vcAeroGlassPattern 40	Vertical color gradient in the color of the fill pattern 
	vcBDiagonalPattern 5	Diagonal lines slanting from bottom left to top right 
	vcCrossPattern 6	Cross-hatch pattern 
	vcDarkDownwardDiagonalPattern 2014	Diagonal lines slanting from top left to bottom right; spaced 50% closer than vcFDiagonalPattern and of twice the line width 
	vcDarkHorizontalPattern 2023	Horizontal lines spaced 50% closer than vcHorizontalPattern and of twice the line width 
	vcDarkUpwardDiagonalPattern 2015	Diagonal lines slanting from bottom left to top right, spaced 50% closer than vcBDiagonalPattern and of twice the line width 

vcDarkVerticalPattern 2022	Vertical lines spaced 50% closer than vcVerticalPattern and of of twice the line width 
vcDashedDownwardDiagonalPattern 2024	Dashed diagonal lines from top left to bottom right 
vcDashedHorizontalPattern 2026	Dashed horizontal lines 
vcDashedUpwardDiagonalPattern 2025	Dashed diagonal lines from bottom left to top right 
vcDashedVerticalPattern 2027	Dashed vertical lines 
vcDiagCrossPattern 7	Diagonal cross-hatch pattern, small 
vcDiagonalBrickPattern 2032	Diagonal brick pattern 
vcDivotPattern 2036	Divot pattern 
vcDottedDiamondPattern 2038	Diagonal cross-hatch pattern of dotted lines 
vcDottedGridPattern 2037	Cross-hatch pattern of dotted lines 
vcFDiagonalPattern 4	Diagonal lines slanting from top left to bottom right 
vcHorizontalBrickPattern 2033	Horizontal brick pattern 
vcHorizontalGradientPattern 52	Horizontal color gradient 
vcHorizontalPattern 3	Horizontal lines 
vcLargeCheckerboardPattern 2044	Checkerboard pattern showing squares of twice the size of vcSmallChecker-BoardPattern 
vcLargeConfettiPattern 2029	Confetti pattern, large 
vcLightDownwardDiagonalPattern 2012	Diagonal lines slanting to from top left to bottom right; spaced 50% closer than vcBDiagonalPattern 
vcLightHorizontalPattern 2019	Horizontal lines spaced 50% closer than vcHorizontalPattern 

vcLightUpwardDiagonalPattern 2013	Diagonal lines slanting from bottom left to top right, spaced 50% closer than vcBDiagonalPattern
vcLightVerticalPattern 2018	Vertical lines spaced 50% closer than vcVerticalPattern
vcNarrowHorizontalPattern 2021	Horizontal lines spaced 75 % closer than vcHorizontalPattern
vcNarrowVerticalPattern 2020	Vertical lines spaced 75% closer than vcVerticalPattern
vcNoPattern 1276	No fill pattern
vcOutlinedDiamondPattern 2045	Diagonal cross-hatch pattern, large
vcPlaidPattern 2035	Plaid pattern
vcShinglePattern 2039	Diagonal shingle pattern
vcSmallCheckerBoardPattern 2043	Checkerboard pattern
vcSmallConfettiPattern 2028	Confetti pattern
vcSmallGridPattern 2042	Cross-hatch pattern spaced 50% closer than vcCrossPattern
vcSolidDiamondPattern 2046	Checkerboard pattern showing diagonal squares
vcSpherePattern 2041	Checkerboard of spheres
vcTrellisPattern 2040	Trellis pattern
vcVerticalBottomLightedConvexPattern 43	Vertical color gradient from dark to bright
vcVerticalConcavePattern 40	Vertical color gradient from dark to bright to dark
vcVerticalConvexPattern 41	Vertical color gradient from bright to dark to bright
vcVerticalGradientPattern 62	Vertical color gradient
vcVerticalPattern 2	Vertical lines
vcVerticalTopLightedConvexPattern 42	Vertical color gradient from bright to dark

vcWavePattern 2031	Horizontal wave pattern
vcWeavePattern 2034	Interwoven stripe pattern
vcWideDownwardDiagonalPattern 2016	Diagonal lines slanting from top left to bottom right, showing the same spacing but three times the line width of vcF-DiagonalPattern
vcWideUpwardDiagonalPattern 2017	Diagonal lines slanting from bottom left to top right right, showing the same spacing but three times the line width of vcBDiagonalPattern
vcZigZagPattern 2030	Horizontal zig-zag lines

ThreeDEffect

Property of VcNumericScale

This property lets you set or retrieve whether the three-dimensional look of the numeric scale is switched on.

	Data Type	Explanation
Property value	Boolean	3D effect switched on (True)/switched off (False) **Default value:** False
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
Dim histogram As VcHistogram
Dim numericScale As VcNumericScale

Set histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
Set numericScale = histogram.NumericScaleCollection.Active

numericScale.ThreeDEffect = True
```

TickColor

Property of VcNumericScale

This property lets you set or retrieve the tick color for all numeric ribbons of histograms.

If you set the color, it will be changed for the border lines of **all** numeric ribbons, retrieving will deliver the tick color of the **first** numeric scale ribbon .

	Data Type	Explanation
Property value	Color RGB ({0...255},{0...255},{0...255})	RGB color values **Default value:** 0,0,0

Title

Property of VcNumericScale

This property lets you set or retrieve a title to/from the numeric scale. The ribbon that displays the title needs to be of the ribbon type **textual**. Scales and ribbons can be generated by the **Edit histogram** dialog box which can be invoked from the **Layout** property page.

	Data Type	Explanation
Parameter:		
⇨ Position	NumericAnnotationPositionEnum	Position of the title
	Possible Values:	
	vc10PercentFromTop 4	10 % of total scale length away from top
	vc30PercentFromTop 3	30 % of total scale length away from top
	vc50PercentFromTop 2	50 % of total scale length away from top
	vc70PercentFromTop 1	70 % of total scale length away from top
	vc90PercentFromTop 0	90 % of total scale length away from top
Property value	String	Title of the numeric scale
	Possible Values:	
		Name of the color map

Example Code

```
Dim numericScaleCltn As VcNumericScaleCollection
Dim activeNumericScale As VcNumericScale

Set numericScaleCltn = VcGantt1.HistogramCollection. _
                            HistogramByName("HISTOGRAM_1"). _
                            NumericScaleCollection
Set activeNumericScale = numericScaleCltn.Active

activeNumericScale.Title(vc90PercentFromTop) = "10 % occupied"
activeNumericScale.Title(vc70PercentFromTop) = "30 % occupied"
```

Unit

Property of VcNumericScale

This property lets you set or retrieve the units of the numeric scale. Also see **set/getUnitWidth**. This property also can be set in the **Edit Histogram** dialog.

	Data Type	Explanation
Property value	Integer	Unit
	Possible Values:	Data field index

UnitEx

Property of VcNumericScale

This property lets you set or retrieve the basic unit of the numeric scale as a double value.

	Data Type	Explanation
Property value	Double	Numbers in the double format {-999 999 .. 999 999}

Example Code

```
Private Sub CommandReduceRibbonUnit_Click()
    Dim numCol As VcNumericScaleCollection
    Dim numScale As VcNumericScale
    Set numCol =
VcGantt1.HistogramCollection.FirstHistogram.NumericScaleCollection
    Set numScale = numCol.FirstNumericScale
    numScale.UnitEx = numScale.UnitEx / 2#
    TextScaleUnit.Text = numScale.UnitEx
End Sub
```

UnitLabel

Property of VcNumericScale

This property lets you set or retrieve the designation of the units of the numeric scale. The designation is displayed centrally at the top border of the numeric scale.

	Data Type	Explanation
Property value	String	Designation of the unit
	Possible Values:	Name of the color map

Example Code

```
Dim numericScaleCltn As VcNumericScaleCollection
Dim activeNumericScale As VcNumericScale

Set numericScaleCltn = VcGantt1.HistogramCollection. _
                            HistogramByName("HISTOGRAM_1"). _
                            NumericScaleCollection

Set activeNumericScale = numericScaleCltn.Active
activeNumericScale.UnitLabel = "Hours"
```

UnitWidth

Property of VcNumericScale

This property lets you set or retrieve the width of the units of the numeric scale (by 1/100 mm). Also see **set/getUnit**. This property also can be set in the **Edit Histogram** dialog.

	Data Type	Explanation
Property value	Long	Unit width (by 1/100 mm)

Example Code

```
Dim numericScaleCltn As VcNumericScaleCollection
Dim activeNumericScale As VcNumericScale

Set numericScaleCltn = VcGantt1.HistogramCollection. _
                            HistogramByName("HISTOGRAM_1"). _
                            NumericScaleCollection

Set activeNumericScale = numericScaleCltn.Active
activeNumericScale.UnitWidth = 200
```

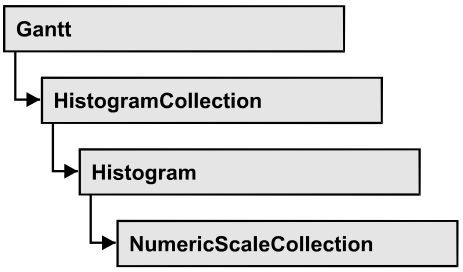
UpdateBehaviorName

Property of VcNumericScale

This property lets you set or retrieve the name of the UpdateBehavior.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the UpdateBehavior Name of the color map

7.67 VcNumericScaleCollection



An object of the type VcNumericScaleCollection automatically contains all available numeric scales. You can access all objects in an iterative loop by **For Each numericScale In NumericScaleCollection** or by the methods **First...** and **Next...**. You can access a single scales by using the methods **NumericScaleByName** and **NumericScaleByIndex**. The number of scales in the collection object can be retrieved by the property **Count**. By the property **Active** you can set or retrieve the scale that is presently active.

Properties

- _NewEnum
- Active
- Count

Methods

- FirstNumericScale
- NextNumericScale
- NumericScaleByIndex
- NumericScaleByName

Properties

_NewEnum

Read Only Property of VcNumericScaleCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all numeric scale objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method **GetEnumerator** is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim numscale As VcNumericScale

For Each numscale In VcGantt1.NumericScaleCollection
   Debug.Print numscale.Name
Next
```

Active

Property of VcNumericScaleCollection

This method lets you set or retrieve the numeric scale currently displayed in the diagram.

	Data Type	Explanation
Property value	VcNumericScale	Numeric scale currently used

Example Code

```
Dim numericScaleCltn As VcNumericScaleCollection
Dim activeNumericScale As VcNumericScale

Set numericScaleCltn = VcGantt1.HistogramCollection. _
                            HistogramByName("HISTOGRAM_1"). _
                            NumericScaleCollection
Set activeNumericScale = numericScaleCltn.Active
```

Count

Property of VcNumericScaleCollection

This property lets you retrieve the number of numeric scales in the NumericScaleCollection object.

	Data Type	Explanation
Property value	Long	Number of numeric scales

Example Code

```
Dim numericScaleCltn As VcNumericScaleCollection
Dim numberOfNumericScales As Long

Set numericScaleCltn = VcGantt1.HistogramCollection. _
                            HistogramByName("HISTOGRAM_1"). _
                            NumericScaleCollection

numberOfNumericScales = numericScaleCltn.Count
```

Methods

FirstNumericScale

Method of VcNumericScaleCollection

This method can be used to access the initial value, i.e. the first numeric scale of a numeric scale collection, and then to continue in a forward iteration loop by the method **NextNumericScale** for the scales following. If there is no scale in the numeric scale collection, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcNumericScale	First numeric scale

Example Code

```
Dim numericScaleCltn As VcNumericScaleCollection
Dim numericScale As VcNumericScale

Set numericScaleCltn = VcGantt1.HistogramCollection. _
                            HistogramByName("HISTOGRAM_1"). _
                            NumericScaleCollection
Set numericScale = numericScaleCltn.FirstNumericScale
```

NextNumericScale

Method of VcNumericScaleCollection

This method can be used in a forward iteration loop to retrieve subsequent numeric scales from a numeric scale collection after initializing the loop by the method **FirstNumericScale**. If there is no numeric scale left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcNumericScale	Subsequent numeric scale

Example Code

```
Dim numericScaleCltn As VcNumericScaleCollection
Dim numericScale As VcNumericScale

Set numericScaleCltn = VcGantt1.HistogramCollection. _
                            HistogramByName("HISTOGRAM_1"). _
                            NumericScaleCollection
Set numericScale = numericScaleCltn.FirstNumericScale
While Not numericScale Is Nothing
    List1.AddItem numericScale.Name
    Set numericScale = numericScaleCltn.NextNumericScale
Wend
```

NumericScaleByIndex

Method of VcNumericScaleCollection

This method lets you access a numeric scale by its index. If a numeric scale of the specified index does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter: ⇨ index	Integer **Possible Values:**	Index of the numeric scale Data field index
Return value	VcNumericScale	Numeric scale object returned

NumericScaleByName

Method of VcNumericScaleCollection

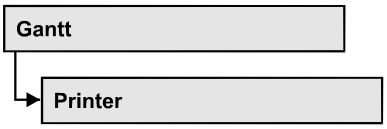
By this method you can retrieve a numeric scale by its name. If a numeric scale of the specified name does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter: ⇨ numericScaleName	String **Possible Values:**	Name of the numeric scale Name of the color map
Return value	VcNumericScale	Numeric scale

Example Code

```
Dim numericScaleCltn As VcNumericScaleCollection

Set numericScaleCltn = VcGantt1.HistogramCollection. _
                            HistogramByName("HISTOGRAM_1"). _
                            NumericScaleCollection
numericScaleCltn.Active = numericScaleCltn. _
                            NumericScaleByName("STEP1")
```

7.68 VcPrinter



The VcPrinter object offers a variety of properties to set up the printing process. You can enter the width of top, bottom, left and right margins, set a page frame, page numbers, a page description, cutting marks and the print date. Beside, you can specify the number of pages that the diagram is to be printed on. Zoom factor, alignment, orientation, paper size and color mode are more properties that you can vary for a perfect print.

Properties

- AbsoluteBottomMarginInCM
- AbsoluteBottomMarginInInches
- AbsoluteLeftMarginInCM
- AbsoluteLeftMarginInInches
- AbsoluteRightMarginInCM
- AbsoluteRightMarginInInches
- AbsoluteTopMarginInCM
- AbsoluteTopMarginInInches
- Alignment
- AllBorderBoxesShownOnCombinedControls
- CombiningControlsEnabled
- CurrentHorizontalPagesCount
- CurrentVerticalPagesCount
- CurrentZoomFactor
- CuttingMarks
- DateFormat
- DefaultPrinterName
- DiagramEnabled
- DocumentName
- FoldingMarksType
- MarginsShownInInches
- MaxHorizontalPagesCount
- MaxVerticalPagesCount
- Orientation
- PageDescription
- PageDescriptionString

- PageFrame
- PageNumberMode
- PageNumbers
- PagePaddingEnabled
- PaperSize
- PrintDate
- PrinterName
- ReOptimizeNodesInGroupsEnabled
- RepeatTableTimeScale
- ScalingMode
- StartUpSinglePage
- TableColumnRanges
- TableWidthAdoptionFromViewOnScreen
- TimeColumnEndDate
- TimeColumnStartDate
- TimeScaleAdjustment
- ZoomFactorAsDouble

Properties

AbsoluteBottomMarginInCM

Property of VcPrinter

This property lets you set or retrieve the absolute height of the bottom margin of the pages to be printed. The true width may be larger if the printer used has to print margins by obligation.

	Data Type	Explanation
Property value	Double	Height of the bottom margin of the page in cm **Default value:** 0

Example Code

```
VcGantt1.Printer.AbsoluteBottomMarginInCM = 1.5
```

AbsoluteBottomMarginInInches

Property of VcPrinter

This property lets you set or retrieve the absolute height of the bottom margin of the pages to be printed in inches. The true width may be larger if the printer used has to print margins by obligation.

Tip: The internal conversion factor is 2.5 cm/inch instead of the actual correct 2.54 cm/inch so that the values shown in the **Page Setup** dialog will be smoother (1.5 cm so add up to 0.6 inches, 1 cm add up to 0.4 inches).

	Data Type	Explanation
Property value	Double	Height of the bottom margin of the page in inches **Default value:** 0

Example Code

```
VcGantt1.Printer.AbsoluteBottomMarginInches = 0.5
```

AbsoluteLeftMarginInCM

Property of VcPrinter

This property lets you set or retrieve the absolute width of the left margin of the pages to be printed. The true width may be larger if the printer used has to print margins by obligation.

	Data Type	Explanation
Property value	Double	Width of the left margin of the page in cm **Default value:** 0

Example Code

```
VcGantt1.Printer.AbsoluteLeftMarginInCM = 1.5
```

AbsoluteLeftMarginInInches

Property of VcPrinter

This property lets you set or retrieve the absolute width of the left margin of the pages to be printed in inches. The true width may be larger if the printer used has to print margins by obligation.

Tip: The internal conversion factor is 2.5 cm/inch instead of the actual correct 2.54 cm/inch so that the values shown in the **Page Setup** dialog will be smoother (1.5 cm so add up to 0.6 inches, 1 cm add up to 0.4 inches).

	Data Type	Explanation
Property value	Double	Width of the left margin of the page in inches **Default value:** 0

Example Code

```
VcGantt1.Printer.AbsoluteLeftMarginInInches = 0.5
```

AbsoluteRightMarginInCM

Property of VcPrinter

This property lets you set or retrieve the absolute width of the right margin of the pages to be printed. The true width may be larger if the printer used has to print margins by obligation.

	Data Type	Explanation
Property value	Double	Width of the right margin of the page in cm **Default value:** 0

Example Code

```
VcGantt1.Printer.AbsoluteRightMarginInCM = 1.5
```

AbsoluteRightMarginInInches

Property of VcPrinter

This property lets you set or retrieve the absolute width of the right margin of the pages to be printed in inches. The true width may be larger if the printer used has to print margins by obligation.

Tip: The internal conversion factor is 2.5 cm/inch instead of the actual correct 2.54 cm/inch so that the values shown in the **Page Setup** dialog will be smoother (1.5 cm so add up to 0.6 inches, 1 cm add up to 0.4 inches).

	Data Type	Explanation
Property value	Double	Width of the right margin of the page in inches **Default value:** 0

Example Code

```
VcGantt1.Printer.AbsoluteRightMarginInInches = 0.5
```

AbsoluteTopMarginInCM

Property of VcPrinter

This property lets you set or retrieve the absolute height of the top margin of the pages to be printed. The true width may be larger if the printer used has to print margins by obligation.

	Data Type	Explanation
Property value	Double	Height of the top margin of the page in cm **Default value:** 0

Example Code

```
VcGantt1.Printer.AbsoluteTopMarginInCM = 1.5
```

AbsoluteTopMarginInInches

Property of VcPrinter

This property lets you set or retrieve the absolute height of the top margin of the pages to be printed in inches. The true width may be larger if the printer used has to print margins by obligation.

Tip: The internal conversion factor is 2.5 cm/inch instead of the actual correct 2.54 cm/inch so that the values shown in the **Page Setup** dialog will be smoother (1.5 cm add up to 0.6 inches, 1 cm add up to 0.4 inches).

	Data Type	Explanation
Property value	Double	Height of the top margin of the page in inches **Default value:** 0

Example Code

```
VcGantt1.Printer.AbsoluteTopMarginInInches = 0.5
```

Alignment

Property of VcPrinter

This property lets you set or retrieve the alignment of the diagram on a page. The property will be effective either if the diagram is put out onto a single page or if the **RepeatTableTimeScale** property was set. In any other case the output will be centered.

	Data Type	Explanation
Property value	PrinterAlignmentEnum	Alignment of the output with its sheet **Default value:** vcPCenterCenter
	Possible Values:	
	vcPBottomCenter 28	Vertical alignment: bottom; horizontal alignment: center
	vcPBottomLeft 27	Vertical alignment: bottom; horizontal alignment: left
	vcPBottomRight 29	Vertical alignment: bottom; horizontal alignment: right
	vcPCenterCenter 25	Vertical alignment: center; horizontal alignment: center
	vcPCenterLeft 24	Vertical alignment: center; horizontal alignment: left
	vcPCenterRight 26	Vertical alignment: center; horizontal alignment: right
	vcPTopCenter 22	Vertical alignment: top; horizontal alignment: center
	vcPTopLeft 21	Vertical alignment: top; horizontal alignment: left
	vcPTopRight 23	Vertical alignment: top; horizontal alignment: right

Example Code

```
VcGantt1.Printer.Alignment = vcPTopLeft
```

AllBorderBoxesShownOnCombinedControls

Property of VcPrinter

If this property is set to "True" all border boxes are printed even if combined printing is activated. If it is set to "False", the border boxes are ignored. See the objects **VcBorderArea** and <!VcBorderBox.

	Data Type	Explanation
Property value	Boolean	Border boxes are (True)/are not (False) printed if combined printing is enabled
	Possible Values:	
		Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.AllBorderBoxesShownOnCombinedControls = True
```

CombiningControlsEnabled

Property of VcPrinter

If this property is set to **True**, all XGantt controls of the parent window are arranged one below the other according to their relative vertical position for exporting or printing and in the print preview. Thus more than one diagram can be displayed at once.

Tip: When this feature is used, the properties **RepeatTableTimeScale** and **TimeScaleAdjustment** will be ignored and their value assumed as "False". Likewise, the property **VcPrinter.FoldingMarksType** will be ignored and its value assumed as "vcFMTNone" .

	Data Type	Explanation
Property value	Boolean **Possible Values:**	XGantt controls of the parent window are (True) / are not (False) arranged one below the other **Default value:** False Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.Printer.CombiningControlsEnabled = True
```

CurrentHorizontalPagesCount

Read Only Property of VcPrinter

This property lets you retrieve the actual number of pages in horizontal direction onto which the chart is to be printed. Also see **CurrentVertical-PagesCount** and **MaxHorizontalPagesCount**.

	Data Type	Explanation
Property value	Long	Current number of pages counted in horizontal direction

CurrentVerticalPagesCount

Read Only Property of VcPrinter

This property lets you retrieve the actual number of pages in vertical direction onto which the chart is to be printed. Also see **CurrentHorizontal-PagesCount** and **MaxVerticalPagesCount**.

	Data Type	Explanation
Property value	Long	Current number of pages counted in vertical direction

CurrentZoomFactor

Read Only Property of VcPrinter

This property lets you retrieve the actual zoom factor for the scaling mode**vcFitToPageCount** (zoom factor = 100: original size, zoom factor > 100: enlargement, zoom factor < 100: reduction).

	Data Type	Explanation
Property value	Double	Actual zoom factor

CuttingMarks

Property of VcPrinter

This property lets you set or retrieve, whether (True) or not (False) cutting marks are to printed onto a page.

	Data Type	Explanation
Property value	Boolean	Cutting marks are (True) / are not (False) printed **Default value:** False
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.Printer.CuttingMarks = True
```

DateFormat

Property of VcPrinter

This property lets you set the date format that is to be used in the DatePicker dialog elements of the **Page Layout** dialog. The empty string represents the default date format TS. To compose the date you can use the below tokens:

D: first letter of the day of the week (not adjustable)

TD: Day of the Week (adjustable by using the event **VcTextEntrySupplying**)

DD: two-digit figure for the day of the month: 01-31

DDD: first three letters of the day of the week (not adjustable)

M: first letter of the name of the month (not adjustable)

TM:	name of the month (adjustable by using the event **VcTextEntrySupplying**)
MM:	two-digit figure for the month: 01-12
MMM:	first three letters of the name of the month (not adjustable)
YY:	two-digit figure for the year
YYYY:	four-digit figure for the year
WW:	two-digit figure for the number of the calendar week: 01-53
TW:	text for "calendar week" (adjustable by using the event **VcTextEntrySupplying**)
Q:	one-digit figure for the quarter: 1-4
TQ:	name of quarter (adjustable by using the event **VcTextEntrySupplying**)
hh	two-digit figure for the hour in 24 hours format: 00-23
HH:	two-digit figure for the hour in 12 hours format: 01-12
Th:	Text of "o' clock" (adjustable by using the event **VcTextEntrySupplying**)
TH:	"am" or "pm" (adjustable by using the event **VcTextEntrySupplying**)
mm	two-digit figure for the minute: 00-59
ss:	two-digit figure for the second: 00-59
TS:	short date format, as defined in the regional settings of the windows control panel
TL:	long date format, as defined in the regional settings of the windows control panel
TT:	time format, as defined in the regional settings of the windows control panel

Note: Characters which are not to be interpreted as part of the date should be preceded by a backslash '\'. '\\' for instance results in "\'. The special characters: **':, /, -**' and **blank** don't need '\' as prefix.

	Data Type	Explanation
Property value	String **Possible Values:**	Date format in Page Layout dialog **Default value:** " " Name of the color map

DefaultPrinterName

Read Only Property of VcPrinter

This property lets you return the current name of the system's current default printer.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of current default printer Name of the color map

DiagramEnabled

Property of VcPrinter

This property lets you specify whether the diagram (time scale and layers) shall be printed or not.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	Diagram is (True) / is not (False) printed **Default value:** True Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.Printer.DiagramEnabled = True
```

DocumentName

Property of VcPrinter

This property lets you set or enquire the name of the document. When printing, the document name is displayed in the list of the documents to print

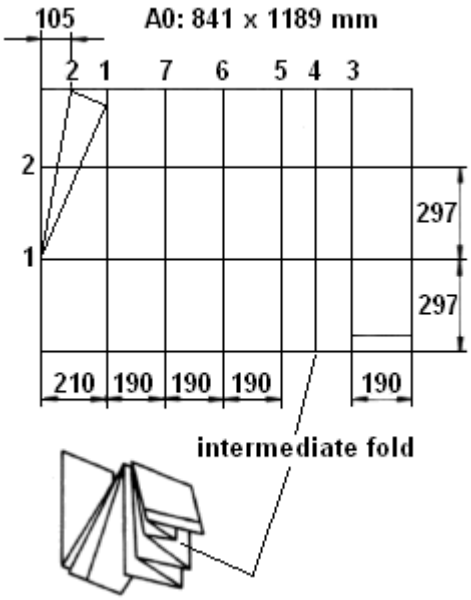
and has special functions with certain printer drivers as e.g. drivers which create PDF files.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of document **Default value:** " " Name of the color map

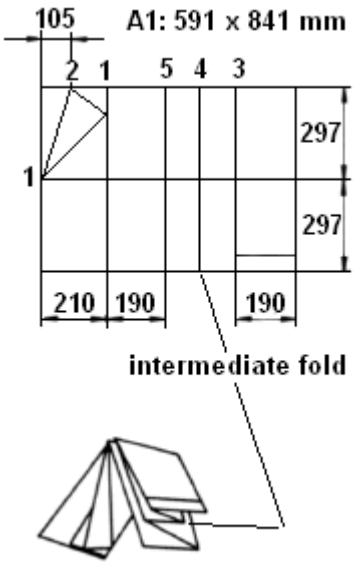
FoldingMarksType

Read Only Property of VcPrinter

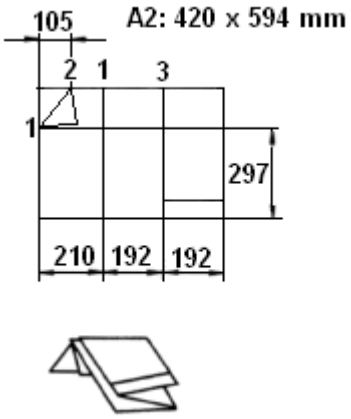
This property lets you set or retrieve the following folding marks according to DIN 824. The folding marks allow to fold paper sheets of the German DIN-A standard:



Folding of the DIN-A-0 format

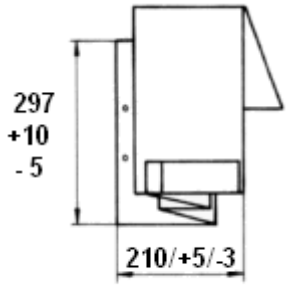
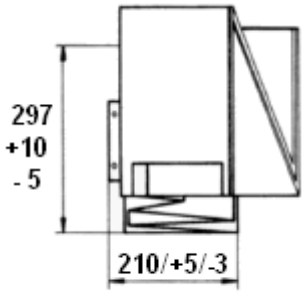
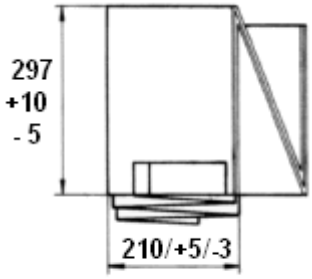


Folding of the DIN-A-1 format



Folding of the DIN-A-2 format

	Data Type	Explanation
Property value	FoldingMarksTypeEnum **Possible Values:**	Folding marks **Default value:** vcFMTNone

vcFMTDIN824FormA 65	Folding marks according to DIN824-A: the drawing can be punched and filed directly to a folder.  DIN 824-A way of folding
vcFMTDIN824FormB 66	Folding marks according to DIN824-B: the chart can be punched and filed to a folder by a flexi filing fastener.  DIN 824-B way of folding
vcFMTDIN824FormC 67	Folding marks according to DIN824-C: the folded chart is not to be punched but to be put in a sheet protector.  DIN 824-C way of folding
vcFMTNone 0	No folding marks

MarginsShownInInches

Property of VcPrinter

This property lets you set or retrieve whether the measuring unit of the margins in the <b"Page Layout dialog shall be switched to inches (at present only possible at runtime).

Tip: The internal conversion factor is 2.5 cm/inch instead of the actual correct 2.54 cm/inch so that the values shown in the **Page Setup** dialog will be smoother (1.5 cm so add up to 0.6 inches, 1 cm add up to 0.4 inches).

	Data Type	Explanation
Property value	Boolean	Measuring unit of the margins in the **Page Layout** dialog in inches (True)/ in cm (False) **Default value:** False
	Possible Values:	Group invisible/visible group nodes are/are not visible

MaxHorizontalPagesCount

Property of VcPrinter

This property lets you set or retrieve the horizontal number of pages für printing and for the print preview. This property only works if the property **ScalingMode** was set to either **vcFitToPageCount** or to **vcZoomWithHorizontalFit**. Also see **MaxVerticalPagesCount** and **CurrentHorizontalPagesCount**.

	Data Type	Explanation
Property value	Long	Maximum number of pages counted in horizontal direction **Default value:** 1

Example Code

```
VcGantt1.Printer.MaxHorizontalPagesCount = 4
```

MaxVerticalPagesCount

Property of VcPrinter

This property lets you set or retrieve the vertical number of pages für printing and for the print preview. This property only works if the property **ScalingMode** was set to **vcFitToPageCount**. Also see **MaxHorizontalPagesCount** and **CurrentVerticalPagesCount**.

	Data Type	Explanation
Property value	Long	Maximum number of pages counted in vertical direction **Default value:** 1

Example Code

```
VcGantt1.Printer.MaxVerticalPagesCount = 4
```

Orientation

Property of VcPrinter

This property lets you set or retrieve the orientation of the output.

	Data Type	Explanation
Property value	OrientationEnum	Orientation **Default value:** VcPortrait
	Possible Values: vcLandscape 42 vcPortrait 41	 Printing orientation **landscape** Printing orientation **portrait**

Example Code

```
VcGantt1.Printer.Orientation = vcLandScape
```

PageDescription

Property of VcPrinter

This property lets you set or retrieve whether (True) or not (False) the page description string is to appear in the bottom left corner of a page. The contents of the page description string you can set by the **PageDescriptionString** property.

	Data Type	Explanation
Property value	Boolean	Page description is (True) / is not (False) printed **Default value:** False
	Possible Values:	 Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.Printer.PageDescription = True
```

PageDescriptionString

Property of VcPrinter

This property lets you set or retrieve a page description string in the bottom left corner of each page. Whether or not the page description string is printed you can control by the **PageDescription** property. For numbering the pages you may enter the following place holders which will be replaced with the appropriate contents on the printout:

{PAGE} = consecutive numbering of pages

{NUMPAGES} = total number of pages

{ROW} = line position of the section in the complete chart

{COLUMN} = column position of the section in the complete chart

	Data Type	Explanation
Property value	String	Page description **Default value:** Empty string ""
	Possible Values:	Name of the color map

Example Code

```
VcGantt1.Printer.PageDescriptionString = "VARCHART chart"
```

PageFrame

Property of VcPrinter

This property lets you set or retrieve, whether (True) or not (False) a frame is to be drawn around the output. If the **RepeatTableTimeScale** property was set, the frame will be drawn around the part on each page, otherwise it will be drawn around the diagram as a whole.

	Data Type	Explanation
Property value	Boolean	Frame is (True) / is not (False) displayed **Default value:** True
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.Printer.PageFrame = True
```

PageNumberMode

Property of VcPrinter

This property lets you set or retrieve in which way the page numbers are to be displayed: "Page N of M pages" or "x.y" (row no./column no.).

	Data Type	Explanation
Property value	pageNumberModeEnum	mode of page numbering **Default value:** vcPRowColumn
	Possible Values: vcPageNOfM 1597 vcPRowColumn 1596	 "Page N of M pages" "x.y" (row no./column no.).

Example Code

```
Dim printer As VcPrinter

Set printer = VcGantt1.printer

With printer
    .Orientation = vcLandscape
    .PageNumberMode = vcPageNOfM
    .PageNumbers = True
    .FitToPage = False
End With

VcGantt1.PrintPreview
```

PageNumbers

Property of VcPrinter

This property lets you set or retrieve, whether (True) or not (False) a page number is printed. The mode of page numbering is set with the help of the property **PageNumberMode**.

	Data Type	Explanation
Property value	Boolean	Page numbers are (True) / are not (False) printed **Default value:** False
	Possible Values:	 Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.Printer.PageNumbers = True
```

PagePaddingEnabled

Property of VcPrinter

This property lets you specify or retrieve whether enough space is to be left between the diagram and the boxes of the title and legend area so that the boxes are always printed in full width and are attached to the margin. If the property is set to **False** there will be no space left between the diagram and the boxes and their width may vary on the different pages depending on the diagram.

	Data Type	Explanation
Property value	Boolean	Space between diagram and boxes for legend/title is (True) / is not (False) left **Default value:** True
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.Printer.PagePaddingEnabled = True
```

PaperSize

Property of VcPrinter

This property lets you set or retrieve the paper size to be used.

	Data Type	Explanation
Property value	PaperSizeEnum	Paper size
	Possible Values:	
	vcDIN_A2 66	DIN A2
	vcDIN_A3 8	DIN A3
	vcDIN_A4 9	DIN A4
	vcISO_C 24	ISO C
	vcISO_D 25	ISO D
	vcISO_E 26	ISO E
	vcUS_LEGAL 5	US LEGAL
	vcUS_LETTER 1	US LETTER

Example Code

```
VcGantt1.Printer.PaperSize = vcDIN_A3
```

PrintDate

Property of VcPrinter

This property lets you set or retrieve, whether (True) or not (False) the print date is to appear in the bottom left corner of a page.

	Data Type	Explanation
Property value	Boolean	Print date is/is not set
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.Printer.PrintDate = True
```

PrinterName

Read Only Property of VcPrinter

This property lets you set or retrieve the name of the currently selected printer. You can use this property for saving and restoring the state of the printer object.

If you transfer an empty string when setting the property, the system printer will be used.

<Tip:> Please note that the name of network printers has to be written in UNC notation, e.g. "\\server01\printer5".

	Data Type	Explanation
Property value	**String**	**Printer name**
	Possible Values:	**Name of the color map**

ReOptimizeNodesInGroupsEnabled

Property of VcPrinter

If the property TimeScaleAdjustment was set to true, this property allows to automatically update for the output or for the print preview the optimized arrangement of groups that are in the optimized state of display. This is only necessary if there are layers with text on the outside. The automatic

optimization is very time-consuming and may lead to high response times in the print preview.

	Data Type	Explanation
Property value	Boolean	With the **TimeScaleAdjustment** property switched on: optimized groups are (True)/are not(False) reoptimized for output or print preview **Default value:** False
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.Printer.ReOptimizeNodesInGroupsEnabled = True
```

RepeatTableTimeScale

Property of VcPrinter

This property lets you set or retrieve, whether (True) or not (False) the title, legend, table and time scale are to appear on each page.

	Data Type	Explanation
Property value	Boolean	Title, legend, table and time scale are repeated on each page (True)/ Title, legend, table and time scale are output only once and cut if necessary (False) **Default value:** False
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.Printer.RepeatTableTimeScale = True
```

ScalingMode

Property of VcPrinter

This property lets you set or retrieve the scaling mode for output. If the scaling mode is set to **vcZoomFactor**, the value of the property **ZoomFactor** defines the size of the output. If set to **vcFitToPageCount**, the values of **MaxHorizontalPagesCount** and **MaxVerticalPagesCount** are essential. If set to **vcZoomWithHorizontalFit**, the values of **ZoomFactor** and **Max-HorizontalPagesCount** define a zoom factor providing a fixed number of pages in width. The number of pages is maintained by downsizing or

expanding the time scale. When using **vcZoomFactor** or **vcFitToPage-Count**, you can achieve at covering the pages evenly by the property **AdjustTimeScale**.

	Data Type	Explanation
Property value	ScalingModeEnum	scaling mode for output
	Possible Values: vcFitToPageCount 1 vcZoomFactor 0 vcZoomWithHorizontalFit 2	 Scaling mode "Fit to Page" Scaling mode: "Zoomfactor". Scaling mode "Combined Fit"

StartUpSinglePage

Property of VcPrinter

This property lets you set or retrieve the mode of starting the page preview: either all pages of the diagram will be displayed (False) or only the first page will be displayed (True).

	Data Type	Explanation
Property value	Boolean	at the start of the page preview: only first page of the diagram (True)/ all pages of the diagram (False)
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
Dim printer As VcPrinter

Set printer = VcGantt1.printer

With printer
    .Orientation = vcLandscape
    .StartUpSinglePage = True
    .FitToPage = False
End With

VcGantt1.PrintPreview
```

TableColumnRanges

Property of VcPrinter

This property lets you set the number of table columns to be printed. Similar to Microsoft Word you can specify single columns or ranges of columns, that are to be separated by comas or semicolons. Example: "1;5-7;3" specifies the columns 1 and 3 and the range from 5 to 7. "0", a simple comma or

semicolon will result in no column printed. By setting the default value -1 you can have all columns printed.

	Data Type	Explanation
Property value	String	Number of table columns which are printed **Default value:** empty string
	Possible Values:	Name of the color map

Example Code

```
VcGantt1.Printer.TableColumnRange = "1;5-7;3"
```

TableWidthAdoptionFromViewOnScreen

Property of VcPrinter

This property lets you specify or retrieve whether the table width that is currently shown on the screen is to be adopted for the print preview and for the output.

This property can be also set in the **Page Layout** dialog.

	Data Type	Explanation
Property value	Boolean	the table width that is currently shown on the screen is (True) / is not (False) to be adopted for the print preview and for the output **Default value:** False
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.Printer.TableWidthAdoptionFromViewOnScreen = True
```

TimeColumnEndDate

Property of VcPrinter

This property lets you set or retrieve the end date of the time range to be used for the output. The time range can only be restriced in comparison to the time range displayed on the screen. Hence only an earlier end date than that having been set by the VcGantt property **TimeScaleStart** leads to a modified output.

This property can be also set in the **Page Layout** dialog.

	Data Type	Explanation
Property value	DateTime	End date of the time range for the output **Default value:** 0.0

TimeColumnStartDate

Property of VcPrinter

This property lets you set or retrieve the start date of the time range to be used for the output. The time range can only be restriced in comparison to the time range displayed on the screen. Hence only a later start date than that having been set by the VcGantt property **TimeScaleStart** leads to a modified output.

This property can be also set in the **Page Layout** dialog.

	Data Type	Explanation
Property value	DateTime	Start date of the time range for the output **Default value:** 0.0

TimeScaleAdjustment

Property of VcPrinter

This property leads to a better utilization of the printing pages:

- If scaling fit to page is selected: The zoom factor is calculated so that the space of the selected number of pages is fully used for printing into the height while the time scale gets downsized or enlarged so that the selected number of pages is used to full capacity into the width.

- If **scaling by zoom factor** was selected: The time scale is downsized or enlarged to equal the total width of the selected number of pages.

	Data Type	Explanation
Property value	TimeScaleAdjustment	Adjustment of time scale **Default value:** False

Example Code

```
VcGantt1.Printer.TimeScaleAdjustment = True
```

ZoomFactorAsDouble

Property of VcPrinter

This property lets you set or retrieve the zoom factor for the scaling modes**VcZoomFactor** and. **vcZoomWithHorizontalFit** to enlarge or downsize the output (zoom factor = 100: original size, zoom factor > 100: enlargement, zoom factor < 100: reduction).

	Data Type	Explanation
Property value	Double	Zoom factor of the diagram **Default value:** 100

Example Code

```
VcGantt1.Printer.ZoomFactorAsDouble = 150
```

7.69 VcRect

Rect

An object of the type **VcRect** designates a rectangle object and is only passed by the event VcGantt.OnShowInPlaceEditor.

Properties

- Bottom
- Height
- Left
- Right
- Top
- Width

Properties

Bottom

Property of VcRect

This property returns/sets the bottom coordinate of the VcRect object.

	Data Type	Explanation
Property value	Long	Position of the bottom border of the rectangle

Height

Read Only Property of VcRect

This property returns the height of the VcRect object.

	Data Type	Explanation
Property value	Long	Height of the rectangle

Left

Property of VcRect

This property returns/sets the left coordinate of the VcRect object.

	Data Type	Explanation
Property value	Long	Position of the left border of the rectangle

Example Code

```
Private Sub VcGantt1_OnShowInPlaceEditor(ByVal editObject As Object, _
                          ByVal editObjectType As _
                          VcGanttLib.VcObjectTypeEnum, _
                          ByVal fieldIndex As Long, ByVal objRectComplete As _
                          VcGanttLib.VcRect, ByVal objRectVisible As _
                          VcGanttLib.VcRect, ByVal fldRectComplete As _
                          VcGanttLib.VcRect, ByVal fldRectVisible As _
                          VcGanttLib.VcRect, returnStatus As Variant)

   Dim oldScaleMode As Long

   If editObjectType = vcObjTypeNodeInTable Then
      returnStatus = vcRetStatFalse

      Set myEditObject = editObject
      myEditObjectType = editObjectType
      myEditObjectFieldIndex = fieldIndex

      oldScaleMode = Me.ScaleMode
      Me.ScaleMode = vbPixels

      Select Case fieldIndex
         Case 1   'Name
            Text1.Left = fldRectVisible.Left + VcGantt1.Left
            Text1.Top = fldRectVisible.Top + VcGantt1.Top
            Text1.Width = fldRectVisible.Width
            Text1.Height = fldRectVisible.Height

            Text1.Text = editObject.DataField(fieldIndex)
            Text1.Visible = True
            Text1.SetFocus

         Case 2, 3     'Start or End
            MonthView1.Left = fldRectVisible.Left + VcGantt1.Left
            MonthView1.Top = fldRectVisible.Top + VcGantt1.Top

            MonthView1.Value = editObject.DataField(fieldIndex)
            MonthView1.Visible = True
            MonthView1.SetFocus

         Case 13   'Employee
            Combo1.Left = fldRectVisible.Left + VcGantt1.Left
            Combo1.Top = fldRectVisible.Top + VcGantt1.Top
            Combo1.Width = fldRectVisible.Width

            Combo1.Text = editObject.DataField(fieldIndex)
            Combo1.Visible = True
            Combo1.SetFocus

      End Select

      Me.ScaleMode = oldScaleMode
```

```
    End If
End Sub
```

Right

Property of VcRect

This property returns/sets the right coordinate of the VcRect object.

	Data Type	Explanation
Property value	Long	position of the right border of the rectangle

Top

Property of VcRect

This property returns/sets the top coordinate of the VcRect object.

	Data Type	Explanation
Property value	Long	position of the top border of the rectangle

Example Code

```
MonthView1.Top = fldRectVisible.Top + VcGantt1.Top
```

Width

Read Only Property of VcRect

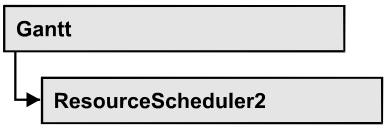
This property returns the width of the VcRect object.

	Data Type	Explanation
Property value	Long	width of the rectangle

Example Code

```
Text1.Width = fldRectVisible.Width
```

7.70 VcResourceScheduler2



The ResourceScheduler2 is a substantial enhancement of Resource-Scheduler1 (version 3.1). The different object types required for resource scheduling are now anticipated in data tables of their own, which was facilitated by version 4.0 of VARCHART XGantt. In contrast, ResourceScheduler1 merely allowed the different objects like tasks, operations, assignments and resources to be implicitely defined in the maindata table.

The below object types exist in ResourceScheduler2 and need to be defined in data tables of their own; resources may even be defined in up to 25 different tables:

- **Tasks**: These objects are composed by operations (see below) and hold basic properties such as the release date, the due date, priority and quantity.
- **Operations**: These objects can be assigned to resources (see below) by assignments (see below) and will receive the start and end dates of the processing time as a result of scheduling. Operations have a defined position within a sequence of their task and can be marked as "started". Beside, several different sequences of operations can be defined that represent mutually exclusive "routes" of processing. All operations of a route selected by the scheduling procedure will be scheduled.
- **Resources**: As their main features, these objects are part of a capacity curve and after scheduling, they also are part of a workload curve. Beside, they time the operations that they have received (timing resource). Therefore, in order to be scheduled, an operation needs to be assigned to a resource. Beside a timing resource, also work and material resources can be assigned to an operation. Another essential feature of a timing resource is its ability to be grouped on multiple levels. A timing resource may belong to different groups at one time.
- **Assignments**: These objects are the links between operations and resources, that allow to specify a factor for the quantity to be multiplied or divided. When groups of timing resources are scheduled, the assignments are marked correspondingly and additional assignments are generated for each single resource, so that they can be scheduled and displayed in VARCHART XGantt.

- **Links**: These objects describe the sequence of tasks, i.e., preceding tasks have to be finished before the succeeding ones can start.

Properties

- AssignmentDataTableName
- AssignmentIsResultFieldIndex
- AssignmentIsVisibleFieldIndex
- AssignmentLoadOrConsumptionPerItemFieldIndex
- AssignmentMaximumLoadFieldIndex
- AssignmentMinimumLoadFieldIndex
- AssignmentMinimumMaximumLoadType
- AssignmentOperationIDFieldIndex
- AssignmentResourceIDFieldIndex
- AssignmentResourceSelectionStrategyFieldIndex
- BaseCalendarUsageForSupplementTimes
- BaseTimeUnit
- BaseTimeUnitsPerStep
- DataRecordEventsEnabled
- DefaultOperationMaximumInterruptionTime
- DefaultResourceCalendarName
- FullUsageOfPlanningUnitsEnabled
- LinkDataTableName
- LinkDurationFieldIndex
- LinkPredecessorOperationIDFieldIndex
- LinkPredecessorTaskIDFieldIndex
- LinkSuccessorOperationIDFieldIndex
- LinkSuccessorTaskIDFieldIndex
- OperationDataTableName
- OperationLoadPerItemFieldIndex
- OperationMaximumInterruptionTimeFieldIndex
- OperationMinimumSupplementTimeFieldIndex
- OperationOverlapQuantityFieldIndex
- OperationPostLoadFieldIndex
- OperationPostOffsetFieldIndex
- OperationPreparationLoadFieldIndex
- OperationPreparationOffsetFieldIndex
- OperationResultEndDateFieldIndex
- OperationResultPostEndDateFieldIndex
- OperationResultPreparationStartDateFieldIndex

- OperationResultProcessingTimeFieldIndex
- OperationResultSelectedTimingResourceIDFieldIndex
- OperationResultStartDateFieldIndex
- OperationResultStatusFieldIndex
- OperationRouteFieldIndex
- OperationSequenceNumberFieldIndex
- OperationStartLockDateFieldIndex
- OperationTaskIDFieldIndex
- OperationWorkInProcessFieldIndex
- PlanningEndDate
- PlanningStartDate
- PlanningStrategy
- ResourceCalendarNameFieldIndex
- ResourceCapacityType
- ResourceCapacityTypeFieldIndex
- ResourceConstraintTypeFieldIndex
- ResourceDataTableName
- ResourceEfficiencyFieldIndex
- ResourceGroupDataTableName
- ResourceGroupIDFieldIndex
- ResourceNameFieldIndex
- ResourceResultLoadCurveNamePrefix
- ResourceResultStockCurveNamePrefix
- ResourceSelectionStrategy
- ResourceType
- ResultProcessingStepCount
- TaskDataTableName
- TaskDueDateFieldIndex
- TaskPlanningStrategyFieldIndex
- TaskPriorityFieldIndex
- TaskQuantityFieldIndex
- TaskReleaseDateFieldIndex
- TaskResultEndDateFieldIndex
- TaskResultPostEndDateFieldIndex
- TaskResultPreparationStartDateFieldIndex
- TaskResultProcessingStepFieldIndex
- TaskResultProcessingTimeFieldIndex
- TaskResultRouteFieldIndex
- TaskResultStartDateFieldIndex
- ToleranceTimeOnASAPDueDates

- ToleranceTimeOnJITReleaseDates
- ToleranceTimeOnStartLockDates
- WorkInProcessType
- WritingDebugFilesEnabled

Methods

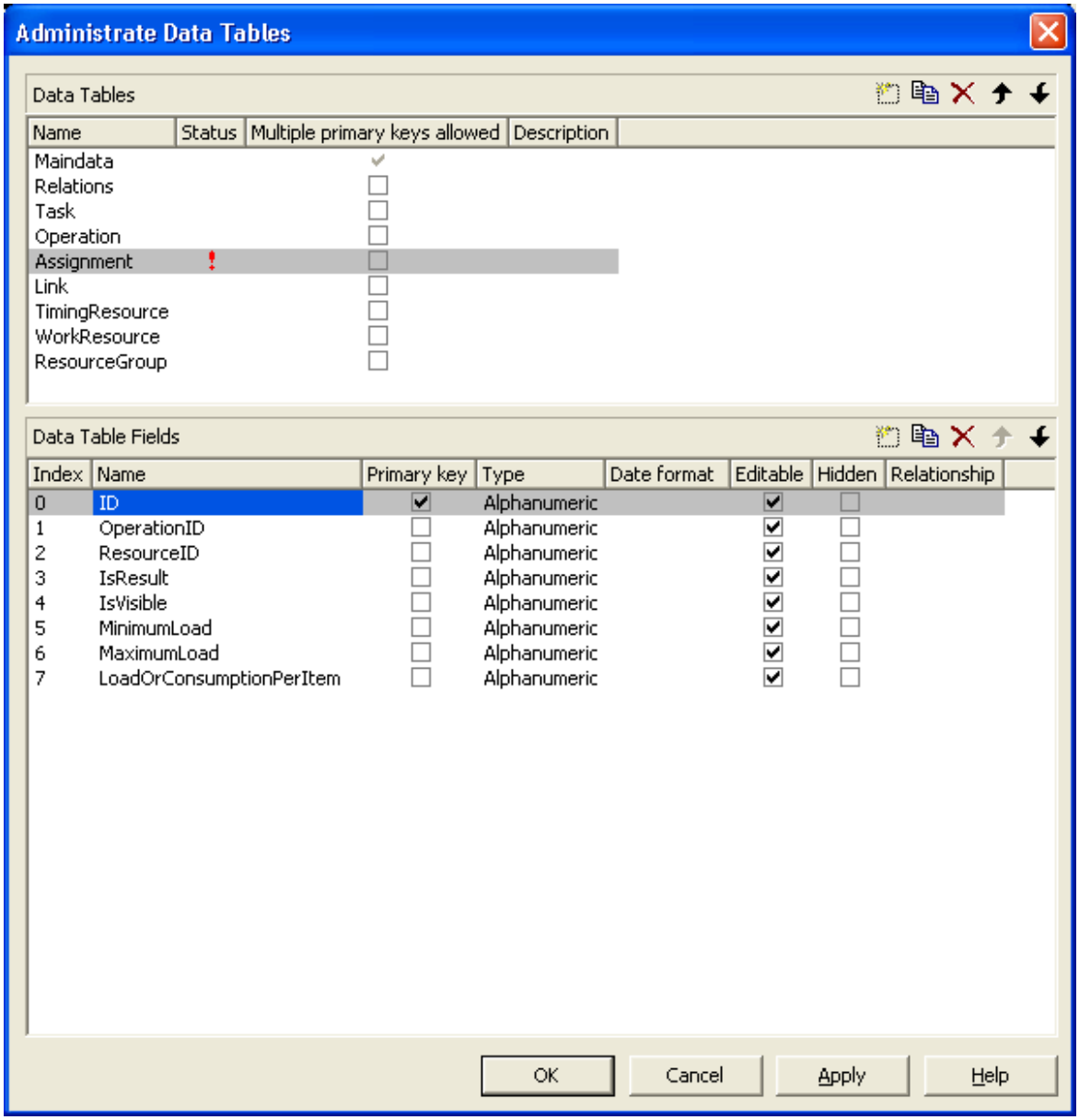
- DetermineIDOfFirstOperationByTaskID
- DetermineIDOfLastOperationByTaskID
- Process

Properties

AssignmentDataTableName

Property of VcResourceScheduler2

This property lets you set or retrieve the name of the assignment data table that holds assignments of operations to resources. Setting this name is mandatory.



	Data Type	Explanation
Property value	String **Possible Values:**	Name of the assignment data table **Default value:** Empty string Name of the color map

Example Code

```
VcGantt1.ResourceScheduler2.AssignmentDataTableName = "Assignment"
```

AssignmentIsResultFieldIndex

Property of VcResourceScheduler2

This property lets you set or retrieve the index of a data field in the assignment table where VARCHART XGantt notes whether the corresponding data set was generated by resource scheduling. In the picture referring to **AssignmentDataTableName**, the field index for example is 3. Setting this property is optional. The scheduling procedure generates assignments only, if during the start among the existing assignments there are ones that refer to resource groups. Then the scheduling procedure generates an assignment to a resource that it selects from the group, and sets its corresponding field to 1. Assignments provided by the application either should not hold a value at all or should set it to 0.

Using this field allows for multiple invoking while the results are kept stable, which saves the application from having to manually re-set the assignments to their original state. The scheduling procedure continues to use assignments once generated in order to avoid dispensable actions of deleting and generating.

	Data Type	Explanation
Property value	Long	Index of the data field in the assignment data table that is designated to hold the values on the identification of data records that were generated by resource scheduling. {-1...NumberOfFieldsInAssignmentDataTable -1}. By setting the index to -1, no data field of the assignment data table will be assigned to this property. **Default value:** -1

Example Code

```
VcGantt1.ResourceScheduler2.AssignmentIsResultFieldIndex = 3
```

AssignmentIsVisibleFieldIndex

Property of VcResourceScheduler2

This property lets you set or retrieve the index of a data field in the assignment data table where the resource scheduling module notes whether the assignment should be made visible. In the picture referring to **AssignmentDataTableName**, the field index for example is 4. The field is useful for instance for displaying assignments to groups of resources in the Gantt graph before running the resource scheduling module, and for displaying the resulting single resources afterwards.

	Data Type	Explanation
Property value	Long	Index of the data field in the assignment data table that is designated to hold the values on the visibility. {-1...NumberOfFieldsInAssignmentDataTable -1}. By setting the index to -1, no data field of the assignment data table will be assigned to this property. **Default value:** -1

Example Code

```
VcGantt1.ResourceScheduler2.AssignmentIsVisibleFieldIndex = 4
```

AssignmentLoadOrConsumptionPerItemFieldIndex

Property of VcResourceScheduler2

This property lets you set or retrieve the index of a data field in the assignment table which holds a value per item (see property **TaskQuantity-FieldIndex**). You can assign values per item to work resources and a material resources only. An index of -1 will be interpreted as 1. If the data field in the data set does not contain a valid value, 0 will be assumed. If the data field is of the type **String**, you can also enter a float value.

	Data Type	Explanation
Property value	Long	Index of the data set in the assignment data table that is designated to hold the value. {-1...NumberOfFieldsInAssignmentDataTable -1}. By setting the index to -1, no data field of the assignment data table will be assigned to this property. **Default value:** -1

AssignmentMaximumLoadFieldIndex

Property of VcResourceScheduler2

The index passed as the property value specifies a data field in the assignment data table that holds the maximum workload limit of a resource. In the picture referring to **AssignmentDataTableName**, the field index for example is 6.

This kind of limit can only be assigned to assignments of timing resources. The data field contains percentage values from {0...100}, where both, the value 0 and an empty field are interpreted as 100.

Values between 1 and 99 in the data field will disable the properties **FullUsageOfPlanningUnitsEnabled** and **OperationMaximumInterruptionTimeFieldIndex**.

Also see **AssignmentMinimumLoadFieldIndex**.

	Data Type	Explanation
Property value	Long	Index of the data field in the assignment data table that is designated to hold the maximum workload limit of a resource. {-1...NumberOfFieldsInAssignmentDataTable -1}. By setting the index to -1, no data field of the assignment data table will be assigned to this property. **Default value:** -1

Example Code

```
VcGantt1.ResourceScheduler2.AssignmentMaximumLoadFieldIndex = 6
```

AssignmentMinimumLoadFieldIndex

Property of VcResourceScheduler2

The index passed as the property value specifies a data field in the assignment data table that holds the minimum workload limit of a resource. In the picture referring to **AssignmentDataTableName**, the field index for example is 5. The limit can only be assigned to timing resources.

The data field contains percentage values from {0...100}. Also see **AssignmentMaximumLoadFieldIndex**.

	Data Type	Explanation
Property value	Long	Index of the data field in the assignment data table that is designated to hold the minimum workload limit of a resource. {-1...NumberOfFieldsInAssignmentDataTable -1}. By setting the index to -1, no data field of the assignment data table will be assigned to this property. **Default value:** -1

Example Code

```
VcGantt1.ResourceScheduler2.AssignmentMinimumLoadFieldIndex = 5
```

AssignmentMinimumMaximumLoadType

Read Only Property of VcResourceScheduler2

This property lets you set or retrieve whether the values that are assigned to the data fields by the indices set by the properties **AssignmentMinimum-LoadFieldIndex** and **AssignmentMaximumLoadFieldIndex** are relative to the resource capacity or absolute.

Absolute values are useful e.g. if the assigned resource is a team with a varying number of persons and the assignment shall not occupy the whole team.

	Data Type	Explanation
Property value	ResourceSchedulingMinimumMaximumLoadTypeEnum	Field values absolute/relative to resource capacity **Default value:** vcResSchedPercentageValues
	Possible Values:	
	vcResSchedAbsoluteValues 2	Data field values absolute to resource capacity
	vcResSchedPercentageValues 0	Data field values relative to resource capacity

AssignmentOperationIDFieldIndex

Property of VcResourceScheduler2

This property lets you set or retrieve the index to a data field in the assignment table which holds the ID of an operation. In the picture referring to **AssignmentDataTableName**, the field index for example is 1. This property must not be set to -1 during a scheduling procedure.

	Data Type	Explanation
Property value	Long	Index of the data field in the assignment data table that is designated to hold the operation ID. {-1...NumberOfFieldsInAssignmentDataTable -1}. By setting the index to -1, no data field of the assignment data table will be assigned to this property. **Default value:** -1

Example Code

```
VcGantt1.ResourceScheduler2.AssignmentOperationIDFieldIndex = 1
```

AssignmentResourceIDFieldIndex

Property of VcResourceScheduler2

This indexed property lets you set or retrieve the index of a data field which holds IDs of resources. In the picture referring to **AssignmentDataTable-Name**, the field index for example is 2.

The index passed as a parameter denotes one out of 25 resource tables. The ones used are set by the indexed property **ResourceDataTableName**.

	Data Type	Explanation
Parameter: resourceTableIndex	Short	Index of a resource data table according to the assignments made by **ResourceDataTableName** {0...24}
Property value	Long	Index of the data field in the assignment data table that is designated to hold resource IDs. {-1...NumberOfFieldsInAssignmentDataTable -1}. By setting the index to -1, no data field of the assignment data table will be assigned to this property. **Default value:** -1

Example Code

```
VcGantt1.ResourceScheduler2.AssignmentResourceIDFieldIndex(0) = 2
```

AssignmentResourceSelectionStrategyFieldIndex

Property of VcResourceScheduler2

The index passed as the property value specifies a data field in the assignment data table that defines a resource selection strategy for the corresponding assignment to a resource group. If this field is empty for a resource or if the property is set to -1, the value of the general property **ResourceSelectionStrategy** is valid (see there).

The data field can contain the below list of values:

0: equals vcResSchedRSSequential

1: equals vcResSchedRSLeastLoaded

2: equals vcResSchedRSMostLoaded

3: equals vcResSchedRSHighestEfficiency

7: equals vcResSchedRSFirstAvailable

The values 1 and 2 (LeastLoaded and MostLoaded) entail consecutive adding of resource occupation that forms the base for selecting the resource loaded least or most. So if planning periods of tasks differ widely or if both planning strategies are applied, the results may not prove satisfactory.

When using the value 7 (FirstAvailable) the selection merely depends on the first timing resource. Other assignments of the operation are not taken into consideration. So when using material and work resources, the results may not turn out satisfactory.

	Data Type	Explanation
Property value	Long	Index of the data field in the assignment data table that is designated to hold the data of the resource selction strategy. {-1...NumberOfFieldsInAssignmentDataTable -1}. By setting the index to -1, no data field of the assignment data table will be assigned to this property. **Default value:** -1

BaseCalendarUsageForSupplementTimes

Property of VcResourceScheduler2

If this property is set to **false**, no calendar will be used to define minimum supplement times (indirectly defined by the property **VcResource-Scheduler2.OperationMinimumSupplementTimeFieldIndex**), so the time period specified will directly apply (example: could be used for drying produced parts). If this property is set to **true**, the base calendar of the Gantt object will be used with the supplement time being worked off as a working time defined in the base calendar (example: could be used for the transport of produced parts).

Please also see **VcResourceScheduler2.OperationMinimumSupplement-TimeFieldIndex**.

	Data Type	Explanation
Property value	Boolean	**True**: The base calendar of the Gantt object will be used. **False**: The specified time period will be used directly. **Default value:** False
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.ResourceScheduler2.BaseCalendarUsageForSupplementTimes = True
```

BaseTimeUnit

Property of VcResourceScheduler2

This property lets you set or enquire the basic time unit for resource scheduling, which may differ from the basic time unit set by the property **VcGantt.TimeUnit**. The values of the capacity, work load and stock curves refer to the base unit defined here.

	Data Type	Explanation
Property value	TimeUnitEnum	Time unit **Default value:** Value, which was set during design time by vcGantt.TimeUnit. If no setting was made, the value is **vcDay**.
	Possible Values: vcDay 5 vcHour 6 vcMinute 7 vcSecond 8	 Time unit **day** Time unit **hour** Time unit **minute** Time unit **second**

Example Code

```
VcResourceScheduler2.BaseTimeUnit = VcTimeUnit.vcMinute
VcResourceScheduler2.BaseTimeUnitsPerStep = 15
```

BaseTimeUnitsPerStep

Property of VcResourceScheduler2

This property lets you set or enquire the size of steps of the scheduling. The larger this value, the faster, but also the coarser the result will be. The value entered here represents a multiple of the base unit set by **VcResource-Scheduler2.BaseTimeUnit**.

	Data Type	Explanation
Property value	Integer	Number of time units per step **Default value:** 1
	Possible Values:	Data field index

Example Code

```
VcGantt1.ResourceScheduler2.BaseTimeUnitsPerStep = 4
```

DataRecordEventsEnabled

Property of VcResourceScheduler2

If this property is set to **True**, events will be triggered that indicate data modifications during the process methods: VcGantt.OnDataRecordModify, VcGantt.OnDataRecordModifyComplete, VcGantt.OnDataRecordCreate, VcGantt.OnDataRecordCreateComplete, VcGantt.OnDataRecordDelete and VcGantt.OnDataRecordDeleteComplete.

	Data Type	Explanation
Property value	Boolean	**True**: events are triggered. **False**: events are not triggered. **Default value:** False
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.ResourceScheduler2.DataRecordEventsEnabled = True
```

DefaultOperationMaximumInterruptionTime

Property of VcResourceScheduler2

By this property you can set or retrieve a default value of the maximum time span, for which the operation is allowed be interrupted. The value is a number that represents base time units (see property **BaseTimeUnit**). The default value applies if the property **OperationMaximumInterruption-TimeFieldIndex** was set to -1 or if the value read from the operations table equals 0 or if the field is empty. If the value is set to 0, no interruption will be allowed.

This property will be disabled by setting the maximum load to less than 100% (see property **AssignmentMaximumLoadFieldIndex**).

	Data Type	Explanation
Property value	Long	Number of base time units **Default value:** 0

Example Code

```
VcGantt1.ResourceScheduler2.DefaultOperationMaximumInterruptionTime = 1
```

DefaultResourceCalendarName

Property of VcResourceScheduler2

This property lets you set a calendar name which is used if no calendar of the same name as the resource is found by the properties **VcResource-Scheduler2.ResourceCalendarNameFieldIndex** and **VcResource-Scheduler2.ResourceNameFieldIndex**. If you do not set the property, the resource will use the default calendar of the XGantt object. (see **VcCalendarCollection.Active**).

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the calendar **Default value:** Empty string Name of the color map

Example Code

```
VcGantt1.ResourceScheduler2.DefaultResourceCalendarName = ""
```

FullUsageOfPlanningUnitsEnabled

Property of VcResourceScheduler2

If this property is set to **True**, during the first and/or the last time unit of the occupation time of a resource allocated to a task, a second task may finish or start. This way, remaining capacities can be used up. If this property is set to **False**, remaining capacities will not be used.

This property merely influences the first operation of a task. It does not have any impact on the operations following.

This property will be disabled by setting the maximum load to less than 100% (see property **AssignmentMaximumLoadFieldIndex**).

	Data Type	Explanation
Property value	Boolean	**True**: remaining capacities are used. **False**: remaining capacities are not used. **Default value:** True
	Possible Values:	Group invisible/visible group nodes are/are not visible

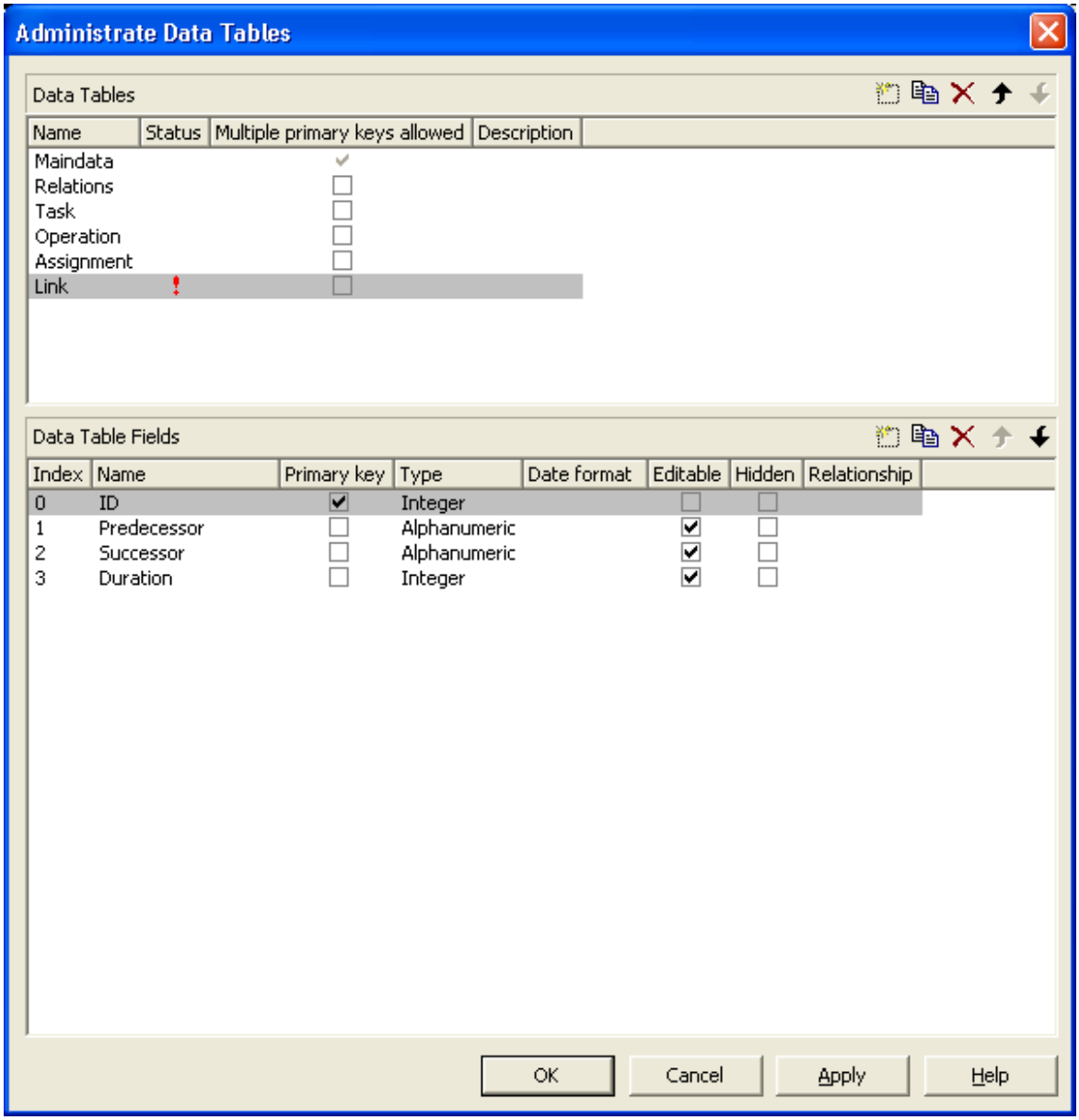
Example Code

```
VcGantt1.ResourceScheduler2.FullUsageOfPlanningUnitsEnabled = True
```

LinkDataTableName

Property of VcResourceScheduler2

This property lets you set or retrieve the name of the linkData table, that holds links. If you do not set this name, links will not be taken into account during the run of the resource scheduling module.



	Data Type	Explanation
Property value	String **Possible Values:**	Name of the link data table **Default value:** Empty string Name of the color map

Example Code

```
VcGantt1.ResourceScheduler2.LinkDataTableName = "Link"
```

LinkDurationFieldIndex

Property of VcResourceScheduler2

This property lets you set or retrieve the index of a data field in the link data table in which a minimum temporal distance between predecessor and successor can be stored. This distance can also be negative. Unit: as set by the method BaseTimeUnit. In the picture referring to **LinkDataTableName**, the field index for example is 3.

As a limit, when applying the planning strategy ASAP, a successor cannot start earlier than a predecessor ; when applying the planning strategy JIT, a predecessor cannot finish later than a successor.

	Data Type	Explanation
Property value	Long	Index of the data field in the link data table that is designated to hold the values on the duration. {-1...NumberOfFieldsInLinkDataTable -1}. By setting the index to -1, no data field of the link data table will be assigned to this property. **Default value:** -1

Example Code

```
VcGantt1.ResourceScheduler2.LinkDurationFieldIndex = 3
```

LinkPredecessorOperationIDFieldIndex

Property of VcResourceScheduler2

This property lets you set or retrieve the index of a data field in the link data table the values of which contain the ID of the predecessor operation. As the resource scheduling module is only able to draw links between tasks, this property facilitates the use of links in XGantt which currently can only be displayed between operations. Thus the links are internally always created between the tasks of the operations specified by the ID.

When using a link data table, it is mandatory to set this property to a value not equal to -1 unless the VcResourceScheduler2 property **LinkPredecessorTaskIDFieldIndex** is used.

	Data Type	Explanation
Property value	Long	Index of the data field in the link data table that is designated to hold the IDs of the predecessor operation. {-1...NumberOfFieldsInLinkDataTable -1}. By setting the index to -1, no data field of the link data table will be assigned to this property. **Default value:** -1

Example Code

```
VcGantt1.ResourceScheduler2.LinkPredecessorOperationIDFieldIndex = 1
```

LinkPredecessorTaskIDFieldIndex

Property of VcResourceScheduler2

This property lets you set or retrieve the index of a data field in the link data table the values of which contain the ID of the predecessor task. In the picture referring to **LinkDataTableName**, the field index for example is 1.

When using a link data table, it is mandatory to set this property to a value not equal to -1 unless the VcResourceScheduler2 property **LinkPredecessorOperationIDFieldIndex** is used.

	Data Type	Explanation
Property value	Long	Index of the data field in the link data table that is designated to hold the IDs of the predecessor task. {-1...NumberOfFieldsInLinkDataTable -1}. By setting the index to -1, no data field of the link data table will be assigned to this property. **Default value:** -1

Example Code

```
VcGantt1.ResourceScheduler2.LinkPredecessorTaskIDFieldIndex = 1
```

LinkSuccessorOperationIDFieldIndex

Property of VcResourceScheduler2

This property lets you set or retrieve the index of a data field in the link data table the values of which contain the ID of the successor operation. As the resource scheduling module is only able to draw links between tasks, this property facilitates the use of links in XGantt which currently can only be displayed between operations. Thus the links are internally always created between the tasks of the operations specified by the ID.

When using a link data table, it is mandatory to set this property to a value not equal to -1 unless the VcResourceScheduler2 property **LinkPredecessorTaskIDFieldIndex** is used.

	Data Type	Explanation
Property value	Long	Index of the data field in the link data table that is designated to hold the IDs of the successor operation. {-1...NumberOfFieldsInLinkDataTable -1}. By setting the index to -1, no data field of the link data table will be assigned to this property. **Default value:** -1

Example Code

```
VcGantt1.ResourceScheduler2.LinkSuccessorOperationIDFieldIndex = 1
```

LinkSuccessorTaskIDFieldIndex

Property of VcResourceScheduler2

This property lets you set or retrieve the index of a data field in the link data table that contains the ID of the successor task. In the picture referring to **LinkDataTableName**, the field index for example is 2.

When using a link data table, it is mandatory to set this property to a value not equal to -1 unless the VcResourceScheduler2 property **LinkSuccessorOperationIDFieldIndex** is used.

	Data Type	Explanation
Property value	Long	Index of the data field in the link data table that is designated to hold the IDs of successor tasks. {-1...NumberOfFieldsInLinkDataTable -1}. By setting the index to -1, no data field of the link data table will be assigned to this property. **Default value:** -1

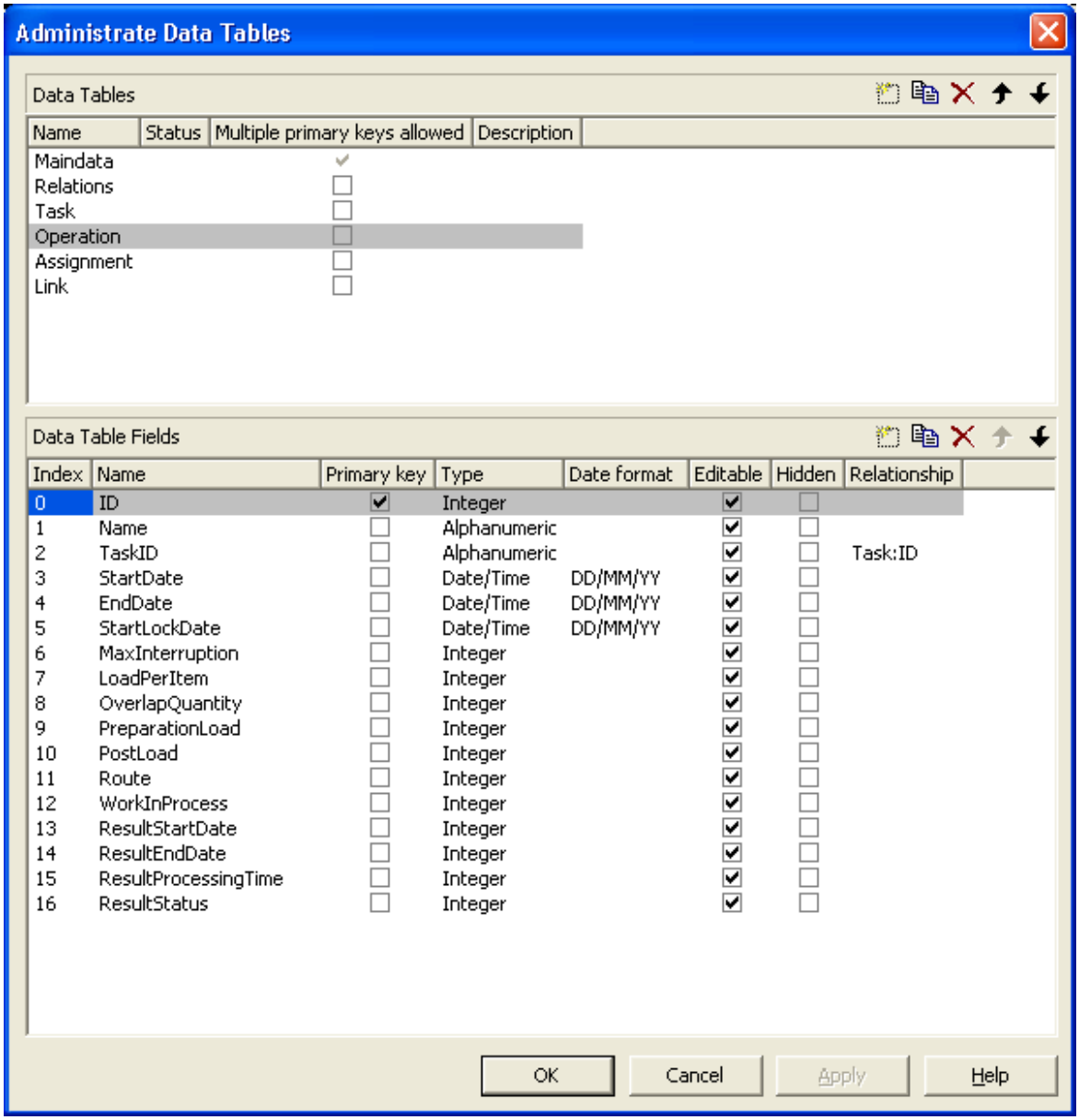
Example Code

```
VcGantt1.ResourceScheduler2.LinkSuccessorTaskIDFieldIndex = 2
```

OperationDataTableName

Property of VcResourceScheduler2

This property lets you set or retrieve the name of the operation data table that holds data of the operations. Setting this name is mandatory.



	Data Type	Explanation
Property value	String **Possible Values:**	Name of the operation data table **Default value:** Empty string Name of the color map

Example Code

```
VcGantt1.ResourceScheduler2.OperationDataTableName = "Operation"
```

OperationLoadPerItemFieldIndex

Property of VcResourceScheduler2

This property lets you set or retrieve the index of a data field in the operation data table that holds the load of a timing resource per item. To receive the total load on the timing resource, the value in the data field specified will be multiplied with the number specified by the task. If the data field holds an invalid value or if this property is set to -1, a value of 0 will be assumed.

	Data Type	Explanation
Property value	Long	Index of the data field in the operation data table that is designated to hold the values of the load. {-1...NumberOfFieldsInOperationDataTable -1}. By setting the index to -1, no data field of the operation data table will be assigned to this property. **Default value:** -1

Example Code

```
VcGantt1.ResourceScheduler2.OperationLoadPerItemFieldIndex = 10
```

OperationMaximumInterruptionTimeFieldIndex

Property of VcResourceScheduler2

The index specifies a data field in the operation data table to which a maximum time span is stored, for which the operation is allowed be interrupted. In the picture referring to **OperationDataTableName**, the field index for example is 9.

An interruption is a period free of activity on a resource that was fully loaded and allocated to an operation. It differs from a "break" by not being caused by a pre-defined workfree time.

The content of this field is a number that represents base time units (see property **BaseTimeUnit**).

If this property is set to -1 or if the value of the field equals zero or is empty, the value set by the property **DefaultOperationMaximumInterruption-Time** will be used. If the latter also equals 0, an interruption is not allowed. If the value is < 0, an interruption also is not allowed, even if the property **DefaultOperationMaximumInterruptionTime** does not equal 0.

This property will be disabled by setting the maximum load to less than 100% (see property **AssignmentMaximumLoadFieldIndex**.

	Data Type	Explanation
Parameter:		
timeSpan	Long	Index of the data field in the operation data table that is designated to hold the degree of completion. {-1...NumberOfFieldsInOperationsDataTable -1}. By setting the index to -1, no data field of the operation data table will be assigned to this property.
Property value	Long	Index of the data field in the operation data table that is designated to hold the values of the maximum interruption time. {-1...NumberOfFieldsInOperationDataTable -1}. By setting the index to -1, no data field of the operation data table will be assigned to this property. **Default value:** -1

Example Code

```
VcGantt1.ResourceScheduler2.OperationMaximumInterruptionTimeFieldIndex = 9
```

OperationMinimumSupplementTimeFieldIndex

Property of VcResourceScheduler2

The index specifies a data field in the operation data table to which a minimum supplement time of the operation is stored. During supplement time, the resources affected by this operation will not be occupied, so this time span can be used for standby or idle times.

The content of this field is a number that represents base time units (s. property **BaseTimeUnit**). In the picture referring to **OperationDataTable-Name**, the field index for example is 7.

Please also see **OperationMaximumSupplementTimeFieldIndex**, **OperationPreparationLoadFieldIndex** and **OperationPostLoadField-Index**

BaseCalendarUsageForSupplementTimes.

	Data Type	Explanation
Property value	Long	Index of the data field in the operation data table that is designated to hold the values of the minimum supplement time. {-1...NumberOfFieldsInOperationDataTable -1}. By setting the index to -1, no data field of the operation data table will be assigned to this property. **Default value:** -1

Example Code

```
VcGantt1.ResourceScheduler2.OperationMinimumSupplementTimeFieldIndex = 7
```

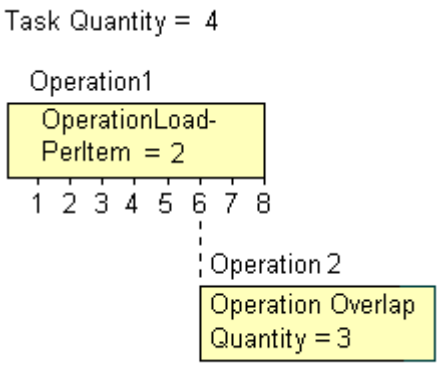
OperationOverlapQuantityFieldIndex

Property of VcResourceScheduler2

This property lets you set or retrieve the index of a data field in the operation data table that holds the 'overlap' quantity of an operation. Overlapping can only occur in tasks that were scheduled according to the strategy ASAP. This is the field to make succeeding resources overlap, which is useful if the succeeding operation does not have to wait for the preceding one to finish.

The quantity specified in the data field refers to the quantity of the task, set by the property **TaskQuantityFieldIndex**. The succeeding operation starts earliest after the preceding one has worked off the quantity specified (or later, optionally), overlapping the preceding one.

In the example below, the value of the overlap field equals 3. It refers to the quantity of 4. After 3 units of those 4 units were worked off by operation1, operation2 will start. A possibly defined load per item for operation1 (in the below example =2) will be multiplied by the overlap value: 3*2=6. Therefore operation2 starts after operation has reached the value of 6.



Scenario sample: 4 candle sticks are to be produced, each one holding 3 candles. 2 candle sticks and 6 candles are put in a package. After 6 candles were produced by operation1, operation2 starts packing.

If the index set by the property is empty or if it contains a value = 0, the operation will not overlap the preceding one; if the value equals -1, the operation will start at the same time as the preceding one.

If a preparation time was defined, it will be taken into consideration within the overlapping period. So probably, the preparation time needs to be divided by the load per item of the operation (see OperationLoadPerItemFieldIndex) and added to the overlapping quantity. This property should not be used simultaneously with the property **ResourceEfficienyFieldIndex**; the same is valid for **AssigmentMaximumLoadFieldIndex**.

	Data Type	Explanation
Property value	Long	Index of the data field in the operation data table that is designated to hold the values of the 'overlap' quantity. {-1...NumberOfFieldsInOperationDataTable -1}. By setting the index to -1, no data field of the operation data table will be assigned to this property. **Default value:** -1

Example Code

```
VcGantt1.ResourceScheduler2.OperationOverlapQuantityFieldIndex = 11
```

OperationPostLoadFieldIndex

Property of VcResourceScheduler2

The index specifies a data field in the operation data table to which a post time of the operation is stored. During the post time, the resources affected by this operation will be occupied.

The content of the designated field is a number that represents the required capacity. In the picture referring to **OperationDataTableName**, the field index for example is 13.

Please also see **OperationPreparationLoadFieldIndex**, **OperationMaximumInterruptionmentTimeFieldIndex** and **OperationMinimumSupplementTimeFieldIndex**.

If you want to define post times that resource-independent you can use the property **OperationPostOffsetFieldIndex**.

	Data Type	Explanation
Property value	Long	Index of the data field in the operation data table that is designated to hold the values of the post time. {-1...NumberOfFieldsInOperationDataTable -1}. By setting the index to -1, no data field of the operation data table will be assigned to this property. **Default value:** -1

Example Code

```
VcGantt1.ResourceScheduler2.OperationPostLoadFieldIndex = 13
```

OperationPostOffsetFieldIndex

Property of VcResourceScheduler2

The index specifies a data field in the operation data table to which a post time of each operation is stored. If the field contains positive integers (in the current base time unit), the time of the operations is resource-independent. If the index equals -1, there are no post times. This also applies if the index refers to a data field that contains a non-valid number or a 0 in the according operation.

Please also see **OperationPreparationOffsetFieldIndex**.

If you want to define resource-dependent post times you can use the property **OperationPostLoadFieldIndex**.

	Data Type	Explanation
Property value	Long	Index of the data field in the operation data table that specifies whether the follow-up time of an operation is to be resource-independent. {-1...NumberOfFieldsInOperationDataTable -1}. If the index is set to -1, there are no follow-up times. **Default value:** -1

Example Code

```
VcGantt1.ResourceScheduler2.OperationPostOffsetFieldIndex = 8
```

OperationPreparationLoadFieldIndex

Property of VcResourceScheduler2

The index specifies a data field in the operation data table to which a preparation time of the operation is stored. During the preparation time, the resources affected by this operation will be occupied.

The content of the designated field is a number that represents the required capacity. In the picture referring to **OperationDataTableName**, the field index for example is 12.

Please also see **OperationPostLoadFieldIndex**, **OperationMaximum-InterruptionTimeFieldIndex** and **OperationMinimumSupplementTime-FieldIndex**.

If you want to define resource-independent post times that you can use the property **OperationPreparationOffsetFieldIndex**.

	Data Type	Explanation
Property value	Long	Index of the data field in the operation data table that is designated to hold the values of the preparation time. {-1...NumberOfFieldsInOperationDataTable -1}. By setting the index to -1, no data field of the operation data table will be assigned to this property. **Default value:** -1

Example Code

```
VcGantt1.ResourceScheduler2.OperationPreparationLoadFieldIndex = 12
```

OperationPreparationOffsetFieldIndex

Property of VcResourceScheduler2

The index specifies a data field in the operation data table to which a preparation time of each operation is stored. If the field contains positive integers (in the current base time unit), the preparation time of the operations is resource-independent. If the index equals -1, there are no preparation times. This also applies if the index refers to a data field that contains a non-valid number or a 0 in the according operation.

Please also see **OperationPostOffsetFieldIndex**.

If you want to define resource-dependent preparation times you can use the property **OperationPreparationLoadFieldIndex**.

	Data Type	Explanation
Property value	Long	Index of the data field in the operation data table that specifies whether the lead time of an operation is to be resource-independent. {-1...NumberOfFieldsInOperationDataTable -1}. If the index is set to -1, there are no lead times. **Default value:** -1

Example Code

```
VcGantt1.ResourceScheduler2.OperationPreparationOffsetFieldIndex = 8
```

OperationResultEndDateFieldIndex

Property of VcResourceScheduler2

This property lets you set or retrieve the index of a data field in the operation data table to which the calculated finish date of the operation is stored.

In the picture referring to **OperationDataTableName**, the field index for example is 17.

To receive sensible results for the scheduling procedure, at least two out of the three properties **OperationResultStartDateFieldIndex**, **Operation-ResultProcessingTimeFieldIndex** and **OperationResultEndDateField-Index** need to be set to a value unequal to -1.

	Data Type	Explanation
Property value	Long	Index of the data field in the operation data table that is designated to hold the values of the end date. {-1...NumberOfFieldsInOperationDataTable -1}. By setting the index to -1, no data field of the operation data table will be assigned to this property. **Default value:** -1

Example Code

```
VcGantt1.ResourceScheduler2.OperationResultEndDateFieldIndex = 17
```

OperationResultPostEndDateFieldIndex

Property of VcResourceScheduler2

The index specifies a data field in the operation data table which holds the scheduled end date of the post time of an operation. If this phase is 0, the date is identical to the value in the data field which is referred to by the property **OperationResultEndDateFieldIndex**.

	Data Type	Explanation
Property value	Long	Index of the data field in the operation data table that is designated to hold the end date of the post time. {-1...NumberOfFieldsInOperationDataTable -1}. By setting the index to -1, no data field of the operation data table will be assigned to this property. **Default value:** -1

Example Code

```
VcGantt1.ResourceScheduler2.OperationResultPostEndDateFieldIndex = 15
```

OperationResultPreparationStartDateFieldIndex

Property of VcResourceScheduler2

The index specifies a data field in the operation data table which holds the scheduled start date of the preparation phase of an operation. If the preparation phase is 0, this date is identical to the value in the data field which is referred to by the property **OperationResultStartDateFieldIndex**.

	Data Type	Explanation
Property value	Long	Index of the data field in the operation data table that is designated to hold the start date of the preparation phase. {-1...NumberOfFieldsInOperationDataTable -1}. By setting the index to -1, no data field of the operation data table will be assigned to this property. **Default value:** -1

Example Code

```
VcGantt1.ResourceScheduler2.OperationResultPreparationStartDateFieldIndex = 10
```

OperationResultProcessingTimeFieldIndex

Property of VcResourceScheduler2

The index specifies a data field in the operation data table to which the calculated duration of the operation is stored. IIn the picture referring to **OperationDataTableName**, the field index for example is 18.

To receive sensible results for the scheduling procedure, at least two out of the three properties **OperationResultStartDateFieldIndex**, **OperationResultProcessingTimeFieldIndex** and **OperationResultEndDateFieldIndex** need to be set to a value unequal to -1.

	Data Type	Explanation
Property value	Long	Index of the data field in the operation data table that is designated to hold the values of the processing time. {-1...NumberOfFieldsInOperationDataTable -1}. By setting the index to -1, no data field of the operation data table will be assigned to this property. **Default value:** -1

Example Code

```
VcGantt1.ResourceScheduler2.OperationResultProcessingTimeFieldIndex = 18
```

OperationResultSelectedTimingResourceIDFieldIndex

Property of VcResourceScheduler2

The index specifies a data field in the operation data table which holds the scheduled ID of a timing resource that was selected by the module. Thus in a table or a layer annotation the assigned resource can easily be shown graphically.

	Data Type	Explanation
Property value	Long	Index of the data field in the operation data table that is designated to hold the ID of the timing resource that was selected by the module. {-0...NumberOfFieldsInOperationsDataTable -1}. By setting the index to -1, no data field of the operation data table will be assigned to this property. **Default value:** -1

Example Code

```
VcGantt1.OperationResultSelectedTimingResourceFieldIndex = 8
```

OperationResultStartDateFieldIndex

Property of VcResourceScheduler2

The index specifies a data field in the operations table to which the calculated start date of the operation is stored. In the picture referring to **OperationDataTableName**, the field index for example is 16.

To receive sensible results for the scheduling procedure, at least two out of the three properties **OperationResultStartDateFieldIndex**, **OperationResultProcessingTimeFieldIndex** and **OperationResultEndDateFieldIndex** need to be set to a value unequal to -1.

	Data Type	Explanation
Property value	Long	Index of the data field in the operation data table that is designated to hold the values of the start date. {-1...NumberOfFieldsInOperationDataTable -1}. By setting the index to -1, no data field of the operations table will be assigned to this property. **Default value:** -1

Example Code

```
VcGantt1.ResourceScheduler2.OperationResultStartDateFieldIndex = 16
```

OperationResultStatusFieldIndex

Property of VcResourceScheduler2

The index specifies a data field in the operation data table to which an error or a warning on scheduling the operation is stored. In the picture referring to **OperationDataTableName**, the field index for example is 19.

Possible values stored by the scheduling procedure:

0: the operation was scheduled

1: the operation was not scheduled because the scheduling procedure selected a different route of the task. This case can only occur if the property **OperationRouteFieldIndex** was set to a value unequal to -1.

1000: the operation was not scheduled

1001: the operation was not scheduled and it was an operation of a task causing the task not to be scheduled. The reasons for this can be various.

	Data Type	Explanation
Property value	Long	Index of the data field in the operation data table that is designated to hold the error values. {-1...NumberOfFieldsInOperationDataTable -1}. By setting the index to -1, no data field of the operation data table will be assigned to this property. **Default value:** -1

Example Code

```
VcGantt1.ResourceScheduler2.OperationResultStatusFieldIndex = 19
```

OperationRouteFieldIndex

Property of VcResourceScheduler2

This property lets you set or retrieve the index of a data field in the operation data table the values of which assign operations to routes. In the picture referring to **OperationDataTableName**, the field index for example is14.

Operations of the same content in this field belong to the same route. The content of this field also represents the name of the route.

Routes represent alternative ways to execute a task. The scheduling procedure checks the routes available and selects one for the task. This way, you can define several alternative operation sequences for the same task. Not more than 10 routes can be defined per task. The routes are selected in the sequence of their occurrence by the operations.

	Data Type	Explanation
Property value	Long	Index of the data field in the operation data table that is designated to hold the name of the route. {-1...NumberOfFieldsInOperationDataTable -1}. By setting the index to -1, no data field of the operation data table will be assigned to this property. **Default value:** -1

Example Code

```
VcGantt1.ResourceScheduler2.OperationRouteFieldIndex = 14
```

OperationSequenceNumberFieldIndex

Property of VcResourceScheduler2

This property lets you set or retrieve the index of a data field in the operation data table the values of which define the sequence of the operations

associated with a task. In the picture referring to **OperationDataTable-Name**, the field index for example is 6.

	Data Type	Explanation
Property value	Long	Index of the data field in the operation data table that is designated to hold the sequence values. {-1...NumberOfFieldsInOperationDataTable -1}. By setting the index to -1, no data field of the operation data table will be assigned to this property. **Default value:** -1

Example Code

```
VcGantt1.ResourceScheduler2.OperationSequenceNumberFieldIndex = 6
```

OperationStartLockDateFieldIndex

Property of VcResourceScheduler2

The index specifies a data field in the operation data table that holds a start date for each operation in case of ASAP planning strategy (see property **PlanningStrategy**. In the picture referring to **OperationDataTableName**, the field index for example is 5.

If the data field contains a valid date, the task will be locked in the place of that start date and will not be moved by the scheduling procedure, which makes sense in particular for tasks already started. Please also see the property **OperationWorkInProcessFieldIndex**).

By the property **ToleranceTimeOnStartLockDates** you can set an allowance by which an operation may differ, i.e. a delay by which the lock date may be belated. Please mind that tasks that have operations with locked start dates are not scheduled automatically by first priority. If you wish this to happen, you need to calculate priorities of the tasks manually (see property **TaskPriorityFieldIndex**).

	Data Type	Explanation
Property value	Long	Index of the data field in the operation data table that is designated to hold the lock date. {-1...NumberOfFieldsInOperationDataTable -1}. By setting the index to -1, no data field of the operation data table will be assigned to this property. **Default value:** -1

Example Code

```
VcGantt1.ResourceScheduler2.OperationStartLockDateFieldIndex = 5
```

OperationTaskIDFieldIndex

Property of VcResourceScheduler2

This property lets you set or retrieve the index of a data field in the operations table which holds the ID of the task that the operation belongs to. IIn the picture referring to **OperationDataTableName**, the field index for example is 2.

To have the operation scheduled, this property needs to be set to a value different from -1. The data field allows to assign several operations to a task. The sequence in which the operations of a task are scheduled depends on the value of the data field, the index of which is set by the property **Operation-SequenceNumberDataFieldIndex**.

	Data Type	Explanation
Property value	Long	Index of the data field in the operation data table that is designated to hold the task ID. {-1...NumberOfFieldsInOperationDataTable -1}. By setting the index to -1, no data field of the operation data table will be assigned to this property. **Default value:** -1

Example Code

```
VcGantt1.ResourceScheduler2.OperationTaskIDFieldIndex = 2
```

OperationWorkInProcessFieldIndex

Property of VcResourceScheduler2

The index specifies a data field in the operation data table that contains a field which holds the degree of completion of an operation. In the picture referring to **OperationDataTableName**, the field index for example is 15.

If the data field index was found to be -1 or no valid value can be provided by the field, 0% ("not started") will be assumed.

	Data Type	Explanation
Property value	Long	Index of the data field in the operation data table that is designated to hold the degree of completion. {-1...NumberOfFieldsInOperationsDataTable -1}. By setting the index to -1, no data field of the operation data table will be assigned to this property. **Default value:** -1

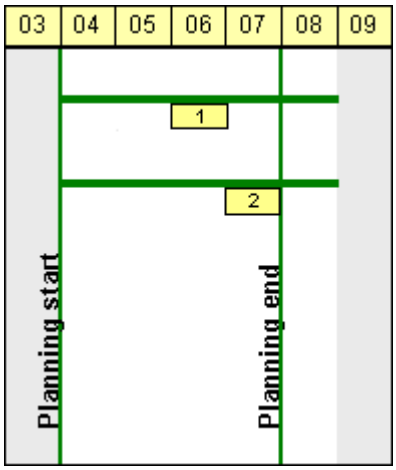
Example Code

```
VcGantt1.ResourceScheduler2.OperationWorkInProcessFieldIndex = 15
```

PlanningEndDate

Property of VcResourceScheduler2

By this property you can set or retrieve the end date of the scheduling period. If you do not set this date, the end date will be taken from the end of the time scale, set by the property **VcGantt.TimeScaleEnd**. The start of the scheduling period can be set by **PlanningStartDate**.



Limited scheduling period

	Data Type	Explanation
Property value	Date/Time	End date of the scheduling period **Default value:** DateTime.MinValue

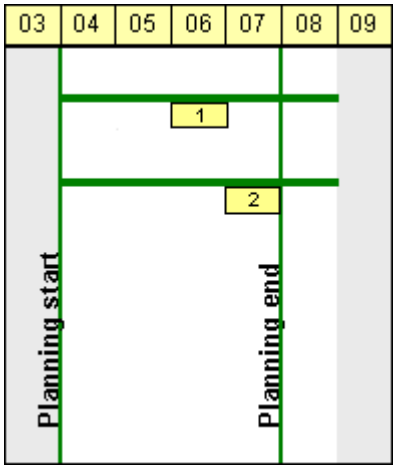
Example Code

```
VcGantt1.ResourceScheduler2.PlanningEndDate = VcGantt1.TimeScaleEnd
```

PlanningStartDate

Property of VcResourceScheduler2

By this property you can set or retrieve the start date of the scheduling period. If you do not set this date, the start date will be taken from the start of the time scale, set by the property **VcGantt.TimeScaleStart**. The end of the scheduling period can be set by **PlanningEndDate**.



Limited scheduling period

	Data Type	Explanation
Property value	Date/Time	Start date of the scheduling period **Default value:** DateTime.MinValue

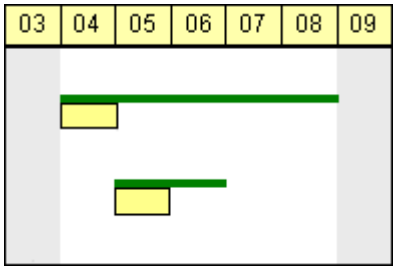
Example Code

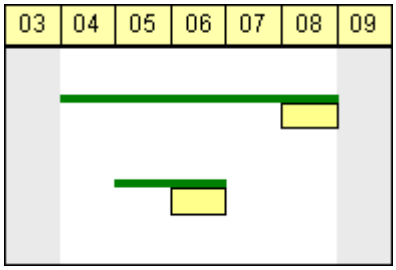
```
VcGantt1.ResourceScheduler2.PlanningStartDate = VcGantt1.TimeScaleStart
```

PlanningStrategy

Property of VcResourceScheduler2

This property specifies the planning strategy for tasks. Two options exist for planning strategies: One strategy aims at working off tasks as fast as possible to achieve a high turnover in the production system. Therefore, tasks start as soon as possible (ASAP). The other strategy aims at finishing tasks duely, for example to keep stocks low. Therefore, tasks finish just in time (JIT).





So in the ASAP strategy the start is early (picture left), while in the JIT strategy the finish is late (picture right). The long slim bars show the available period to complete a task, while the short big bars represent the actually allocated time for completion. So ASAP tasks tend to appear at the

beginning of the available period of completion, while JIT tasks tend to appear at its end.

If an individual setting of the planning strategy per task is required, you can assign a data field by **TaskPlanningStrategyFieldIndex** to individually overwrite settings of **PlanningStrategy**.

	Data Type	Explanation
Property value	ResourceSchedulingPlanningStrategy	Planning strategy **Default value:** vcResSchedPSASAP
	Possible Values: vcResSchedPSASAP 1- vcResSchedPSJIT 0	 As soon as possible Just in time

Example Code

```
VcGantt1.ResourceScheduler2.PlanningStrategy = vcResSchedPSASAP
```

ResourceCalendarNameFieldIndex

Property of VcResourceScheduler2

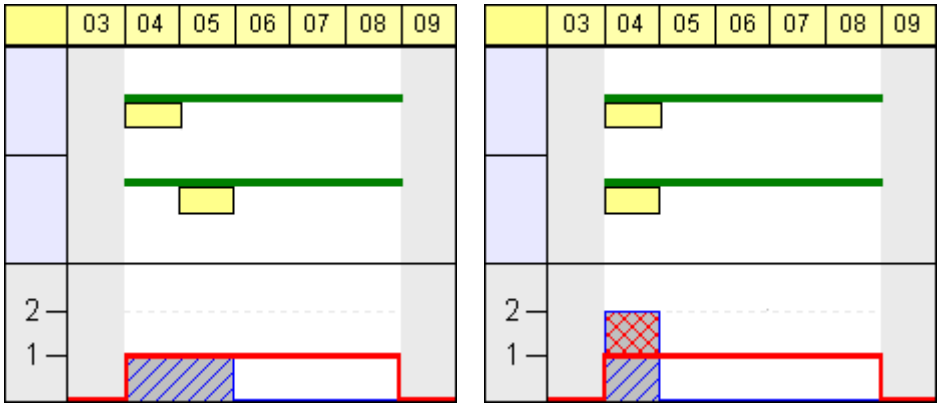
The index passed as the property value specifies a data field in the resource data table that defines the name of a calendar for a resource of the type **TimingResource** or **WorkResource**. If the field of the resource is empty, if it contains an invalid name or if this property is set to -1, as a substitute the name of the resource will be used for the calendar name, which is indirectly derived by the property **ResourceNameFieldIndex**.

	Data Type	Explanation
Property value	Long	Index of the data field in the resource data table that that defines a calendar name for the resource of the type **TimingResource** or **WorkResource** **Default value:** -1

ResourceCapacityType

Property of VcResourceScheduler2

This property specifies the capacity type for all resources, if it is not set individually for each resource by **ResourceCapacityTypeFieldIndex**.



Finite capacities (left) may require tasks to be allocated sequentially while infinite capacities (right) allow to schedule them simultaneously.

	Data Type	Explanation
Property value	ResourceCapacityType	Capacity types **Default value:** vcResSchedCTFinite
	Possible Values: vcResSchedCTFinite -1 vcResSchedCTInfinite 0	 Finite capacities Infinite capacities

Example Code

```
VcGantt1.ResourceScheduler2.ResourceCapacityType = vcResSchedCTFinite
```

ResourceCapacityTypeFieldIndex

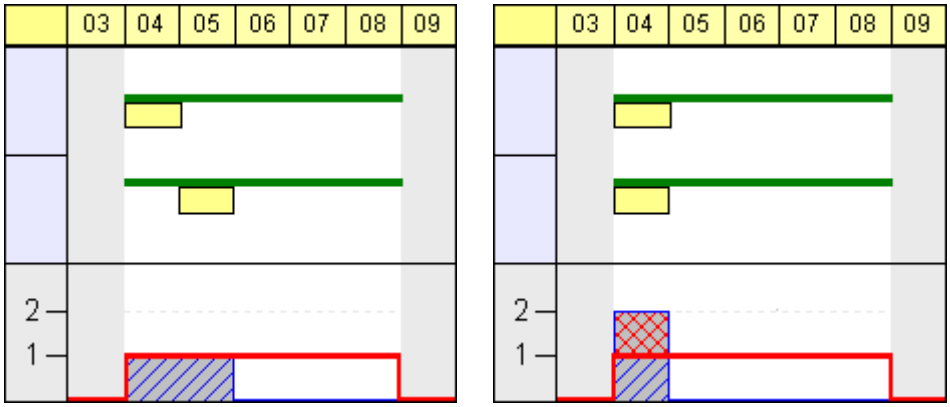
Property of VcResourceScheduler2

This property lets you set or retrieve the index of a field in a resource data table that holds the capacity type of a single timing resource. I In the picture referring to **ResourceDataTableName**, the field index for example is 2.

The index passed as a parameter denotes one out of 25 resource tables. The one to be used is defined by the indexed property **ResourceDataTableName**.

Permitted values of the data field content:

1 finite capacity 2 infinite capacity



Finite capacities (left) may require tasks to be allocated sequentially while infinite capacities (right) allow to work them off simultaneously. By the property **ResourceCapacityType** you can set the capacity for all data records of a resource table. The latter property is overwritten by this property if set.

If a resource belongs to more than one group, it has to have the same capacity type in all groups.

	Data Type	Explanation
Parameter: resourceTableIndex	Short	Index of the resource data table. {0...24}
Property value	Long	Index of the data field in the resource data table that is designated to hold the capacity type. {-1...NumberOfFieldsInResourceDataTable -1}. By setting the index to -1, no data field of the resource data table will be assigned to this property. **Default value:** -1

Example Code

```
VcGantt1.ResourceScheduler2.ResourceCapacityTypeFieldIndex(0) = 1
```

ResourceConstraintTypeFieldIndex

Property of VcResourceScheduler2

The index passed as the property value specifies a data field in the resource data table that holds a constraint for a single work or material resource.

Among the 25 possibly existing recource tables the one sought for is referred to by the index passed as the parameter.

As types, the values 0,1 or 3 or no value may be specified.

The values "" or "1" or no field indicate, that the given capacity of the resource is truely valid (this is what is called a "hard" resource).

The value "0" indicates, that the given capacity of the resource may be ignored if there is an increasing demand for it, since it then would be available by an unlimited capacity ("soft" resource).

The value "3" indicates that the resource is "hard", but workfree periods will be taken into account which do not cause interruptions when the operation is scheduled.

	Data Type	Explanation
Parameter: index	Short	Index of the resource data table {0...24}
Property value	Long	Index of the data field in the resource data table that is designated to hold the constraint data. {-1...NumberOfFieldsInResourceDataTable -1}. By setting the index to -1, no data field of the resource data table will be assigned to this property. **Default value:** -1

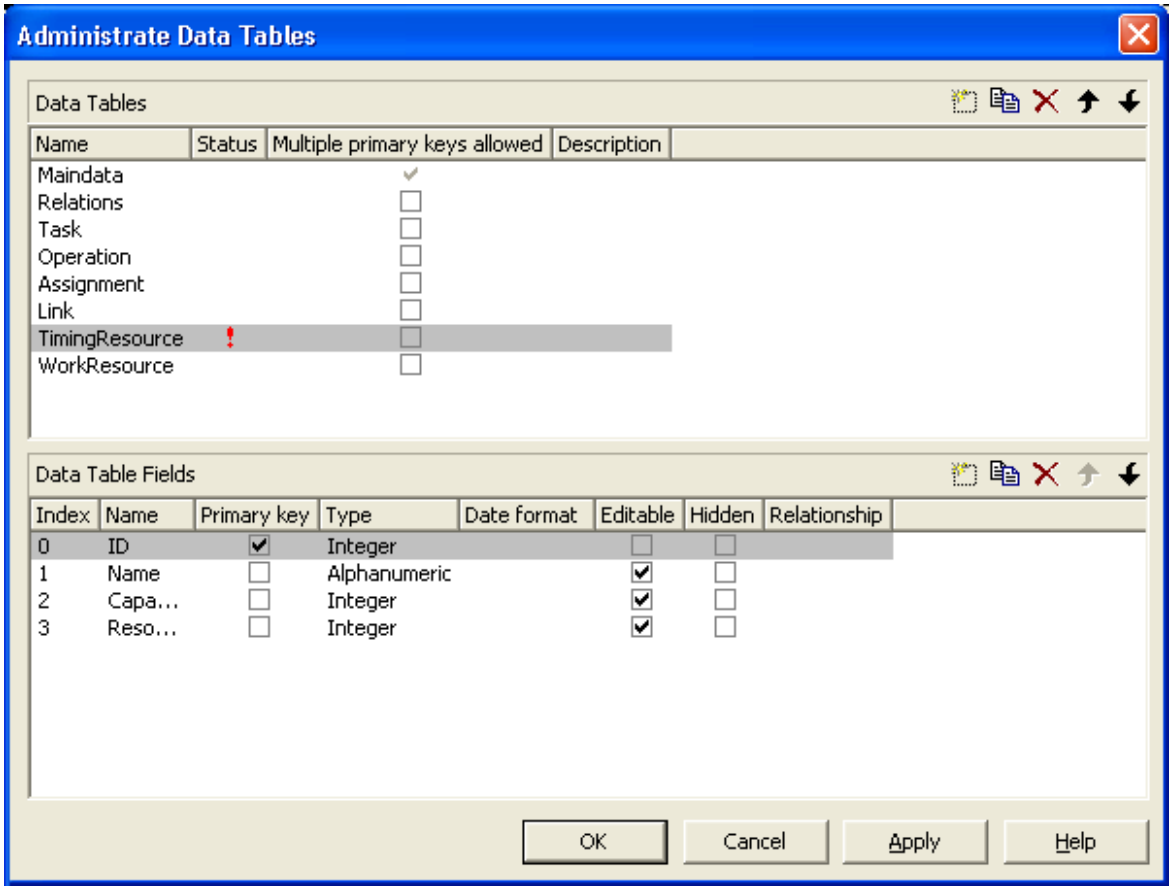
Example Code

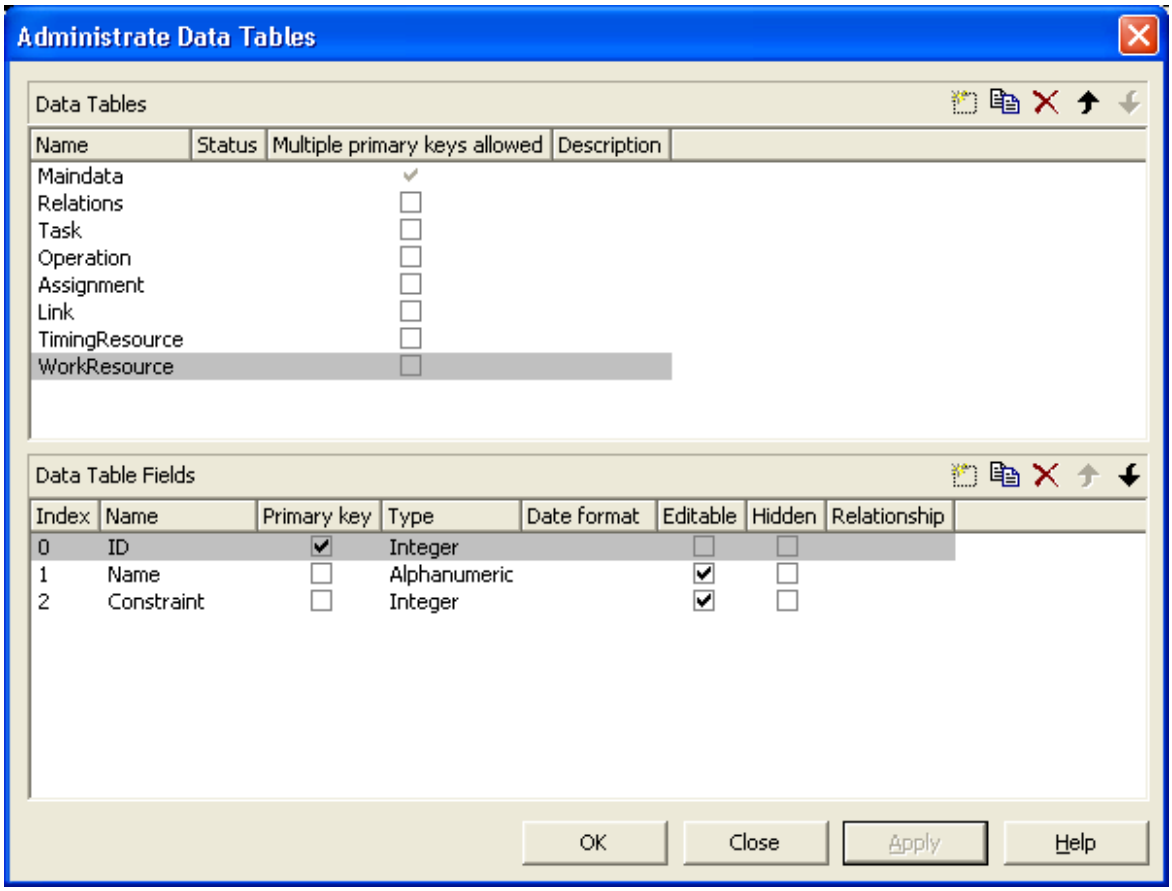
```
VcGantt1.ResourceScheduler2.ResourceConstraintTypeFieldIndex(0) = 1
```

ResourceDataTableName

Property of VcResourceScheduler2

This property lets you set or retrieve the names of up to 25 resource data tables. The name at the index 0 is to be set by obligation. If more than one name is set, the indices need to be stocked continuously without a gap from 0 onward. For each resource data table set by this property a corresponding field has to be allocated in the assignment data table by the property **AssignmentResourceIDFieldIndex**.


Administrate Data Tables
Data Tables
Name
Status
Multiple primary keys allowed
Description
Maindata
Relations
Task
Operation
Assignment
Link
TimingResource
WorkResource
Data Table Fields
Index
Name
Primary key
Type
Date format
Editable
Hidden
Relationship
0 ID Integer
1 Name Alphanumeric
2 Capa... Integer
3 Reso... Integer
OK
Cancel
Apply
Help


Administrate Data Tables
Data Tables
Name
Status
Multiple primary keys allowed
Description
Maindata
Relations
Task
Operation
Assignment
Link
TimingResource
WorkResource
Data Table Fields
Index
Name
Primary key
Type
Date format
Editable
Hidden
Relationship
0 ID Integer
1 Name Alphanumeric
2 Constraint Integer
OK
Close
Apply
Help

	Data Type	Explanation
Parameter: resourceTableIndex	Short	Index of the resource data table. {0...24}
Property value	String **Possible Values:**	Name of the data table **Default value:** Empty string Name of the color map

Example Code

```
VcGantt1.ResourceScheduler2.ResourceDataTableName(1) = "Timing Resource"
```

ResourceEfficiencyFieldIndex

Property of VcResourceScheduler2

The index passed as the property value specifies a data field in the resource data table that indicates an efficiency in percent for the resource of the type **TimingResource**. If this field of a resource is empty or if the property is set to -1, the efficiency by default equals 100. If however a value is set, the total of the allocations is multiplied by the efficiency value by assigning before scheduling this resource. So if the efficiency is lower than 100 per cent, an operation assigned to this resource will take longer than the default wheras values above 100 per cent will cause an assigned operation to be worked off faster than could the default. This is particularly interesting regarding the definition of resource groups (please see also **ResourceSelectionStrategy**), where from the available resources the one of greatest efficiency can be selected.

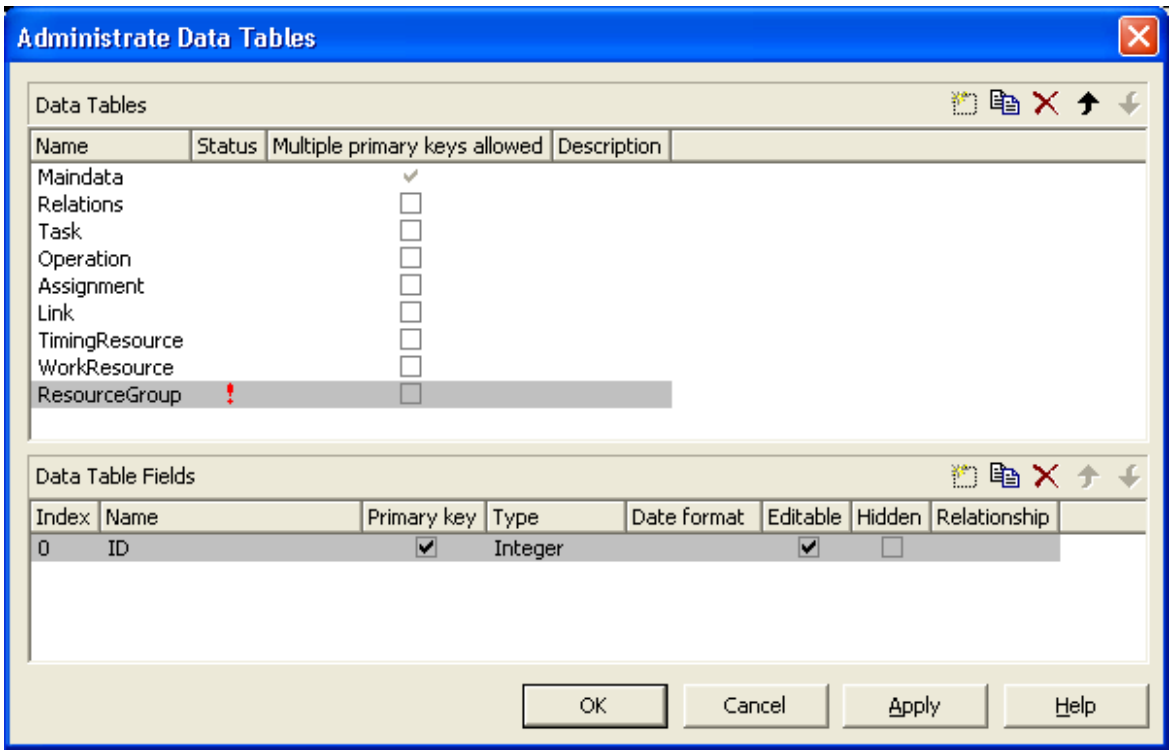
Being a percentage, the values of efficiency in general range between 1 and 100. Values of > 1,000 automatically will be put back to 1,000. The efficiency should NOT be set to a value as high as to reduce the occupation of a resource below 1.

	Data Type	Explanation
Property value	Long	Index of the data field in the resource data table that indicates the efficiency in per cent of a resource of the type **Timing Resource**. **Default value:** -1

ResourceGroupDataTableName

Property of VcResourceScheduler2

This indexed property lets you set or retrieve the data table in which the resource groups can be found, of which the IDs are held by fields referred to by **ResourceGroupIDFieldIndex**. So for each field index that you specify by the property **ResourceGroupIDFieldIndex**, you need to set the name of a data field by this property, which uses the same data tables as does **ResourceDataTableName**. The resource data table index passed as the parameter denotes one out of 25 available resource data tables assigned by the indexed property **ResourceDataTableName**.



	Data Type	Explanation
Parameter: resourceGroupTableIndex	Short	Index of the resource group data table. {0...24}
Property value	String **Possible Values:**	Name of the resource group data table Name of the color map

Example Code

```
VcGantt1.ResourceScheduler2.ResourceGroupDataTableName(1) = "Printer Resource"
```

ResourceGroupIDFieldIndex

Property of VcResourceScheduler2

The index passed as the property value specifies a data field in the resource data table that is designated to hold the ID of a group resource. By setting the ID, the resource is described as one belonging to the group. In the picture referring to **ResourceGroupDataTableName**, the field index for example is 0. If the field index is set to -1 or if the resource data field referred to is empty, the resource will not belong to a group. This property must only be set to timing resources (see property **ResourceType**).

The index passed as a parameter denotes one out of 25 resource tables. They can be set by the indexed property **ResourceDataTableName**.

	Data Type	Explanation
Parameter: resourceTableIndex	Short	Index of the resource data table {0...24}
Property value	Long	Index of the data field in the resource data table that is designated to hold the groupID. {-1...NumberOfFieldsInResourceDataTable -1}. By setting the index to -1, no data field of the resource data table will be assigned to this property. **Default value:** -1

Example Code

```
VcGantt1.ResourceScheduler2.ResourceGroupIDFieldIndex(0) = 1
```

ResourceNameFieldIndex

Property of VcResourceScheduler2

The index passed as the property value specifies a data field in the resource data table that holds the names of resources. In the picture referring to **ResourceDataTableName**, the field index for example is 1.

The resource name serves to identify histogram names, curve names and calendar names. Beside, it is used with groups to allocate a resource to several groups simultaneously. For this, a resource in different data records needs to be specified by the same name but by different IDs of the group resources. If no field index is specified, names of histograms, curves and calendars will be retrieved on the base of the resource ID.

The index passed as a parameter denotes one out of 25 resource tables. The resource tables used can be set by the indexed property **ResourceDataTable-Name**.

	Data Type	Explanation
Parameter: resourceTableIndex	Short	Index of the resource data table. {0...24}
Property value	Long	Index of the data field in the resource data table that is designated to hold the name. {-1...NumberOfFieldsInResourceDataTable -1}. By setting the index to -1, no data field of the resource data table will be assigned to this property. **Default value:** -1

Example Code

```
VcGantt1.ResourceScheduler2.ResourceNameFieldIndex(0) = 1
```

ResourceResultLoadCurveNamePrefix

Property of VcResourceScheduler2

Prefix for the name of the curve that after the scheduling procedure contains the resource capacity for each timing resource and for each work and material resource.

The curves for the work load need to have been defined before invoking the method **Process**, otherwise they cannot be visualized. The resource name or the resource ID will be used to form the remaining part of the name (see property **ResourceMameFieldIndex**). If a curve is not found, the results of the work load will be lost for the resource affected.

Beside, the property **CurveSource** needs to have been set to **vcSetCurve** for the curves, i.e. assignments must be feasible by the **VcCurve.SetValues** method of the API.

	Data Type	Explanation
Property value	String **Possible Values:**	Character string that contains the prefix **Default value:** "Load_" Name of the color map

Example Code

```
VcGantt1.ResourceScheduler2.ResourceResultLoadCurveNamePrefix = "LoadCurve_"
```

ResourceResultStockCurveNamePrefix

Property of VcResourceScheduler2

Prefix for the name of the curve that after the scheduling procedure contains the available stock of each material resource.

The stock curves need to have been defined before invoking the method **Process**, otherwise they cannot be visualized. The resource name or the resource ID will be used to form the remaining part of the name.

If a curve is not found, the results of the stock will be lost for the resource affected. The availabe stock is calculated from the cumulation of material supply (that is, from the supply curve that has to be put up before the scheduling procedure starts) and from the utilization by the operations that were assigned to the resource.

Beside, the property **CurveSource** needs to have been set to **vcSetCurve** for the curves, i.e. assignments must be feasible by the **VcCurve.SetValues** method of the API.

	Data Type	Explanation
Property value	String	Character string that contains the prefix **Default value:** "Stock_"
	Possible Values:	Name of the color map

Example Code

```
VcGantt1.ResourceScheduler2.ResourceResultStockCurveNamePrefix = 1
```

ResourceSelectionStrategy

Property of VcResourceScheduler2

This property specifies the selection strategy of the scheduling process for resources to be selected from a group (therefore for timing resources only).

	Data Type	Explanation
Property value	VcResourceSchedulingResource-SelectionStrategyEnum	Selection types **Default value:** VcResSchedRSSequential
	Possible Values:	
	vcResSchedRSFirstAvailable 6	The resource which is first available when the scheduling is performed will be selected if its available capacity permits. When using this constant, the selection merely depends on the first timing resource. Other assignments of the operation are not taken into account. So when using material and work resources, the results may not turn out satisfactorily.
	vcResSchedRSHighestEfficiency 2	The resource most efficient when the scheduling is performed will be selected (makes sense only if the property **Resource-EfficiencyFieldIndex** is used) if its available capacity permits.
	vcResSchedRSLeastLoaded 0	The resource least loaded when the scheduling is performed will be selected, if its availabe capacity permits. This strategy is useful if the workload is to be distributed evenly between resources. This value entails consecutive adding of resource occupation that forms the base for selecting the resource least loaded. So if planning periods of tasks differ widely or if both planning strategies are applied, the results may not prove satisfactory.
	vcResSchedRSMostLoaded 1	The resource most loaded when the scheduling is performed will be selected, if its available capacity permits. This strategy is useful if the workload is to be concentrated on as few resources as possible. This value entails consecutive adding of resource occupation that forms the base for selecting the resource most loaded. So if planning periods of tasks differ widely or if both planning strategies are applied, the results may not prove satisfactory.
	vcResSchedRSSequential -1	The resources are tried to be used in the sequence defined.

Example Code

```
VcGantt1.ResourceScheduler2.ResourceSelectionStrategy = vcResSchedRSLeastLoaded
```

ResourceType

Property of VcResourceScheduler2

This property lets you set or retrieve the type of a resource data table. The index passed specifies one of the 25 possibly existing resource data tables. Three possible resource types exist:

1. **Timing Resources**

For a resource to time an operation, the operation needs to be assigned to exactly one resource. Both, finite and infinite capacity types are permitted (s. property **ResourceCapacityTypeFieldIndex**). Resources of this type can be grouped (s. properties **ResourceGroupDataTableName** and **ResourceGroupIDFieldIndex**). Beside, the work load of the resource can be limited (s. properties **AssignmentMinimumLoadFieldIndex** and **AssignmentMaximumLoadFieldIndex**). A timing resource requires capacity curves as an indirect resource information and uses work load curves to put the results (s. properties **ResourceNameFieldIndex** and **ResourceResultLoadCurvePrefix).**

2. Work Resources

This resource type shows two particular features. An operation can be assigned to more than one resource of this type. As the timing type, the work type requires capacity curves as an indirect source of information and uses work load curves to put the results (s. properties **ResourceNameFieldIndex** and **ResourceResultLoadCurvePrefix**).

3. **Material Resources**

The material resource also shows two characteristic features. An operation can be assigned to more than one resource of this type. The material resource differs by its source of indirect information and the result output: it requires supply curves as an indirect source of information and uses stock curves to put the results (s. properties **ResourceNameFieldIndex** and **ResourceResultStockCurvePrefix**).

	Data Type	**Explanation**
Parameter: resourceTableIndex	Short	Index of the resource data table. {0...24}.
Property value	VcResourceSchedulingResourceTypeEnum **Possible Values:** vcResSchedMaterial -1 vcResSchedTiming 1 vcResSchedWork 0	Type of the resource data table **Default value:** vcTiming The resource type is "material". The resource type is "timing". The resource type is "work".

Example Code

```
VcGantt1.ResourceScheduler2.ResourceType(0) = vcResSchedTiming
```

ResultProcessingStepCount

Property of VcResourceScheduler2

This property provides the number of scheduled operations in the chart after a scheduling procedure.

	Data Type	Explanation
Property value	Long	Number of tasks **Default value:** 0

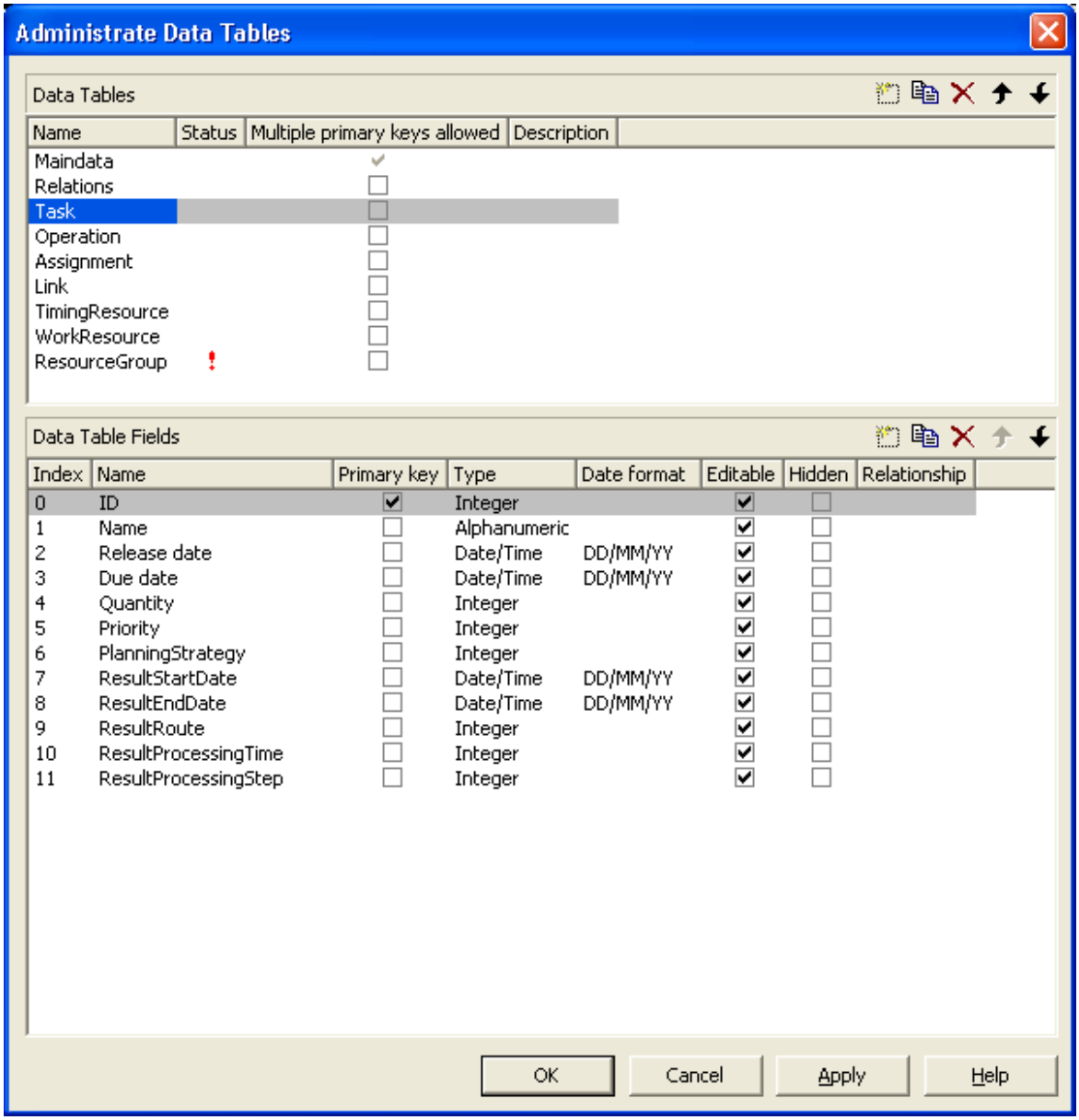
Example Code

```
Dim i As Integer
i = VcGantt1.ResourceScheduler2.ResultProcessingStepCount
```

TaskDataTableName

Property of VcResourceScheduler2

This property lets you set or retrieve the name of the task data table. A valid table name has to be used with the property.



	Data Type	Explanation
Property value	String **Possible Values:**	Name of the task data table **Default value:** Empty string Name of the color map

Example Code

```
VcGantt1.ResourceScheduler2.TaskDataTableName = "Task"
```

TaskDueDateFieldIndex

Property of VcResourceScheduler2

The index specifies a data field in the tasks data table which holds the due date at which a task must be finished. If no valid value is found in the data field, the value set by the VcGantt property **TimeScaleEnd** will be used. If you wish the task to be scheduled, the value of this property must not be set to -1. In the picture referring to **TaskDataTableName**, the field index for example is 3.

To due dates, a general allowance can be set by the property **ToleranceTime-OnASAPDueDates** Please mind that tasks that have a close due date or only a short period between the release date and the due date are not scheduled automatically by first priority. If you wish this to happen, you need to calculate the priorities of the tasks manually (see property **TaskPriority-FieldIndex**).

	Data Type	Explanation
Property value	Long	Index of the data field in the task data table that is designated to hold the due date. {-1...NumberOfFieldsInTaskDataTable -1}. By setting the index to -1, no data field of the task data table will be assigned to this property. **Default value:** -1

Example Code

```
VcGantt1.ResourceScheduler2.TaskDueDateFieldIndex = 3
```

TaskPlanningStrategyFieldIndex

Property of VcResourceScheduler2

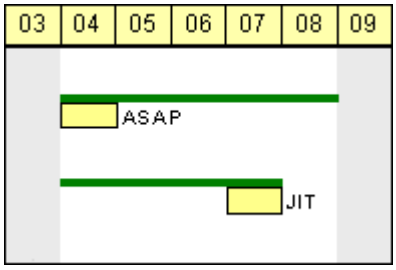
The index specifies a data field which holds an individual planning strategy for a task.

If no value is set or if the value is < 1 or > 2, the value set by the property **Planning Strategy** will be used. In the picture referring to **TaskDataTable-Name**, the field index for example is 6.

Defined values of data fields {1...2}:

1 - ASAP: as soon as possible

2 - JIT: just in time



In the ASAP strategy a task is scheduled early, while in the JIT strategy it is scheduled late. The long slim bars show the available period to complete a task, while the short big bars represent the actually allocated time for completion. So ASAP tasks tend to appear at the left end of the available period of completion, while JIT tasks tend to appear at its right end.

	Data Type	Explanation
Property value	Long	Index of the data field in the task data table that is designated to hold the data of the planning strategy. {-1...NumberOfFieldsInTaskDataTable -1}. By setting the index to -1, no data field of the task data table will be assigned to this property. **Default value:** -1

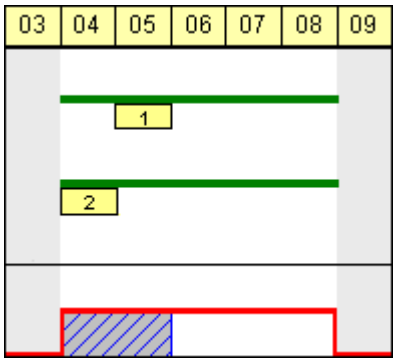
Example Code

```
VcGantt1.ResourceScheduler2.TaskPlanningStrategyFieldIndex = 6
```

TaskPriorityFieldIndex

Property of VcResourceScheduler2

The index specifies a data field in the task data table which holds a priority for a task. The higher the priority value, the better the activity is positioned in the queue of scheduling.



A priority 2 task will be scheduled before a priority 1 task.

Please note: If tasks are linked, their priorities should be set very carefully. When using the ASAP strategy, predecessors should have the same priority as their successors; when using the JIT strategy, predecessors should have at least the same priority as their successors. Tasks can be grouped by their priorities. For example, when grouping tasks of equal priority, preparation and cleaning times of the device may be saved.

	Data Type	Explanation
Property value	Long	Index of the data field in the task data table that is designated to hold the priority. {-1...NumberOfFieldsInTaskDataTable -1}. By setting the index to -1, no data field of the task data table will be assigned to this property. **Default value:** -1

Example Code

```
VcGantt1.ResourceScheduler2.TaskPriorityFieldIndex = 5
```

TaskQuantityFieldIndex

Property of VcResourceScheduler2

The index specifies a data field in the tasks data table which holds the quantity to be worked off by a task. The value of this property must not be set to -1.

The quantity indirectly influences the amount of time required by the task to finish. The amount of time can also be influenced by the efficiency of the resources (see **ResourceEfficiencyFieldIndex**), by multipliers of operations (see **OperationLoadPerItemFieldIndex**) and of assignments (see **AssignmentLoadOrConsumptionPerItemFieldIndex**).

If no valid value is found in the data field, a quantity of 1 will be assumed. In the picture referring to **TaskDataTableName**, the field index for example is 4.

	Data Type	Explanation
Property value	Long	Index of the data field in the task data table that is designated to hold the quantity. {-1...NumberOfFieldsInTaskDataTable -1}. By setting the index to -1, no data field of the task data table will be assigned to this property. **Default value:** -1

Example Code

```
VcGantt1.ResourceScheduler2.TaskQuantityFieldIndex = 4
```

TaskReleaseDateFieldIndex

Property of VcResourceScheduler2

The index specifies a data field in the tasks data table which holds the release date from which onward a task can be scheduled. The value of this property must not be set to -1.

If no valid value is found in the data field, the value set by the VcGantt property **TimeScaleStart** will be used. In the picture referring to **TaskDataTableName**, the field index for example is 2.

You can set a general allowance to release dates by the property **ToleranceTimeOnJITReleaseDates**.

	Data Type	Explanation
Property value	Long	Index of the data field in the task data table that is designated to hold the release date. {-1...NumberOfFieldsInTaskDataTable -1}. By setting the index to -1, no data field of the task data table will be assigned to this property. **Default value:** -1

Example Code

```
VcGantt1.ResourceScheduler2.TaskReleaseDateFieldIndex = 2
```

TaskResultEndDateFieldIndex

Property of VcResourceScheduler2

The index specifies a data field in the tasks data table which holds the calculated end date of the latest operation scheduled that is part of the task. In the picture referring to **TaskDataTableName**, the field index for example is 8.

	Data Type	Explanation
Property value	Long	Index of the data field in the task data table that is designated to hold the end date. {-1...NumberOfFieldsInTaskDataTable -1}. By setting the index to -1, no data field of the task data table will be assigned to this property. **Default value:** -1

Example Code

```
VcGantt1.ResourceScheduler2.TaskResultEndDateFieldIndex = 8
```

TaskResultPostEndDateFieldIndex

Property of VcResourceScheduler2

The index specifies a data field in the task data table which holds the scheduled end date of the post time of a task. If the post time equals 0, the date is identical to the value in the data field which is referred to by the property **TaskResultEndDateFieldIndex**.

	Data Type	Explanation
Property value	Long	Index of the data field in the task data table that is designated to hold the end date of the post time. {-1...NumberOfFieldsInTaskDataTable -1}. By setting the index to -1, no data field of the task data table will be assigned to this property. **Default value:** -1

Example Code

```
VcGantt1.ResourceScheduler2.TaskResultPostEndDateFieldIndex = 15
```

TaskResultPreparationStartDateFieldIndex

Property of VcResourceScheduler2

The index specifies a data field in the task data table which holds the scheduled start date of the preparation phase of a task. If the preparation phase is 0, this date is identical to the value in the data field which is referred to by the property **TaskResultStartDateFieldIndex**.

	Data Type	Explanation
Property value	Long	Index of the data field in the task data table that is designated to hold the start date of the preparation phase. {-1...NumberOfFieldsInTaskDataTable -1}. By setting the index to -1, no data field of the task data table will be assigned to this property. **Default value:** -1

Example Code

```
VcGantt1.ResourceScheduler2.TaskResultPreparationStartDateFieldIndex = 10
```

TaskResultProcessingStepFieldIndex

Property of VcResourceScheduler2

The index specifies a data field in the tasks data table which holds a sequence number by which the task was scheduled. This value is useful to recognize the first task that cannot be scheduled due to resource bottlenecks.

The task scheduled first will receive 0, the tasks following will receive the consecutive numbers in ascending order. In the picture referring to **TaskDataTableName**, the field index for example is 11.

	Data Type	Explanation
Property value	Long	Index of the data field in the task data table that is designated to hold the sequence number. {-1...NumberOfFieldsInTaskDataTable -1}. By setting the index to -1, no data field of the task data table will be assigned to this property. **Default value:** -1

Example Code

```
VcGantt1.ResourceScheduler2.TaskResultProcessingStepFieldIndex = 11
```

TaskResultProcessingTimeFieldIndex

Property of VcResourceScheduler2

The index specifies a data field in the tasks data table which holds the calculated total processing time of the operations that form the task and that were scheduled. It is the time span between the start date of the first operation and the final date of the last operation. Units: as set by the base time unit. In the picture referring to **TaskDataTableName**, the field index for example is 10.

	Data Type	Explanation
Property value	Long	Index of the data field in the task data table that is designated to hold the processing time. {-1...NumberOfFieldsInTaskDataTable -1}. By setting the index to -1, no data field of the task data table will be assigned to this property. **Default value:** -1

Example Code

```
VcGantt1.ResourceScheduler2.TaskResultProcessingTimeFieldIndex = 10
```

TaskResultRouteFieldIndex

Property of VcResourceScheduler2

The index specifies a data field in the tasks data table which holds the name of a route that was selected for the task by the scheduling procedure.

The value of this property should be set to a value different from -1, if the property **OperationRouteFieldIndex** is also used. In the picture referring to **TaskDataTableName**, the field index for example is 9.

	Data Type	Explanation
Property value	Long	Index of the data field in the task data table that is designated to hold the name of the route. {-1...NumberOfFieldsInTaskDataTable -1}. By setting the index to -1, no data field of the task data table will be assigned to this property. **Default value:** -1

Example Code

```
VcGantt1.ResourceScheduler2.TaskResultRouteFieldIndex = 9
```

TaskResultStartDateFieldIndex

Property of VcResourceScheduler2

The index specifies a data field in the tasks data table which holds the calculated start date of the earliest operation scheduled that is part of the task. In the picture referring to **TaskDataTableName**, the field index for example is 7.

	Data Type	Explanation
Property value	Long	Index of the data field in the task data table that is designated to hold the start date. {-1...NumberOfFieldsInTaskDataTable -1}. By setting the index to -1, no data field of the task data table will be assigned to this property. **Default value:** -1

Example Code

```
VcGantt1.ResourceScheduler2.TaskResultStartDateFieldIndex = 7
```

ToleranceTimeOnASAPDueDates

Property of VcResourceScheduler2

By this property you can set or retrieve an allowance to due dates. It only works with the ASAP planning strategy. The unit equals the one set by the property **BaseTimeUnit**.

During the scheduling procedure, the due dates of the tasks are postponed by the number of units set by this property, prolonging the period of time allowed to a task. This property is useful to detect whether after enlarging the scheduling period all operations and tasks could be scheduled. It saves you from modifying and testing tasks individually.

Please also see **ToleranceTimeOnJITReleaseDates**.

	Data Type	Explanation
Property value	Long	Number of base time units {>=0} **Default value:** 0

Example Code

```
VcGantt1.ResourceScheduler2.ToleranceTimeOnASAPDueDates = 1
```

ToleranceTimeOnJITReleaseDates

Property of VcResourceScheduler2

By this property you can set or retrieve a variation allowed to release dates. This setting only works if the JIT planning strategy is set. The unit equals what was set by the property **BaseTimeUnit**.

During the scheduling procedure, the release dates of the tasks are put earlier by the number of units set by this property, prolonging the period of time

allowed to a task. This property is useful to detect what scheduling periods are needed for all tasks to be scheduled. It saves you from modifying the release dates of tasks individually.

Please also see **ToleranceTimeOnASAPDueDates**.

	Data Type	Explanation
Property value	Long	Number of base time units {>=0} **Default value:** 0

Example Code

```
VcGantt1.ResourceScheduler2.ToleranceTimeOnJITReleaseDates = 1
```

ToleranceTimeOnStartLockDates

Property of VcResourceScheduler2

By this property you can set or retrieve an allowance to a locked start date of an operation (see **OperationStartLockDateFieldIndex**). Its unit equals the one set by the property **BaseTimeUnit**.

During the scheduling procedure, an operation can be postponed by the number of units set by this property, if the resources to be occupied are not available at the lock start date.

	Data Type	Explanation
Property value	Long	Number of base time units {>=0} **Default value:** 0

Example Code

```
VcGantt1.ResourceScheduler2.ToleranceTimeOnStartLockDatess = 1
```

WorkInProcessType

Property of VcResourceScheduler2

This property sets the unit to specify the degree of completion (please see **OperationWorkInProcessFieldIndex**).

	Data Type	Explanation
Property value	ResourceSchedulingWorkInProcessTypeEnum **Possible Values:**	Unit of the degree of completion **Default value:** vvcResSchedWIPPercentage

vcResSchedWIPCompleted 0	Unit: quantity already completed
vcResSchedWIPPercentage -1	Unit: percentage (0...100)
vcResSchedWIPRemaining 1	Unit: quantity to be completed

Example Code

```
VcGantt1.ResourceScheduler2.WorkInProcessType = vcResSchedWIPCompleted
```

WritingDebugFilesEnabled

Property of VcResourceScheduler2

If this property is set to **True**, debug files can be stored to the directory of the application, which may be useful for error analysis.

	Data Type	Explanation
Property value	Boolean	**True**: debug files can be written into the current directory. **False**: debug files cannot be written into the current directory. **Default value:** False
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.ResourceScheduler2.WritingDebugFilesEnabled = True
```

Methods

DetermineIDOfFirstOperationByTaskID

Method of VcResourceScheduler2

This method determines the ID of the first operation of a task by the given TaskID and helps the developer updating the data field of a link which contains the first operation of a task.

For further information please see the description of the VcResourceScheduler2 properties **LinkPredecessorOperationIDFieldIndex** and **LinkSuccessorOperationIDFieldIndex**.

	Data Type	Explanation
Parameter:		
⇨ taskID	String **Possible Values:**	ID of a task of the corresponding data table which was set by the VcResourceScheduler2 property **TaskDataTableName**. Name of the color map
Return value	String	ID of the first operation of the corresponding data table which was set by the VcResourceScheduler2 property **OperationDataTableName**.

DetermineIDOfLastOperationByTaskID

Method of VcResourceScheduler2

This method determines the ID of the last operation of a task by the given TaskID and helps the developer updating the data field of a link which contains the last operation of a task.

For further information please see the description of the VcResourceScheduler2 properties **LinkPredecessorOperationIDFieldIndex** and **LinkSuccessorOperationIDFieldIndex**.

	Data Type	Explanation
Parameter:		
taskID	String **Possible Values:**	ID of a task of the corresponding data table which was set by the VcResourceScheduler2 property **TaskDataTableName**. Name of the color map
Return value	String	ID of the last operation of the corresponding data table which was set by the VcResourceScheduler2 property **OperationDataTableName**.

Process

Method of VcResourceScheduler2

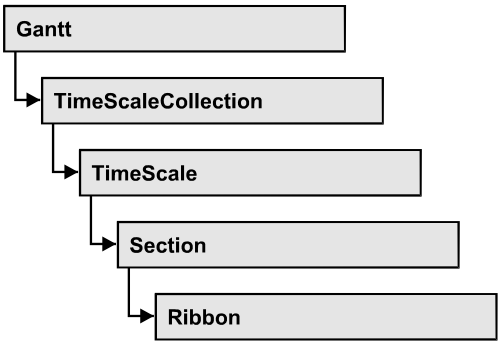
This method starts the scheduling procedure after the desired properties were set. For messages on the progress please also see **OnResourceScheduling-Progress**. Beside, warnings are put out by **ResourceSchedulingWarning**.

	Data Type	Explanation
Return value	Boolean	**True**: No error occurred during the scheduling procedure. **False**: An error occurred or the scheduling procedure was abandoned. If allowed by the settings, error codes are stored for each job by the data field that is addressed by the property **OperationResultStatusFieldIndex** .

Example Code

```
VcGantt1.ResourceScheduler2.Process
```

7.71 VcRibbon



An object of the type VcRibbon represents a defined ribbon in the time scale of homogeneous units and scaling. You can set the background color, the type of unit separation, font type, color, size, alignment and other attributes to a ribbon.

Properties

- CalendarName
- DateOutputFormat
- Font
- FontColor
- MajorTicks
- MinorTicks
- ObserveDST
- PatternBackgroundColorAsARGB
- PatternColorAsARGB
- PatternEx
- Position
- ReferenceDate
- TextAlignment
- TickColor
- TickPosition
- Type
- UnitSeparation
- UseReferenceDate

Properties

CalendarName

Property of VcRibbon

This property lets you set or retrieve the calendar name.

	Data Type	Explanation
Property value	String	Calendar name
	Possible Values:	Name of the color map

DateOutputFormat

Property of VcRibbon

This property lets you specify the date output format of a ribbon. To compose the date you can use the below codes:

D: first letter of the day of the week (not adjustable)

TD: Day of the Week (adjustable by using the event **OnSupplyTextEntry**)

DD: two-digit figure for the day of the month: 01-31

DDD: first three letters of the day of the week (not adjustable)

M: first letter of the name of the month (not adjustable)

TM: name of the month (adjustable by using the event **OnSupplyTextEntry**)

MM: two-digit figure for the month: 01-12

MMM: first three letters of the name of the month (not adjustable)

YY: two-digit figure for the year

YYYY: four-digit figure for the year

WW: two-digit figure for the number of the calendar week: 01-53

TW: text for "calendar week" (adjustable by using the event **OnSupplyTextEntry**)

Q: one-digit figure for the quarter: 1-4

TQ: name of quarter (adjustable by using the event **OnSupplyTextEntry**)

hh: two-digit figure for the hour in 24 hours format: 00-23

HH: two-digit figure for the hour in 12 hours format: 01-12

Th: Text of "o' clock" (adjustable by using the event **OnSupplyTextEntry**)

TH: "am" or "pm" (adjustable by using the event **OnSupplyTextEntry**)

mm two-digit figure for the minute: 00-59

ss: two-digit figure for the second: 00-59

TS: short date format, as defined in the regional settings of the windows control panel

TL: long date format, as defined in the regional settings of the windows control panel

TT: time format, as defined in the regional settings of the windows control panel

xC/XC: You can set a maximum ten-place, simple upward counting from a reference date onward, for example "15:05:07:16:00", which equals 15 months, 5 days, 7 hours, 16 minutes, 0 seconds. The notation is: **xC44:C33:C22:C11:C00**. In written language: Show at least 2 digits for the counters 4...0 and a preceding "-" symbol if the value is negative. The separators are variable and can be replaced by other separators symbols. "x" means: Display a preceding "-" symbol if the value is negative, but no "+" symbol if it is positive. "X" means: Display a preceding "-" symbol if the value is negative and a "+" symbol for positive values. In the dialog **Edit Time Scale Section...** the check boxes **Use reference date** and **Adjust major ticks to reference date** need to be ticked, also, the parameter **Serial annotation** has to be set to **No**. In the application the reference date is set at run time by the call **VcRibbon.set ReferenceDate**, overriding any settings in the dialog.

Note: Characters which are not to be interpreted as part of the date should be preceded by a backslash '\'. '\\' for instance results in '\'. The special characters: **':, /, -'** and **blank** don't need '\' as prefix

	Data Type	Explanation
Property value	String **Possible Values:**	Date format {DMYhms:;/} or {01234CXx} Name of the color map

Example Code

```
Dim timeScale As VcTimeScale
Dim ribbon As VcRibbon

Set timeScale = VcGantt1.TimeScaleCollection.Active
Set ribbon = timeScale.Ribbon(0, 0)
ribbon.DateOutputFormat = "DD.MMM.YYYY hh:mm:ss"
```

Font

Property of VcRibbon

This property lets you set or retrieve all font attributes of the ribbon.

	Data Type	Explanation
Property value	StdFont	Font attributes of the ribbon

Example Code

```
Dim timeScale As VcTimeScale
Dim ribbon As VcRibbon
Dim newFont As New StdFont

newFont.Name = "Times New Roman"
newFont.Italic = True
newFont.Bold = True
newFont.Size = 12

Set timeScale = VcGantt1.timeScaleCollection.Active
Set ribbon = timeScale.ribbon(0, 0)
Set ribbon.Font = newFont
```

FontColor

Property of VcRibbon

This property lets you set or retrieve the font color of the ribbon.

	Data Type	Explanation
Property value	Color	RGB color values

Example Code

```
Dim timeScale As VcTimeScale
Dim ribbon As VcRibbon
```

```
Set timeScale = VcGantt1.TimeScaleCollection.Active
Set ribbon = timeScale.Ribbon(0, 0)
ribbon.FontColor = RGB(240, 130, 220)
```

MajorTicks

Property of VcRibbon

This property lets you set or retrieve after how many time units a major tick is drawn. The time unit depends on the ribbon type used. The major ticks are labelled when there is enough space. This property you can also set in the **Edit Time Scale Section** dialog.

	Data Type	Explanation
Property value	Integer **Possible Values:**	Number of units between two major ticks Data field index

Example Code

```
Dim timeScale As VcTimeScale
Dim ribbon As VcRibbon

Set timeScale = VcGantt1.TimeScaleCollection.Active
Set ribbon = timeScale.Ribbon (0, 1)
ribbon.MajorTicks = 7
```

MinorTicks

Property of VcRibbon

This property lets you set or retrieve after how many time units a minor tick is drawn. The time unit depends on the ribbon type used. The minor ticks are not labelled. This property you can also set in the **Edit Time Scale Section** dialog.

	Data Type	Explanation
Property value	Integer **Possible Values:**	Number of units between two minor ticks Data field index

Example Code

```
Dim timeScale As VcTimeScale
Dim ribbon As VcRibbon

Set timeScale = VcGantt1.TimeScaleCollection.Active
Set ribbon = timeScale.Ribbon (0, 1)
ribbon.MinorTicks = 1
```

ObserveDST

Read Only Property of VcRibbon

This property lets you set or retrieve whether for this ribbon daylight saving time is considered or not.

	Data Type	Explanation
Property value	RibbonObserveDSTEnum	Daylight saving time is/is not considered.
	Possible Values:	
	vcGODDefault 9999	Default setting from .INI file is used
	vcRODNo 0	Daylight saving time is not considered
	vcRODYes 1	Daylight saving time is considered

PatternBackgroundColorAsARGB

Property of VcRibbon

This property lets you set or retrieve the background color of the ribbon. Color values have a transparency or alpha value, followed by a value for a red, a blue and a green partition (ARGB). The values range between 0..255. An alpha value of 0 equals complete transparency, whereas 255 represents a completely solid color. When casting an RGB value on an ARGB value, an alpha value of 255 has to be added.

	Data Type	Explanation
Property value	Long	ARGB color values ({0...255},{0...255},{0...255},{0...255})

Example Code

```
Dim timeScale As VcTimeScale
Dim ribbon As VcRibbon

Set timeScale = VcGantt1.TimeScaleCollection.Active
Set ribbon = timeScale.Ribbon(0, 0)
ribbon.PatternBackgroundColorAsARGB = &h88FF0A06
```

PatternColorAsARGB

Property of VcRibbon

This property lets you set or retrieve the pattern color of the ribbon. Color values have a transparency or alpha value, followed by a value for a red, a blue and a green partition (ARGB). The values range between 0..255. An alpha value of 0 equals complete transparency, whereas 255 represents a

completely solid color. When casting an RGB value on an ARGB value, an alpha value of 255 has to be added.

	Data Type	Explanation
Property value	Integer **Possible Values:**	Background color of ribbon Data field index

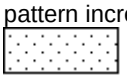
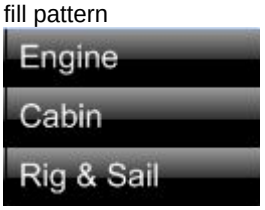
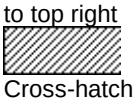
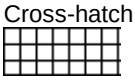
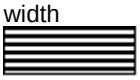
Example Code

```
Dim timeScale As VcTimeScale
Dim ribbon As VcRibbon

Set timeScale = VcGantt1.TimeScaleCollection.Active
Set ribbon = timeScale.Ribbon(0, 0)
ribbon.PatternColorAsARGB = &h88FF0A06
```

PatternEx

Property of VcRibbon

This property lets you set or retrieve the fill pattern type of the ribbon.

	Data Type	Explanation
Property value	FillPatternEnum	Pattern type
	Possible Values: vc05PercentPattern... vc90PercentPattern 01 - 11	Dots in foreground color on background color, the density of the foreground pattern increasing with the percentage 
	vcAeroGlassPattern 40	Vertical color gradient in the color of the fill pattern 
	vcBDiagonalPattern 5	Diagonal lines slanting from bottom left to top right 
	vcCrossPattern 6	Cross-hatch pattern 
	vcDarkDownwardDiagonalPattern 2014	Diagonal lines slanting from top left to bottom right; spaced 50% closer than vcFDiagonalPattern and of twice the line width 
	vcDarkHorizontalPattern 2023	Horizontal lines spaced 50% closer than vcHorizontalPattern and of twice the line width 

vcDarkUpwardDiagonalPattern 2015	Diagonal lines slanting from bottom left to top right, spaced 50% closer than vcBDiagonalPattern and of twice the line width
vcDarkVerticalPattern 2022	Vertical lines spaced 50% closer than vcVerticalPattern and of of twice the line width
vcDashedDownwardDiagonalPattern 2024	Dashed diagonal lines from top left to bottom right
vcDashedHorizontalPattern 2026	Dashed horizontal lines
vcDashedUpwardDiagonalPattern 2025	Dashed diagonal lines from bottom left to top right
vcDashedVerticalPattern 2027	Dashed vertical lines
vcDiagCrossPattern 7	Diagonal cross-hatch pattern, small
vcDiagonalBrickPattern 2032	Diagonal brick pattern
vcDivotPattern 2036	Divot pattern
vcDottedDiamondPattern 2038	Diagonal cross-hatch pattern of dotted lines
vcDottedGridPattern 2037	Cross-hatch pattern of dotted lines
vcFDiagonalPattern 4	Diagonal lines slanting from top left to bottom right
vcHorizontalBrickPattern 2033	Horizontal brick pattern
vcHorizontalGradientPattern 52	Horizontal color gradient
vcHorizontalPattern 3	Horizontal lines
vcLargeCheckerboardPattern 2044	Checkerboard pattern showing squares of twice the size of vcSmallChecker-BoardPattern
vcLargeConfettiPattern 2029	Confetti pattern, large
vcLightDownwardDiagonalPattern 2012	Diagonal lines slanting to from top left to bottom right; spaced 50% closer than vcBDiagonalPattern

vcLightHorizontalPattern 2019	Horizontal lines spaced 50% closer than vcHorizontalPattern
vcLightUpwardDiagonalPattern 2013	Diagonal lines slanting from bottom left to top right, spaced 50% closer than vcBDiagonalPattern
vcLightVerticalPattern 2018	Vertical lines spaced 50% closer than vcVerticalPattern
vcNarrowHorizontalPattern 2021	Horizontal lines spaced 75 % closer than vcHorizontalPattern
vcNarrowVerticalPattern 2020	Vertical lines spaced 75% closer than vcVerticalPattern
vcNoPattern 1276	No fill pattern
vcOutlinedDiamondPattern 2045	Diagonal cross-hatch pattern, large
vcPlaidPattern 2035	Plaid pattern
vcShinglePattern 2039	Diagonal shingle pattern
vcSmallCheckerBoardPattern 2043	Checkerboard pattern
vcSmallConfettiPattern 2028	Confetti pattern
vcSmallGridPattern 2042	Cross-hatch pattern spaced 50% closer than vcCrossPattern
vcSolidDiamondPattern 2046	Checkerboard pattern showing diagonal squares
vcSpherePattern 2041	Checkerboard of spheres
vcTrellisPattern 2040	Trellis pattern
vcVerticalBottomLightedConvexPattern 43	Vertical color gradient from dark to bright
vcVerticalConcavePattern 40	Vertical color gradient from dark to bright to dark
vcVerticalConvexPattern 41	Vertical color gradient from bright to dark to bright
vcVerticalGradientPattern 62	Vertical color gradient
vcVerticalPattern 2	Vertical lines

vcVerticalTopLightedConvexPattern 42	Vertical color gradient from bright to dark
vcWavePattern 2031	Horizontal wave pattern
vcWeavePattern 2034	Interwoven stripe pattern
vcWideDownwardDiagonalPattern 2016	Diagonal lines slanting from top left to bottom right, showing the same spacing but three times the line width of vcF-DiagonalPattern
vcWideUpwardDiagonalPattern 2017	Diagonal lines slanting from bottom left to top right right, showing the same spacing but three times the line width of vcBDiagonalPattern
vcZigZagPattern 2030	Horizontal zig-zag lines

Position

Property of VcRibbon

This property lets you set or retrieve the position of the ribbon.

	Data Type	Explanation
Property value	RibbonPositionEnum	ribbon position
	Possible Values:	
	vcRPBottom 2	bottom
	vcRPNone 0	none
	vcRPTop 1	top

ReferenceDate

Property of VcRibbon

This property lets you set or retrieve the reference date.

	Data Type	Explanation
Parameter:		
⇨ Rückgabewert	Boolean	Ribbon uses (True) / does not use (False) reference date
	Possible Values:	
		Group invisible/visible

		group nodes are/are not visible
Property value	Date	Reference date

TextAlignment

Property of VcRibbon

This property lets you set or retrieve the alignment of the major ticks of the ribbon.

	Data Type	Explanation
Property value	HorRibbonTextAlignmentEnum **Possible Values:** vcRTAtTickAligned 1039 vcRTHorCenterAligned -1 vcRTLeftAligned -3 vcRTRightAligned -2	Type of text alignement Text placed at tick Text horizontally centered between two major ticks Text left aligned between two major ticks Text right aligned between two major ticks

Example Code

```
Dim timeScale As VcTimeScale
Dim ribbon As VcRibbon

Set timeScale = VcGantt1.TimeScaleCollection.Active
Set ribbon = timeScale.Ribbon (0, 0)
ribbon.TextAlignment = vcRTLeftAligned
```

TickColor

Property of VcRibbon

This property lets you set or retrieve the color of ticks.

	Data Type	Explanation
Property value	Color RGB ({0...255},{0...255},{0...255})	RGB color values **Default value:** 0,0,0

TickPosition

Property of VcRibbon

This property lets you set or retrieve the tick position.

	Data Type	Explanation
Property value	RibbonTickPositionEnum **Possible Values:** vcTPAbove 1044 vcTPBelow 1045	Tick position above below

Type

Property of VcRibbon

This property lets you set or retrieve the ribbon type. The types available are listed below.

	Data Type	Explanation
Property value	RibbonTypeEnum **Possible Values:** vcDayRibbon 5 vcFiscalQuarterRibbon 3002 vcFiscalYearRibbon 3001 vcHourRibbon 6 vcMinuteRibbon 7 vcMonthRibbon 3 vcQuarterRibbon 10 vcSecondRibbon 9 vcShiftRibbon 8 vcWeekRibbon 4 vcYearRibbon 1	Ribbon type Ribbon showing days' units Ribbon showing fiscal quarters' units Ribbon showing fiscal years' units Ribbon showing hours' units Ribbon showing minutes' units Ribbon showing months' units Ribbon showing quarters' units Ribbon showing seconds' units Ribbon showing shifts Ribbon showing weeks' units Ribbon showing years' units

Example Code

```
Dim timeScale As VcTimeScale
Dim ribbon As VcRibbon

Set timeScale =  VcGantt1.TimeScaleCollection.Active
Set ribbon = timeScale.Ribbon(0, 0)
ribbon.Type = vcWeekRibbon
```

UnitSeparation

Property of VcRibbon

This property lets you set or retrieve the appearance of the major ticks of the ribbon. A full line, a tick and no line are the features available.

	Data Type	Explanation
Property value	UnitSeparationEnum **Possible Values:** vcUSFullLine 4 vcUSNone 1	Appearance of the major tick Units separated by full lines Units not separated

vcUSTick 1035	Units separated by ticks

Example Code

```
Dim timeScale As VcTimeScale
Dim ribbon As VcRibbon

Set timeScale = VcGantt1.TimeScaleCollection.Active
Set ribbon = timeScale.Ribbon (0, 1)
ribbon.UnitSeparation = vcUSTick
```

UseReferenceDate

Property of VcRibbon

This property lets you set or retrieve whether the ribbon uses a reference date.

	Data Type	Explanation

7.72 VcScheduler

Scheduler

An object of the type **VcScheduler** represents a module for calculating simple project data, such as the early end of a project or its early start (if calculations are performed backward), or its free float and total float.

Properties

- ActualEndDateDataFieldIndex
- ActualStartDateDataFieldIndex
- AutomaticSchedulingEnabled
- DurationDataFieldIndex
- EarlyEndDateDataFieldIndex
- EarlyStartDateDataFieldIndex
- EndDateForAutomaticScheduling
- EndDateNotLaterThanDataFieldIndex
- FreeFloatDataFieldIndex
- LateEndDateDataFieldIndex
- LateStartDateDataFieldIndex
- LinkDurationDataFieldIndex
- ScheduledProjectEndDate
- ScheduledProjectStartDate
- ScheduleSuccessorsOnlyEnabled
- StartDateForAutomaticScheduling
- StartDateNotEarlierThanDataFieldIndex
- TotalFloatDataFieldIndex

Methods

- ScheduleProject

Properties

ActualEndDateDataFieldIndex

Property of VcScheduler

With this property you can set/retrieve the index of the data field which contains the present end date of the activity. This is only possible as long as no data has been loaded.

	Data Type	Explanation
Property value	Long	Index of the data field which holds the valid end date

ActualStartDateDataFieldIndex

Property of VcScheduler

This property lets you set/retrieve the index of the data field which contains the start date set to the activity. This is only possible as long as no data has been loaded.

	Data Type	Explanation
Property value	Long	Index of the data field which holds the valid start date

AutomaticSchedulingEnabled

Property of VcScheduler

This property lets you set or retrieve whether automatic time scheduling is switched on or off.

	Data Type	Explanation
Property value	Boolean	Automatic time scheduling is switched on (True) or off (False) **Default value:** False
	Possible Values:	Group invisible/visible group nodes are/are not visible

DurationDataFieldIndex

Property of VcScheduler

With this property you can set/retrieve the index of the data field which contains the duration of the activity. This is only possible as long as no data has been loaded.

	Data Type	Explanation
Property value	Long	Index of the data field which holds the duration of the activity

EarlyEndDateDataFieldIndex

Property of VcScheduler

With this property you can set/retrieve the index of the data field which contains the calculated earliest possible end date of the activity. This is only possible as long as no data has been loaded.

	Data Type	Explanation
Property value	Long	Index of the data field which holds the earliest possible end date of an acitivity

EarlyStartDateDataFieldIndex

Property of VcScheduler

With this property you can set/retrieve the index of the data field which contains the calculated earliest possible start date of the activity. This is only possible as long as no data has been loaded.

	Data Type	Explanation
Property value	Long	Index of the data field which holds the earliest possible start date of an activity

EndDateForAutomaticScheduling

Property of VcScheduler

In case **Automatic scheduling** is activated, this property lets you set or retrieve the end date of the project.

	Data Type	Explanation
Property value	Date	Desired end date for automatic scheduling

EndDateNotLaterThanDataFieldIndex

Property of VcScheduler

With this property you can set/retrieve the index of the data field which contains the desired latest end date of the activity. This is only possible as long as no data has been loaded.

	Data Type	Explanation
Property value	Long	Index of the data field which holds the desired latest end date

FreeFloatDataFieldIndex

Property of VcScheduler

With this property you can set/retrieve the index of the data field which contains the calculated free float of the activity. This is only possible as long as no data has been loaded.

	Data Type	Explanation
Property value	Long	Index of the data field which holds the free float

LateEndDateDataFieldIndex

Property of VcScheduler

With this property you can set/retrieve the index of the data field which contains the calculated latest possible end date of the activity. This is only possible as long as no data has been loaded.

	Data Type	Explanation
Property value	Long	Index of the data field which holds the latest possible end date

LateStartDateDataFieldIndex

Property of VcScheduler

With this property you can set/retrieve the index of the data field which contains the calculated latest possible start date of the activity. This is only possible as long as no data has been loaded.

	Data Type	Explanation
Property value	Long	Index of the data field which holds the latest possible start date

LinkDurationDataFieldIndex

Property of VcScheduler

This property lets you set or retrieve the index of a data field in the project in which a minimum temporal distance between predecessor and successor can be stored. This is only possible as long as no data has been loaded.

	Data Type	Explanation
Property value	Long	Index of the data field which holds the minimum time space between a predecessor and a successor node

ScheduledProjectEndDate

Read Only Property of VcScheduler

This property returns the **early end** of a project after having calculated the project dates by **VcScheduler.ScheduleProject** if the start date was set before.

This property can also be set on the **General** property page.

	Data Type	Explanation
Property value	Date	Index of the data field which holds the scheduled end date of the project

ScheduledProjectStartDate

Read Only Property of VcScheduler

This property returns the **late start** of a project after the project dates were calculated by **VcScheduler.ScheduleProject** if an end date was set before.

This property can also be set on the **General** property page.

	Data Type	Explanation
Property value	Date	Index of the data field which holds the scheduled start date of the project

ScheduleSuccessorsOnlyEnabled

Property of VcScheduler

With this property you can set/retrieve whether the scheduling of only those nodes that have a predecessor node is switched on or off; otherwise all nodes will be scheduled. A "project start" will thus be ignored.

	Data Type	Explanation
Property value	Boolean	Scheduling of nodes only with predecessors is switched on/off
	Possible Values:	Group invisible/visible group nodes are/are not visible

StartDateForAutomaticScheduling

Property of VcScheduler

In case **Automatic scheduling** is activated, this property lets you set or retrieve the start date of the project.

	Data Type	Explanation
Property value	Date	Desired start date for automatic scheduling

StartDateNotEarlierThanDataFieldIndex

Property of VcScheduler

This property lets you set or retrieve the index of the data field which contains the desired earliest start date of the activity. This is only possible as long as no data has been loaded.

	Data Type	Explanation
Property value	Long	Index of the data field which holds the desired early start date

TotalFloatDataFieldIndex

Property of VcScheduler

This property lets you set or retrieve the index of the data field which contains the calculated total float of the activity. This is only possible as long as no data has been loaded.

	Data Type	Explanation
Property value	Long	Index of the data field which holds the total float

Methods

ScheduleProject

Method of VcScheduler

This method lets you calculate the dates of a project (early / late start, early / late end, free float, total float) of a project. The desired start and end date can be set by this method. By passing only the end date, the project start will be calculated, by passing only the start date, the project end will be calculated. You can pass both dates, which will add the corresponding float to the activities. (This only works with matching dates, which means that the end date for example should not be within the project time period.) At least one date must be passed, otherwise an error message will occur. If a cycle amongst the nodes and links is identified, the ones affected will be marked.

The results will be stored to fields that you can set by the properties**Early-StartDateDataFieldIndex**, **LateStartDateDataFieldIndex**, **EarlyEndDate-**

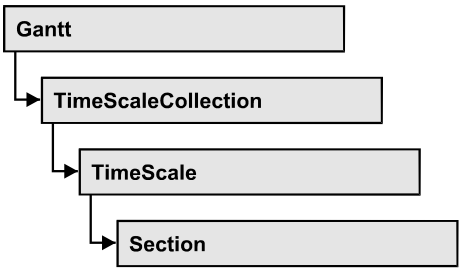
DataFieldIndex, **LateEndDateDataFieldIndex**, **FreeFloatDataFieldIndex** and **TotalFloatDataFieldIndex**.

	Data Type	Explanation
Parameter:		
⇨ startDate	Date	Desired start date
⇨ endDate	Date	Desired end date
Return value	Boolean	The project data were successfully calculated (true) / were not calculated (False)

Example Code

```
VcScheduler.ScheduleProject (3.5.2012,1.10.2012)
```

7.73 VcSection



An object of the type VcSection represents a section of the time scale.

Properties

- CalendarGridEx
- Collapse
- DateLineGrid
- LineColor
- LineColor
- Ribbon
- StartDate
- Unit
- UnitWidth
- UnitWidthEx

Properties

CalendarGridEx

Read Only Property of VcSection

This property lets you enquire a calendar grid of the section.

	Data Type	Explanation
Parameter: ⇨ gridIndex	Integer **Possible Values:**	Index of the calendar grid Data field index
Property value	VcCalendarGrid	CalendarGrid object

Collapse

Property of VcSection

This property lets you set or retrieve whether workfree periods of this section are to be collapsed. This property can also be set in the subdialog **Edit time scale section** of the **Specify Time Scale** dialog which you can reach by the **Time scales...** button on the property page **Objects**.

Tip: Please note that the visible time scale section will be shifted when you modify the property value at runtime. If you want to make sure that always the same reference date is displayed on the left, please call the following method:

```
Set_NonWorkIntervalsCollapsed(vcGantt1, true);

private static void Set_NonWorkIntervalsCollapsed(VcGantt gantt, bool collapse)

    {

        DateTime dt_left = new DateTime();

        DateTime dt_right = new DateTime();

        gantt.GetCurrentViewDates(ref dt_left, ref dt_right);


gantt.TimeScaleCollection.Active.get_Section(0).NonWorkIntervalsCollapsed = collapse;



        gantt.ScrollToDate(dt_left, VcHorizontalAlignment.vcLeftAligned, 0);

    }
```

	Data Type	***Explanation***
Property value	*Boolean*	*Workfree periods are/are not collapsed*
	Possible Values:	*Group invisible/visible* *group nodes are/are not visible*

Example Code

```
Dim timeScaleCltn As VcTimeScaleCollection
Dim timeScale As VcTimeScale
Dim section As VcSection

Set timeScaleCltn = VcGantt1.TimeScaleCollection
Set timeScale = timeScaleCltn.Active
Set section = timeScale.Section(1)
section.Collapse = True
```

DateLineGrid

Read Only Property of VcSection

This property gives you access to the DateLineGrid object, that lets you mark time periods such as days, weeks or months by vertical lines.

	Data Type	***Explanation***
Parameter:		
⇨ *gridIndex*	*Integer*	*Index of the date line grid*
	Possible Values:	*Data field index*
Property value	*VcDateLineGrid*	*DateLine object*

Example Code

```
Dim timeScale As VcTimeScale
Dim section As VcSection
Dim dateLineGrid As VcDateLineGrid

Set timeScale = VcGantt1.TimeScaleCollection.Active
Set section = timeScale.Section(0)
Set dateLineGrid = section.DateLineGrid (0)
```

LineColor

Property of VcSection

This property lets you set or retrieve the color of the (border) lines of ***all*** time scale sections and returns the color of the first time scale section. It is not possible to set a color for each section.l

	Data Type	**Explanation**
Parameter:		
⇨ Rückgabewert	Integer	RGB color values ({0...255},{0...255},{0...255})
	Possible Values:	Data field index

Property value	Color	RGB color values ({0...255},{0...255},{0...255})

LineColor

Property of VcSection

This property lets you set or retrieve the color of the (border) lines of **all** time scale sections and returns the color of the first time scale section. It is not possible to set a color for each section.l

	Data Type	**Explanation**
Parameter: ⇨ Rückgabewert	Integer **Possible Values:**	RGB color values ({0...255},{0...255},{0...255}) Data field index
Property value	Color	RGB color values ({0...255},{0...255},{0...255})

Ribbon

Property of VcSection

This property gives access to each ribbon of a section.

	Data Type	**Explanation**
Parameter: ⇨ ribbonIndex	Integer **Possible Values:**	Index of the ribbon Data field index
Property value	VcRibbon	Ribbon object

Example Code

```
Dim timeScale As VcTimeScale
Dim section As VcSection
Dim ribbon As VcRibbon

Set timeScale = VcGantt1.TimeScaleCollection.Active
Set section = timeScale.Section(0)
Set ribbon = section.ribbon(0)
```

StartDate

Property of VcSection

This property lets you set or retrieve the start date of a section. Note: The start date of the first section (Section 0) is specified by the project start and must not be edited here. Besides, you cannot specify a start date that is beyond the time scale.

	Data Type	Explanation
Property value	Date/Time	Start date of the time scale section

Example Code

```
Dim timeScale As VcTimeScale
Dim section As VcSection

Set timeScale = VcGantt1.TimeScaleCollection.Active
Set section = timeScale.Section(1)
section.StartDate = "21.06.14"
```

Unit

Property of VcSection

This property lets you set or retrieve the time unit that a section is based on.

	Data Type	Explanation
Property value	TimeUnitEnum **Possible Values:** vcDay 5 vcHour 6 vcMinute 7 vcSecond 8	Time unit of the section Time unit **day** Time unit **hour** Time unit **minute** Time unit **second**

Example Code

```
Dim timeScale As VcTimeScale
Dim section As VcSection

Set timeScale =  VcGantt1.TimeScaleCollection.Active
Set section = timeScale.Section(0)
section.Unit = vcHour
```

UnitWidth

Property of VcSection

This property lets you set or retrieve the unit width of a section (in 1/100 mm). This property also can be set in the **Specify Time Scale** dialog.

	Data Type	Explanation
Property value	Long	Unit width (by 1/100 mm; minimum width: > 0)

Example Code

```
Dim timeScale As VcTimeScale
Dim section As VcSection

Set timeScale =  VcGantt1.TimeScaleCollection.Active
Set section = timeScale.Section(0)
section.Unit = vcDay
section.UnitWidth = 660
```

UnitWidthEx

Property of VcSection

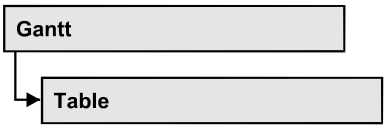
This property only differs from the property **UnidtWidth** by the data type **Double** that is more exact than the data type **Long**.

	Data Type	Explanation
Property value	Double	unit width (1/100 mm)

Example Code

```
Dim timeScale As VcTimeScale
Dim section As VcSection

Set timeScale =  VcGantt1.TimeScaleCollection.Active
Set section = timeScale.Section(0)
section.Unit = vcDay
section.UnitWidthEx = 660
```

7.74 VcTable



An object of the type VcTable object controls the graphical design of the table section of the diagram: the table heading, column widths and the available formats.

Properties

- ColumnTitle
- ColumnWidth
- Name
- NoOfColumns
- Position
- TableFormatCollection
- UpdateBehaviorName
- Visible

Methods

- IdentifyFormatField
- OptimizeColumnWidth

Properties

ColumnTitle

Property of VcTable

This property lets you specify the caption for each table column. This property also can be set in the **Edit Table** dialog.

	Data Type	Explanation
Parameter: ⇨ colNumber	Integer **Possible Values:**	Number of table column Data field index
Property value	String	Column title

	Possible Values:	Name of the color map

Example Code

```
Dim table As VcTable

Set table = VcGantt1.Table
table.ColumnTitle(1) = "ID"
```

ColumnWidth

Property of VcTable

This property lets you specify the width of each table column. This property can also be set in the **Edit Table** dialog.

	Data Type	Explanation
Parameter: ⇨ colNumber	Integer **Possible Values:**	Number of table column Data field index
Property value	Long	Column width in units of 1/100 mm

Example Code

```
Dim table As VcTable

Set table = VcGantt1.Table
table.ColumnWidth(1) = 1200
```

Name

Property of VcTable

This property lets you set or retrieve the name of the table. This property also can be set in the **Edit Table** dialog.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the table Name of the color map

NoOfColumns

Read Only Property of VcTable

This property lets you retrieve the number of columns of the table.

	Data Type	Explanation
Property value	Integer	Number of table columns
	Possible Values:	
		Data field index

Position

Read Only Property of VcTable

This property lets you enquire whether the table is displayed left or right of the diagram.

	Data Type	Explanation
Property value	TablePositionEnum	Position of the table
	Possible Values:	
	vcLeftTable 0	Table on the left of the diagram
	vcRightTable 1	Table on the right of the diagram

Example Code

```
Dim table As VcTable

Set table = VcGantt1.Table
MsgBox (table.Position)
```

TableFormatCollection

Read Only Property of VcTable

This property gives access to the table format collection that contains all table formats available to the table.

	Data Type	Explanation
Property value	VcTableFormatCollection	TableFormatCollection object

Example Code

```
Dim table As VcTable
Dim formatCltn As VcTableFormatCollection

Set table = VcGantt1.Table
Set formatCltn = table.TableFormatCollection
```

UpdateBehaviorName

Property of VcTable

This property lets you set or retrieve the name of the UpdateBehavior.

	Data Type	Explanation
Property value	String	Name of the UpdateBehavior
	Possible Values:	Name of the color map

Visible

Property of VcTable

This property lets you set or retrieve whether the table is visible or not.

	Data Type	Explanation
Property value	Boolean	Table visible/invisible
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
Private Sub chkBoxTable_Click()

    Dim table As VcTable

    Set table = VcGantt1.table

    If chkBoxTable.Value = vbChecked Then
        table.Visible = True
    Else
        table.Visible = False
    End If

End Sub
```

Methods

IdentifyFormatField

Method of VcTable

This method lets you retrieve the index of the format field at the specified position. If there is a field at the position specified, **True** will be returned, if there isn't, the method will deliver **False**.

	Data Type	Explanation
Parameter:		
⇨ x	Long	X coordinate of the position
⇨ y	Long	Y coordinate of the position
⇦ format	VcTableFormat	Identified format
⇦ formatFieldIndex	Integer	Index of the format field
	Possible Values:	Data field index
Return value	Boolean	A format field exists/does not exist at the position specified

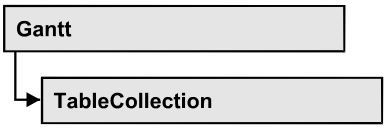
OptimizeColumnWidth

Method of VcTable

This method lets you calculate the optimized width of a column. It depends on the length of the longest text in the column. The setting ColumnNo = 0 optimizes all columns.

	Data Type	Explanation
Parameter:		
⇨ columnNo	Integer	Column number
	Possible Values:	Data field index
Return value	Void	

7.75 VcTableCollection



An object of the type VcTableCollection contains all available tables. You can access all objects in an iterative loop by **For Each table In Table-Collection** or by the methods **First...** and **Next...**. You can access a single table using the methods **TableByName** and **TableByIndex**. The number of tables in the collection object can be retrieved by the property **Count**. By the property **Active** you can set or retrieve the table that is presently active.

Properties

- _NewEnum
- Active
- Count

Methods

- FirstTable
- NextTable
- TableByIndex
- TableByName

Properties

_NewEnum

Read Only Property of VcTableCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all table objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method **GetEnumerator** is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim table As VcTable

For Each table In VcGantt1.TableCollection
   Debug.Print table.Name
Next
```

Active

Property of VcTableCollection

This property lets you set or retrieve the table currently displayed in the diagram.

	Data Type	Explanation
Property value	VcTable	Table currently used

Example Code

```
Dim tableCltn As VcTableCollection
Dim activeTable As VcTable

Set tableCltn = VcGantt1.TableCollection. _
                            TableByName("TABLE_1"). _
                            TableCollection
Set activeTable = tableCltn.Active
```

Count

Read Only Property of VcTableCollection

This property lets you retrieve the number of tables in the table collection.

	Data Type	Explanation
Property value	Long	Number of tables

Example Code

```
Dim tableCltn As VcTableCollection
Dim numberTables  As Integer

Set tableCltn = VcGantt1.TableCollection
numberTables = tableCltn.Count
```

Methods

FirstTable

Method of VcTableCollection

This method can be used to access the initial value, i.e. the first table of a table collection, and to continue in a forward iteration loop by the method **NextTable** for the tables following. If there is no table in the table collection, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcTable	First table

Example Code

```
Dim tableCltn As VcTableCollection
Dim table As VcTable

Set tableCltn = VcGantt1.TableCollection
Set table = tableCltn.FirstTable
```

NextTable

Method of VcTableCollection

This method can be used in a forward iteration loop to retrieve subsequent tables from a table collection after initializing the loop by the method **FirstTable**. If there is no table left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcTable	Subsequent table

Example Code

```
Dim tableCltn As VcTableCollection
Dim table As VcTable
Dim stringTable As String

Set tableCltn = VcGantt1.TableCollection
Set table = tableCltn.FirstTable

While Not table Is Nothing
     stringTable = table.AllData
     Listbox.AddItem (stringTable)
     Set table = tableCltn.NextTable
Wend
```

TableByIndex

Method of VcTableCollection

This method lets you access a table by its index. If a table of the specified index does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter:		
⇨ index	Integer	Index of the table
	Possible Values:	Data field index
Return value	VcTable	Table object returned

TableByName

Method of VcTableCollection

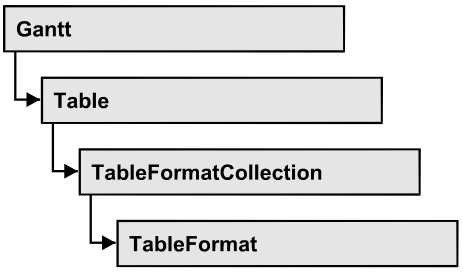
This method retrieves a table object by its name. If a table of the specified name does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter:		
⇨ tableName	String	Name of the data table
	Possible Values:	Name of the color map
Return value	VcTable	Data table returned

Example Code

```
Dim tableCltn As VcTableCollection
Dim table As VcTable

Set tableCltn = VcGantt1.TableCollection
Set table = tableCltn.TableByName("Standard")
```

7.76 VcTableFormat



An object of the type VcTableFormat defines the content and the appearance of a table row. A table row contains either the activity data or the group headings. In a table format, you can specify the data field contained in a table field. Each table field is specified by its column. Furthermore, you can specify a font (name, size, body, color), a background color, an horizontal alignment and margins individually for each field.

Available table formats:

- StandardList (for activities that are not summarized)
- ListFormat2 (alternative of StandardList, can be assigned by filters)
- ListFormat3 (alternative of StandardList, can be assigned by filters)
- Subtitle (for group headings when group is expanded)
- Subtitle_n (for multi-level grouping for group headings when group is expanded)
- Collapsed (for group headings when group is collapsed)
- Collapsed_n (for multi-level grouping for group headings when group is collapsed)
- Hierarchy (für summarized activities in a hierarchy)
- HierarchyCollapsed (for collapsed summarized activities in a hierarchy)

Properties

- _NewEnum
- CollapseColumn
- FieldsSeparatedByLines
- FilterName
- FormatField
- FormatFieldCount
- IndentColumn
- IndentWidth

- Name
- SeparationLineColor
- ThreeDEffect

Properties

_NewEnum

Read Only Property of VcTableFormat

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all table format field objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method **GetEnumerator** is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim formatField As VcTableFormatField

For Each formatField In format
    Debug.Print formatField.Index
Next
```

CollapseColumn

Property of VcTableFormat

This property lets you specify whether in a column which contains more than one line + or - for collapsing or showing the lines shall be displayed.

	Data Type	Explanation
Property value	Integer	Display of +/- in column switched on
	Possible Values:	Data field index

Example Code

```
' Display of +/- in the fifth column
VcGantt1.Table.TableFormatCollection.FormatByName("Hierarchy").CollapseColumn =
5
```

```
VcGantt1.Table.TableFormatCollection.FormatByName("HierarchyCollapsed").Collapse
Column = 5
```

FieldsSeparatedByLines

Property of VcTableFormat

This property lets you set or retrieve whether the table fields are to be separated by lines.

	Data Type	Explanation
Property value	Boolean	Table fields are separated by lines (True)/ are not separated by lines (False).
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
Dim format As VcTableFormat

Set format = VcGantt1.Table.TableFormatCollection.FormatByName("StandardList")
format.FieldsSeparatedByLines = True
```

FilterName

Property of VcTableFormat

This property lets you specify the name of the filter that defines what activities the table format is to apply to.

	Data Type	Explanation
Property value	String	Name of the filter
	Possible Values:	Name of the color map

Example Code

```
Dim format As VcTableFormat

Set format = VcGantt1.Table.TableFormatCollection.FormatByName("ListFormat2")
format.FilterName = "Code1NotA"
```

FormatField

Read Only Property of VcTableFormat

This property lets you retrieve a VcTableFormatField object by an index. The index has to be in the range from 0 to FormatFieldCount-1.

Note to users of versions previous to 3.0: The index does **not** count in the range from 1 to FormatFieldCount as in the versions up to 3.0.

	Data Type	Explanation
Parameter: ⇨ index	Integer **Possible Values:**	Index of the table format field 0 ... .FormatFieldCount-1 Data field index
Property value	VcTableFormatField	Table format field

FormatFieldCount

Read Only Property of VcTableFormat

This property lets you retrieve the number of table columns of this table format.

	Data Type	Explanation
Property value	Integer **Possible Values:**	Number of table columns Data field index

Example Code

```
Dim format As VcTableFormat
Dim numberOfColumns As Integer

Set format = VcGantt1.Table.TableFormatCollection.FormatByName("StandardList")
numberOfColumns = FormatFieldCount
```

IndentColumn

Property of VcTableFormat

This property lets you specify the number of the column which shall be indented.

	Data Type	Explanation
Property value	Integer **Possible Values:**	Number of indented column Data field index

Example Code

```
 ' Second column is indented
 VcGantt1.Table.TableFormatCollection.FormatByName("StandardList").IndentColumn
= 2
```

IndentWidth

Property of VcTableFormat

This property lets you set or retrieve the indentation (in mm) of the text lines in a table column

	Data Type	Explanation
Property value	Long	Measure of indentation

Example Code

```
 ' Second column is indented by 100 mm
 VcGantt1.Table.TableFormatCollection.FormatByName("StandardList").IndentColumn
= 2
 VcGantt1.Table.TableFormatCollection.FormatByName("StandardList").IndentWidth =
100
```

Name

Property of VcTableFormat

This property lets you set or retrieve the name of the table format.

	Data Type	Explanation
Property value	String **Possible Values:**	Table format name Name of the color map

Example Code

```
Dim format As VcTableFormat
Dim formatName As String

Set format = VcGantt1.Table.TableFormatCollection.FirstFormat
formatName = format.Name
```

SeparationLineColor

Property of VcTableFormat

This property lets you set or retrieve the color of the separation lines of the table fields. The default color is white.

	Data Type	Explanation
Property value	Color RGB	Color value ({0...255},{0...255},{0...255}) **Default value:** RGB(0,0,0)

Example Code

```
VcTableFormat.SeparationLineColor = RGB(255, 204, 204)
```

ThreeDEffect

Property of VcTableFormat

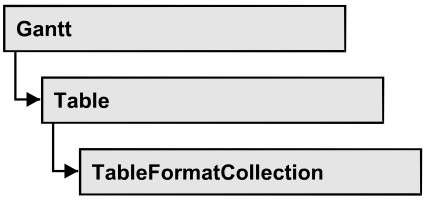
This property lets you set or retrieve whether this table format will be highlighted by a 3D effect.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	3D effect switched on (True)/switched off (False) Group invisible/visible group nodes are/are not visible

Example Code

```
Dim format As VcTableFormat

Set format = VcGantt1.Table.TableFormatCollection.FormatByName("StandardList")
format.ThreeDEffect = True
```

7.77 VcTableFormatCollection



An object of the type VcTableFormatCollection automatically contains all formats available to the table. You can access all objects in an iterative loop by **For Each format In FormatCollection** or by the methods **First...** and **Next...**. You can access a single format using the methods **FormatByName** and **FormatByIndex**. The number of tables in the collection object can be retrieved by the property **Count**.

Properties

- _NewEnum
- Count

Methods

- FirstFormat
- FormatByIndex
- FormatByName
- NextFormat

Properties

_NewEnum

Read Only Property of VcTableFormatCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all table format objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method **GetEnumerator** is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim format As VcTableFormat

For Each format In VcGantt1.Table.TableFormatCollection
   Debug.Print format.Name
Next
```

Count

Read Only Property of VcTableFormatCollection

This property lets you retrieve the number of table formats in the table format collection.

	Data Type	Explanation
Property value	Long	Number of table formats

Example Code

```
Dim formatCltn As VcTableFormatCollection
Dim numberOfFormats As Long

Set formatCltn = VcGantt1.TableFormatCollection
numberOfFormats = formatCltn.Count
```

Methods

FirstFormat

Method of VcTableFormatCollection

This method can be used to access the initial value, i.e. the first table format of a table format collection and then to continue in a forward iteration loop by the method **NextFormat** for the table formats following. If there is no table format in the table format collection, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcTableFormat	First table format

Example Code

```
Dim format As VcTableFormat
```

```
Set format = VcGantt1.Table.TableFormatCollection.FirstFormat
```

FormatByIndex

Method of VcTableFormatCollection

This method lets you access a table format by its index. If a table format of the specified index does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter: ⇨ index	Integer **Possible Values:**	Index of the table format Data field index
Return value	VcTableFormat	TableFormat object returned

FormatByName

Method of VcTableFormatCollection

By this method you can retrieve a table format by its name. If a table format of the specified name does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter: ⇨ formatName	String **Possible Values:**	Name of the table format Name of the color map
Return value	VcTableFormat	Table format

Example Code

```
Dim format As VcTableFormat

Set format = VcGantt1.Table.TableFormatCollection.FormatByName("StandardList")
```

NextFormat

Method of VcTableFormatCollection

This method can be used in a forward iteration loop to retrieve subsequent table formats from a table format collection after initializing the loop by the

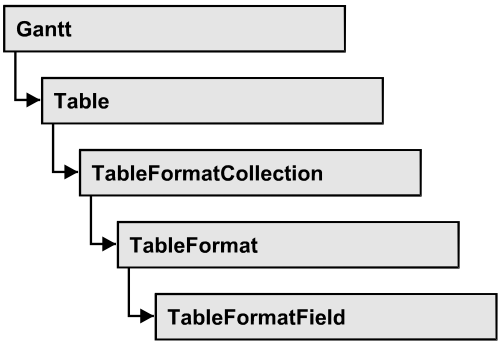
method **FirstFormat**. If there is no format left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcTableFormat	Subsequent table format

Example Code

```
Dim formatCltn As VcTableFormatCollection
Dim format As VcTableFormat

Set formatCltn = VcGantt1.Table.TableFormatCollection
Set format = formatCltn.FirstFormat

While Not format Is Nothing
     ListBox.AddItem format.Name
     Set format = formatCltn.NextFormat
Wend
```

7.78 VcTableFormatField



An object of the type **VcTableFormatField** represents a field of a **VcTableFormat** object. A table format field does not have a name as have many other objects, but is identified by ist index that defines ist position in the table format. A table can have 100 format fields at maximum.

Properties

- Alignment
- BottomMargin
- CombiField
- ConstantText
- FormatName
- GraphicsFileName
- GraphicsFileNameDataFieldIndex
- GraphicsFileNameMapName
- GraphicsHeight
- Index
- LeftMargin
- MaximumTextLineCount
- MinimumTextLineCount
- MultiState
- PatternBackgroundColorAsARGB
- PatternBackgroundColorDataFieldIndex
- PatternBackgroundColorMapName
- PatternColorAsARGB
- PatternColorDataFieldIndex
- PatternColorMapName
- PatternEx
- PatternExDataFieldIndex
- PatternExMapName

- RightMargin
- TextDataFieldIndex
- TextFont
- TextFontColor
- TextFontColorDataFieldIndex
- TextFontColorMapName
- TextFontDataFieldIndex
- TextFontMapName
- TopMargin
- Type

Properties

Alignment

Property of VcTableFormatField

This property lets you set or retrieve the alignment of the content of the table format field.

	Data Type	Explanation
Property value	FormatFieldAlignmentEnum	Alignment of the field content
	Possible Values:	
	vcFFABottom 28	bottom
	vcFFABottomLeft 27	bottom left
	vcFFABottomRight 29	bottom right
	vcFFACenter 25	center
	vcFFALeft 24	left
	vcFFARight 26	right
	vcFFATop 22	top
	vcFFATopLeft 21	top left
	vcFFATopRight 23	top right

BottomMargin

Property of VcTableFormatField

This property lets you set or retrieve the width (in mm) of the bottom margin of the table format field. It can also be set in the **Edit Table Format** dialog box.

	Data Type	Explanation
Property value	Integer	Width of the bottom margin of the table format field
	Possible Values:	0...9 Data field index

CombiField

Property of VcTableFormatField

This property lets you set or retrieve whether the table field is a combi field. (See also **Edit Table Format** dialog.)

	Data Type	Explanation
Property value	Boolean	Combi field (True)/ no combi field (False)
	Possible Values:	Group invisible/visible group nodes are/are not visible

ConstantText

Property of VcTableFormatField

This property allows the table format field to display a constant text, if the table format field is of the type ***vcFFTText*** and if the property **TextDat-FieldIndex** was set to **-1**.

	Data Type	Explanation
Property value	String	Constant text
	Possible Values:	Name of the color map

FormatName

Read Only Property of VcTableFormatField

This property lets you retrieve the name of the table format to which this table format field belongs.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of a table format object Name of the color map

GraphicsFileName

Property of VcTableFormatField

only for the type ***vcFFTGraphics***: This property lets you set or retrieve the name of a graphics file the content of which is displayed in the table format field. The graphics file name has to be valid. *Available formats:*

- *.BMP (Microsoft Windows Bitmap)
- *.EMF (Enhanced Metafile oder Enhanced Metafile Plus)
- *.GIF (Graphics Interchange Format)
- *.JPG (Joint Photographic Experts Group)
- *.PNG (Portable Network Graphics)
- *.TIF (Tagged Image File Format)
- *.VMF (Viewer Metafile)
- *.WMF (Microsoft Windows Metafile)
- *.WMF mit eingebautem EMF

EMF, EMF+, VMF and WMF are vector formats that allow to store a file independent of pixel resolution. All other formats are pixel-oriented and confined to a limited resolution.

The VMF format basically has been deprecated, but it will still be supported for some time to maintain compatibility with existing applications.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the graphics file

		Name of the color map

GraphicsFileNameDataFieldIndex

Property of VcTableFormatField

only for the type ***vcFFTGraphics***: This property lets you set or retrieve the data field index that is specified in the property **GraphicsFileNameMapName**. If the property has the value **-1**, in the table format field the graphics that is specified for the corresponding table format will be displayed. If a valid data field index is specified, but no map is specified, the graphics file name will be read from the specified data field.

	Data Type	Explanation
Property value	Integer	Index of the data field
	Possible Values:	Data field index

GraphicsFileNameMapName

Property of VcTableFormatField

only for the type ***vcFFTGraphics***: This property lets you set or retrieve the name of a map of the type **vcGraphicsFileMap** or "". If a name and additionally a data field index is specified in the property **GraphicsFile-NameDataFieldIndex**, a graphics of the map will be displayed. If no data field entry applies, the graphics specified in the property **GraphicsFileName** will be displayed.

	Data Type	Explanation
Property value	String	Name of the graphics map
	Possible Values:	Name of the color map

GraphicsHeight

Property of VcTableFormatField

This property lets you set or retrieve the height of a graphics of the type **vcFFTGraphics** in the table format field.

	Data Type	Explanation
Property value	Integer	Height of the graphics in mm
	Possible Values:	0 ... 99 Data field index

Index

Read Only Property of VcTableFormatField

This property lets you enquire the index of the table format field in the corresponding table format.

	Data Type	Explanation
Property value	Integer	Index of the table format field
	Possible Values:	Data field index

LeftMargin

Property of VcTableFormatField

This property lets you set or retrieve the width (in mm) of the left margin of the table format field. It can also be set in the **Edit Table Format** dialog box.

	Data Type	Explanation
Property value	Integer	Width of the left margin of the table format field
	Possible Values:	0…9 Data field index

MaximumTextLineCount

Property of VcTableFormatField

This property lets you set or retrieve the maximum number of lines in the table format field, if the table format field is of the type **vcFFTText**. Also see the property **MinimumTextLineCount**.

	Data Type	Explanation
Property value	Integer	Maximum number of lines
	Possible Values:	Data field index

MinimumTextLineCount

Property of VcTableFormatField

This property lets you set or retrieve the minimum number of lines in the table format field, if it is of the type **vcFFTText**. If there is more text than can be taken by the lines, the format field will be enlarged dynamically up to the maximum number of lines. When assigning a value by this property, please also remember to set the **MaximumTextLineCount** value anew, since otherwise the minimum value might overwrite the maximum value.

	Data Type	Explanation
Property value	Integer	Minimum number of lines
	Possible Values:	Data field index

MultiState

Property of VcTableFormatField

This property lets you set or retrieve, whether the table format field is a multi-state field. Multi-state fields are used for example to trigger a rotating sequence of different states and of the associated data fields when clicked.

	Data Type	Explanation
Property value	Boolean	Multi-state field (True) / no multi-state field (False)
	Possible Values:	Group invisible/visible group nodes are/are not visible

PatternBackgroundColorAsARGB

Property of VcTableFormatField

This property lets you set or retrieve the background color of the table format field. Color values have a transparency or alpha value, followed by a value for a red, a blue and a green partition (ARGB). The values range between 0..255. An alpha value of 0 equals complete transparency, whereas 255 represents a completely solid color. When casting an RGB value on an ARGB value, an alpha value of 255 has to be added.

If the table format field shall have the color of the table format, select the value **-1**.

If by the property **BackColorMapName** a map is specified, the map will set the background color of the table format field in dependence on data.

	Data Type	Explanation
Property value	Long	Background color of the table format **Default value:** -1

PatternBackgroundColorDataFieldIndex

Property of VcTableFormatField

This property lets you set or retrieve the data field index to be used with a color map specified by the property **PatternBackgroundColorMapName**. If you set this property to **-1**, no map will be used.

	Data Type	Explanation
Property value	Long	Data field index

PatternBackgroundColorMapName

Property of VcTableFormatField

This property lets you set or retrieve the name of a color map (type vcColorMap). If set to "", no map will be used. If a map name and additionally a data field index is specified in the property

PatternBackgroundColorDataFieldIndex, then the background color is controlled by the map. If no data field entry applies, the background color that is specified in the property **BackColor** will be used.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the color map Name of the color map

PatternColorAsARGB

Property of VcTableFormatField

This property lets you set or retrieve the pattern color of the table format field. Color values have a transparency or alpha value, followed by a value for a red, a blue and a green partition (ARGB). The values range between 0..255. An alpha value of 0 equals complete transparency, whereas 255 represents a completely solid color. When casting an RGB value on an ARGB value, an alpha value of 255 has to be added.

If the table format field shall have the background color of the table format, select the value **-1**.

	Data Type	Explanation
Property value	Long	Pattern color of the table format field

Example Code

```
Dim tableFormatCltn As VcTableFormatCollection
Dim tableFormatField As VcTableFormatField

Set tableFormatCltn = VcGantt1.TableFormatCollection
Set tableFormatField = tableFormatCltn.FirstFormat.formatField(0)
tableFormatField.PatternColor = RGB(0, 255, 0)
```

PatternColorDataFieldIndex

Read Only Property of VcTableFormatField

This property lets you set or retrieve the data field index that has to be specified if the property **PatternColorMapName** is used. If you set this property to **-1**, no map will be used.

	Data Type	Explanation
Property value	Integer **Possible Values:**	Data field index Data field index

PatternColorMapName

Read Only Property of VcTableFormatField

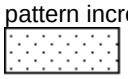
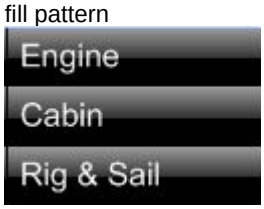
This property lets you set or retrieve the name of a color map (type vcColorMap). If set to "", no map will be used. Only if a map name and a data field index are specified in the property **PatternColorDataFieldIndex**, the pattern color is controlled by the map. If no data field entry applies, the pattern color of the calendar grid that is specified in the property **PatternColor** will be used.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the color map Name of the color map

PatternEx

Read Only Property of VcTableFormatField

This property lets you set or retrieve the pattern of the field background of the table format field.

	Data Type	Explanation
Property value	FillPatternEnum **Possible Values:** vc05PercentPattern... vc90PercentPattern 01 - 11 vcAeroGlassPattern 40	Pattern type **Default value:** As defined in the dialog Dots in foreground color on background color, the density of the foreground pattern increasing with the percentage  Vertical color gradient in the color of the fill pattern 

vcBDiagonalPattern 5	Diagonal lines slanting from bottom left to top right
vcCrossPattern 6	Cross-hatch pattern
vcDarkDownwardDiagonalPattern 2014	Diagonal lines slanting from top left to bottom right; spaced 50% closer than vcFDiagonalPattern and of twice the line width
vcDarkHorizontalPattern 2023	Horizontal lines spaced 50% closer than vcHorizontalPattern and of twice the line width
vcDarkUpwardDiagonalPattern 2015	Diagonal lines slanting from bottom left to top right, spaced 50% closer than vcBDiagonalPattern and of twice the line width
vcDarkVerticalPattern 2022	Vertical lines spaced 50% closer than vcVerticalPattern and of of twice the line width
vcDashedDownwardDiagonalPattern 2024	Dashed diagonal lines from top left to bottom right
vcDashedHorizontalPattern 2026	Dashed horizontal lines
vcDashedUpwardDiagonalPattern 2025	Dashed diagonal lines from bottom left to top right
vcDashedVerticalPattern 2027	Dashed vertical lines
vcDiagCrossPattern 7	Diagonal cross-hatch pattern, small
vcDiagonalBrickPattern 2032	Diagonal brick pattern
vcDivotPattern 2036	Divot pattern
vcDottedDiamondPattern 2038	Diagonal cross-hatch pattern of dotted lines
vcDottedGridPattern 2037	Cross-hatch pattern of dotted lines
vcFDiagonalPattern 4	Diagonal lines slanting from top left to bottom right
vcHorizontalBrickPattern 2033	Horizontal brick pattern
vcHorizontalGradientPattern 52	Horizontal color gradient

vcHorizontalPattern 3	Horizontal lines
vcLargeCheckerboardPattern 2044	Checkerboard pattern showing squares of twice the size of vcSmallChecker-BoardPattern
vcLargeConfettiPattern 2029	Confetti pattern, large
vcLightDownwardDiagonalPattern 2012	Diagonal lines slanting to from top left to bottom right; spaced 50% closer than vcBDiagonalPattern
vcLightHorizontalPattern 2019	Horizontal lines spaced 50% closer than vcHorizontalPattern
vcLightUpwardDiagonalPattern 2013	Diagonal lines slanting from bottom left to top right, spaced 50% closer than vcBDiagonalPattern
vcLightVerticalPattern 2018	Vertical lines spaced 50% closer than vcVerticalPattern
vcNarrowHorizontalPattern 2021	Horizontal lines spaced 75 % closer than vcHorizontalPattern
vcNarrowVerticalPattern 2020	Vertical lines spaced 75% closer than vcVerticalPattern
vcNoPattern 1276	No fill pattern
vcOutlinedDiamondPattern 2045	Diagonal cross-hatch pattern, large
vcPlaidPattern 2035	Plaid pattern
vcShinglePattern 2039	Diagonal shingle pattern
vcSmallCheckerBoardPattern 2043	Checkerboard pattern
vcSmallConfettiPattern 2028	Confetti pattern
vcSmallGridPattern 2042	Cross-hatch pattern spaced 50% closer than vcCrossPattern
vcSolidDiamondPattern 2046	Checkerboard pattern showing diagonal squares
vcSpherePattern 2041	Checkerboard of spheres
vcTrellisPattern 2040	Trellis pattern
vcVerticalBottomLightedConvexPattern 43	Vertical color gradient from dark to bright

vcVerticalConcavePattern 40	Vertical color gradient from dark to bright to dark
vcVerticalConvexPattern 41	Vertical color gradient from bright to dark to bright
vcVerticalGradientPattern 62	Vertical color gradient
vcVerticalPattern 2	Vertical lines
vcVerticalTopLightedConvexPattern 42	Vertical color gradient from bright to dark
vcWavePattern 2031	Horizontal wave pattern
vcWeavePattern 2034	Interwoven stripe pattern
vcWideDownwardDiagonalPattern 2016	Diagonal lines slanting from top left to bottom right, showing the same spacing but three times the line width of vcF-DiagonalPattern
vcWideUpwardDiagonalPattern 2017	Diagonal lines slanting from bottom left to top right right, showing the same spacing but three times the line width of vcBDiagonalPattern
vcZigZagPattern 2030	Horizontal zig-zag lines

PatternExDataFieldIndex

Property of VcTableFormatField

This property lets you set or retrieve the data field index to be used together with the property **PatternExMapName**. If you set this property to **-1**, no map will be used.

	Data Type	Explanation
Property value	Long	Data field index

PatternExMapName

Property of VcTableFormatField

This property lets you set or retrieve the name of a font map (type vcPatternMap). If set to "", no map will be used. If a map name and additionally a data field index is specified in the property **PatternExDataFieldIndex**, then the pattern is controlled by the map. If no data field entry applies, the pattern that is specified in the property **PatternEx** will be used.

	Data Type	Explanation
Property value	String	Name of the pattern map
	Possible Values:	Name of the color map

RightMargin

Property of VcTableFormatField

This property lets you set or retrieve the width (in mm) of the right margin of the table format field. It can also be set in the **Edit Table Format** dialog box.

	Data Type	Explanation
Property value	Integer	Width of the right margin of the table format field
		0...9
	Possible Values:	Data field index

TextDataFieldIndex

Property of VcTableFormatField

only for the type ***vcFFTText***: This property lets you set or retrieve the index of the data field the content of which is to be displayed in the table format field. If its value equals **-1**, the content of the property **ConstantText** will be returned.

	Data Type	Explanation
Property value	Integer	Index of the data field
	Possible Values:	

		Data field index

TextFont

Property of VcTableFormatField

This property lets you set or retrieve the font color of the table format field, if it is of the type **vcFFTText**. If a map was set by the property **TextFontMap-Name**, the map will control the text font in dependence of the data.

	Data Type	Explanation
Property value	StdFont	Font type of the table format

TextFontColor

Property of VcTableFormatField

This property lets you set or retrieve the font color of the table format field, if it is of the type **vcFFTText**. If a map was set by the property **TextFontMap-Name**, the map will control the text font color in dependence of the data.

	Data Type	Explanation
Property value	OLE_COLOR	Font color of the table format **Default value:** -1

TextFontColorDataFieldIndex

Property of VcTableFormatField

This property lets you set or retrieve the data field index to be used with a font color map specified by the property **TextFontColorMapName**. If you set this property to **-1**, no map will be used.

	Data Type	Explanation
Property value	Integer **Possible Values:**	Data field index Data field index

TextFontColorMapName

Property of VcTableFormatField

This property lets you set or retrieve the name of a color map (type vcColorMap) for the font color, if the format field is of the type **vcFFTText**. If the name of the color map is set to "", no map will be used. If a map name and a data field index are specified by the property **TextFontColorDataFieldIndex**, the font color will be controlled by the map. If no map entry applies, the font color specified in the property **TextFontColor** will be used.

	Data Type	Explanation
Property value	String	Name of the font color map
	Possible Values:	Name of the color map

TextFontDataFieldIndex

Property of VcTableFormatField

This property lets you set or retrieve the data field index required by the property **TextFontMapName** for a font map. If you set this property to **-1**, no map will be used.

	Data Type	Explanation
Property value	Integer	Data field index
	Possible Values:	Data field index

TextFontMapName

Property of VcTableFormatField

This property lets you set or retrieve the name of a font map (type vcFontMap). If set to "", no map will be used. If a map name and additionally a data field index is specified in the property **TextFontDataFieldIndex**, then the font is controlled by the map. If no data field entry applies, the font that is specified in the property **TextFont** will be used.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the font map Name of the color map

TopMargin

Property of VcTableFormatField

This property lets you set or retrieve the width (in mm) of the top margin of the table format field. It can also be set in the **Edit Table Format** dialog box.

	Data Type	Explanation
Property value	Integer **Possible Values:**	Width of the top margin of the table format field 0...9 Data field index

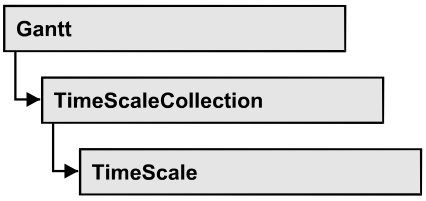
Type

Property of VcTableFormatField

This property lets you enquire the type of the table format field.

	Data Type	Explanation
Property value	FormatFieldTypeEnum **Possible Values:** vcFFTGraphics 64 vcFFTText 36	Type of the table format field graphics text

7.79 VcTimeScale



The VcTimeScale object represents the time scale at the top of the node area in the diagram. From several time scales that display different units, such as hours or weeks, you can select the time scale that meets your demands. The color and several font attributes can be set as you like. In the settings of the time scale the (vertical) grid lines and possibly the emphasizing of weekends also can be activated.

Properties

- BackgroundColor
- Font
- FontColor
- Name
- Ribbon
- Section
- ShowCalendarGrids
- ShowDateGrids
- ThreeDEffect
- UpdateBehaviorName

Properties

BackgroundColor

Property of VcTimeScale

This property lets you set or retrieve the background color of the time scale.

	Data Type	Explanation
Property value	Color	RGB color values

Example Code

```
Dim timeScale As VcTimeScale
```

```
Set timeScale = VcGantt1.TimeScaleCollection.Active
timeScale.BackgroundColor = RGB(200, 100, 150)
```

Font

Property of VcTimeScale

This property lets you set or retrieve all font attributes of the time scale.

	Data Type	Explanation
Property value	StdFont	Font attributes of the timescale

Example Code

```
Dim font As StdFont
font = VcGantt1.TimeScaleCollection.Active.Font
```

FontColor

Property of VcTimeScale

This property lets you set or retrieve the font color of the time scale.

	Data Type	Explanation
Property value	Color	RGB color values

Example Code

```
Dim timeScale As VcTimeScale

Set timeScale = VcGantt1.TimeScaleCollection.Active
timeScale.FontColor = RGB(10, 220, 220)
```

Name

Property of VcTimeScale

This property lets you set or retrieve the name of the time scale.

	Data Type	Explanation
Property value	String **Possible Values:**	Name Name of the color map

Example Code

```
Dim timeScale As VcTimeScale

Set timeScale = VcGantt1.TimeScaleCollection.Active
MsgBox "Active Timescale: " & timeScale.Name
```

Ribbon

Read Only Property of VcTimeScale

This property gives access to the ribbons of a time scale.

	Data Type	Explanation
Parameter:		
⇨ sectionIndex	Integer **Possible Values:**	Index of the time scale section Data field index
⇨ ribbonIndex	Integer **Possible Values:**	Index of the ribbon Data field index
Property value	VcRibbon	Ribbon object

Example Code

```
Dim timeScale As VcTimeScale
Dim ribbon As VcRibbon

Set timeScale = VcGantt1.TimeScaleCollection.Active
Set ribbon = timeScale.Ribbon (0, 0)
```

Section

Read Only Property of VcTimeScale

This property gives access to the sections of a time scale.

	Data Type	Explanation
Parameter:		
⇨ sectionIndex	Integer **Possible Values:**	Index of the section Data field index
Property value	VcSection	Section object

Example Code

```
Dim timeScale As VcTimeScale
Dim section As VcSection

Set timeScale = VcGantt1.TimeScaleCollection.Active
Set section = timeScale.Section(0)
```

ShowCalendarGrids

Property of VcTimeScale

This property lets you set or retrieve whether workfree periods are marked by gray shadings. This property also can be set in the **Specify Time Scale** dialog.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	Workfree periods are/are not displayed in gray. Group invisible/visible group nodes are/are not visible

Example Code

```
Private Sub chkBoxCalGrids_Click()

    Dim timescale As VcTimeScale

    Set timescale = VcGantt1.TimeScaleCollection.Active

    If chkBoxCalGrids = vbChecked Then
        timescale.ShowCalendarGrids = True
    Else
        timescale.ShowCalendarGrids = False
    End If

End Sub
```

ShowDateGrids

Property of VcTimeScale

This property lets you set or retrieve whether a vertical date grid is displayed. This property also can be set in the **Specify Time Scale** dialog.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	Date grids are/are not displayed. Group invisible/visible group nodes are/are not visible

Example Code

```
Private Sub chkBoxDateGrid_Click()

    Dim timescale As VcTimeScale

    Set timescale = VcGantt1.TimeScaleCollection.Active

    If chkBoxCalGrids = vbChecked Then
        timescale.ShowDateGrids = True
    Else
```

```
        timescale.ShowDateGrids = False
    End If

End Sub
```

ThreeDEffect

Property of VcTimeScale

This property lets you set or retrieve whether the time scale should have or has a three-dimensional appearance. This property also can be set in the **Specify Time Scale** dialog.

	Data Type	Explanation
Property value	Boolean	3D effect switched on (True)/switched off (False)
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
Dim timescale As VcTimeScale

Set timescale = VcGantt1.TimeScaleCollection.Active
timescale.ThreeDEffect = False
```

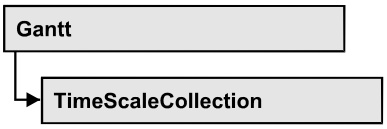
UpdateBehaviorName

Property of VcTimeScale

This property lets you set or retrieve the name of the UpdateBehavior.

	Data Type	Explanation
Property value	String	Name of the UpdateBehavior
	Possible Values:	Name of the color map

7.80 VcTimeScaleCollection



The VcTimeScaleCollection object contains all available time scales. You can access all objects in an iterative loop by **For Each timeScale In TimeScaleCollection** or by the methods **First...** and **Next...**. You can access a single time scale using the methods **TimeScaleByName** and **TimeScaleByIndex**. The number of time scales in the collection object can be retrieved by the property **Count**. By the property **Active** you can set or retrieve the time scale that is presently active.

Properties

- _NewEnum
- Active
- Count

Methods

- FirstTimeScale
- NextTimeScale
- TimeScaleByIndex
- TimeScaleByName

Properties

_NewEnum

Read Only Property of VcTimeScaleCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all time scale objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method **GetEnumerator** is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim timescale As VcTimeScale

For Each timescale In VcGantt1.TimeScaleCollection
   Debug.Print timescale.Name
Next
```

Active

Property of VcTimeScaleCollection

This method lets you set or retrieve the time scale currently displayed in the diagram.

	Data Type	Explanation
Property value	VcTimeScale	Timescale currently used

Example Code

```
Dim timeScaleCltn As VcTimeScaleCollection
Dim timeScale As VcTimeScale

Set timeScaleCltn = VcGantt1.TimeScaleCollection
Set timeScale = timeScaleCltn.Active
```

Count

Read Only Property of VcTimeScaleCollection

This property lets you retrieve the number of time scales in the TimeScaleCollection object.

	Data Type	Explanation
Property value	Long	Number of time scales

Example Code

```
Dim numberOfTimeScales As Long

numberOfTimeScales = VcGantt1.TimeScaleCollection.Count
```

Methods

FirstTimeScale

Method of VcTimeScaleCollection

This method can be used to access the initial value, i.e. the first time scale of a time scale collection, and then to continue in a forward iteration loop by the method **NextTimeScale** for the scales following. If there is no scale in the time scale collection, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcTimeScale	First time scale

Example Code

```
Dim timeScaleCltn As VcTimeScaleCollection
Dim timeScale As VcTimeScale

Set timeScaleCltn = VcGantt1.TimeScaleCollection
Set timeScale = timeScaleCltn.FirstTimeScale
```

NextTimeScale

Method of VcTimeScaleCollection

This method can be used in a forward iteration loop to retrieve subsequent time scales from a time scale collection after initializing the loop by the method **FirstTimeScale**. If there is no time scale left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcTimeScale	Subsequent time scale

Example Code

```
Dim timeScaleCltn As VcTimeScaleCollection
Dim timeScale As VcTimeScale

Set timeScaleCltn = VcGantt1.TimeScaleCollection
Set timeScale = timeScaleCltn.FirstTimeScale

While Not timeScale Is Nothing
    List1.AddItem timeScale.Name
    Set timeScale = timeScaleCltn.NextTimeScale
Wend
```

TimeScaleByIndex

Method of VcTimeScaleCollection

This method lets you access a time scale by its index. If a time scale of the specified index does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter: ⇨ index	Integer **Possible Values:**	Index of the time scale Data field index
Return value	VcTimeScale	Time scale object returned

TimeScaleByName

Method of VcTimeScaleCollection

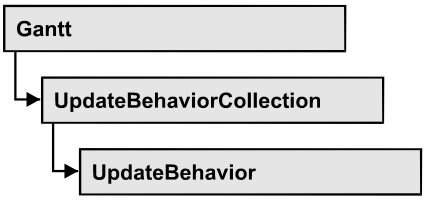
By this method you can retrieve a time scale by its name. If a time scale of the specified name does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter: ⇨ timeScaleName	String **Possible Values:**	Name of the time scale Name of the color map
Return value	VcTimeScale	Time scale

Example Code

```
Dim timeScaleCltn As VcTimeScaleCollection

Set timeScaleCltn = VcGantt1.TimeScaleCollection

timeScaleCltn.Active = timeScaleCltn.TimeScaleByName("Days")
```

7.81 VcUpdateBehavior



An object of the type **VcUpdateBehavior** contains a set of properties and methods that control the live update behavior of those objects on the screen to which it was assigned.

Properties

- IsEditable
- Name
- Specification

Methods

- Context
- PutInOrderAfter

Properties

IsEditable

Property of VcUpdateBehavior

This property lets you set or retrieve whether the update behavior should be editable at run time.

	Data Type	Explanation
Property value	Boolean	Update behavior editable (True) / not editable (False) **Default value:** True
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
Dim updBeh As VcUpdateBehavior

Set updBeh = VcUpdateBehavior.UpdateBehaviorByName("Start")
```

```
updBeh.IsEditable = False
```

Name

Property of VcUpdateBehavior

This property lets you set or retrieve the name of an update behavior

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the update behavior Name of the color map

Example Code

```
Dim updBehCltn As VcUpdateBehaviorCollection
Dim updBeh As VcUpdateBehavior

Set updBehCltn = VcGantt1.UpdateBehaviorCollection

For Each updBeh In updBehCltn
    ComboBox1.AddItem updBeh.Name
Next updBeh
```

Specification

Property of VcUpdateBehavior

This property lets you retrieve the specification of an update beavior. A specification is a string that contains legible ASCII characters from 32 to 127 only, so it can be stored without problems to text files or data bases. This allows for persistency. A specification can be used to create an update behavior by the method **VcUpdateBehaviorCollection.AddBySpecification**.

	Data Type	Explanation
Property value	String **Possible Values:**	Specification of the update behavior Name of the color map

Example Code

```
Dim updateBehaviorCltn As VcUpdateBehaviorCollection
Dim updateBehavior As VcUpdateBehavior

updateBehaviorCltn = VcGantt1.UpdateBehaviorCollection
updateBehavior = updateBehaviorCltn.FirstUpdateBehavior
MsgBox(updateBehavior.Specification)
```

Methods

Context

Method of VcUpdateBehavior

This method lets you retrieve the context settings of the update behavior.

	Data Type	Explanation

PutInOrderAfter

Method of VcUpdateBehavior

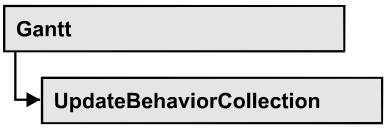
This method lets you set the update behavior behind an update behavior specified by name, within the UpdateBehaviorCollection. If you set the name to "", the update behavior will be put in the first position. The order of the update behaviors within the collection determines the order by which they apply to the objects that they were assigned to.

	Data Type	Explanation
Parameter:		
refUpdateBehaviorName	String **Possible Values:**	Name of the update behavior behind which the current update behavior is to be put. Name of the color map
Return value	Void	

Example Code

```
Dim updBehCltn As VcUpdateBehaviorCollection
Dim updBeh1 As VcUpdateBehavior
Dim updBeh2 As VcUpdateBehavior

updBehCltn = VcGantt1.UpdateBehaviorCollection()
updBeh1 = updBehCltn.Add("updBeh1")
updBeh2 = updBehCltn.Add("updBeh2")
updBeh1.PutInOrderAfter("updBeh2")
updBehCltn.Update()
```

7.82 VcUpdateBehaviorCollection



The VcUpdateBehaviorCollection object contains all update behaviors available. You can access all objects in an iterative loop by **For Each updateBehavior In UpdateBehaviorCollection** or by the methods **First…** and **Next…**. You can access a single update behavior by the methods **UpdateBehaviorByName** and **UpdateBehaviorByIndex**. The number of update behaviors in the collection object can be retrieved by the property **Count**. The methods **Add**, **Copy** and **Remove** allow to handle the update behaviors in the corresponding way.

Properties

- _NewEnum
- Active
- Count

Methods

- Add
- AddBySpecification
- Copy
- FirstUpdateBehavior
- NextUpdateBehavior
- Remove
- UpdateBehaviorByIndex
- UpdateBehaviorByName

Properties

_NewEnum

Read Only Property of VcUpdateBehaviorCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all update behavior objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method

GetEnumerator is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim updBeh As VcUpdateBehavior

For Each updBeh In VcGantt1.UpdateBehaviorCollection
   Debug.Print updBeh.Name
Next
```

Active

Property of VcUpdateBehaviorCollection

This property lets you set or retrieve the update behavior that currently is in effect.

	Data Type	Explanation
Property value	VcUpdateBehavior	Update behavior currently used

Example Code

```
Dim updBehCltn As VcUpdateBehaviorCollection
Dim updBeh As VcUpdateBehavior

Set updBehCltn = VcGantt1.UpdateBehaviorCollection
Set updBeh = UpdateBehaviorCltn.Active
```

Count

Read Only Property of VcUpdateBehaviorCollection

This property lets you retrieve the number of update behaviors in the collection.

	Data Type	Explanation
Property value	Long	Number of update behaviors

Example Code

```
Dim updBeh As VcUdateBehvior
Dim updBehCltn As VcUpdateBehaviorCollection
Dim numberOfUpdateBehaviors As Long

Set updBeh =
VcGantt1.UpdateBehviorCollection.UpdateBehviorByName("UPDBEHAVIOR_1")
Set updBehCltn = updBeh.UpdateBehaviorCollection
```

```
numberOfUpdateBehaviors = updBehCltn.Count
```

Methods

Add

Method of VcUpdateBehaviorCollection

By this method you can create an update behavior as a member of the UpdateBehaviorCollection. If the name was not used before, the new update behavior object will be returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
Parameter: ⇨ updateBehaviorName	String **Possible Values:**	Update behavior name Name of the color map
Return value	VcUpdateBehavior	New update behavior object

Example Code

```
Set newUpdateBehavior = VcGantt1.UpdateBehaviorCollection.Add("test1")
```

AddBySpecification

Method of VcUpdateBehaviorCollection

This method lets you create an update behavior by using a specification. This way of creating allows update behavior objects to become persistent. The specification of an update behavior can be saved and re-loaded (see VcUpdateBehavior property **Specification**). In a subsequent session the update behavior can be created again from the specification and is identified by ist name.

	Data Type	Explanation
Parameter: ⇨ Specification	String **Possible Values:**	Update behavior specification Name of the color map
Return value	VcUpdateBehavior	New update behavior object

Example Code

```
Dim updBehCltn As VcUpdateBehaviorCollection

updBehCltn = VcGantt1.UpdateBehaviorCollection
updBehCltn.AddBySpecification(textSpecification)
```

Copy

Method of VcUpdateBehaviorCollection

By this method you can copy an update behavior. If the update behavior that is to be copied exists, and if the name for the new update behavior does not yet exist, the new update behavior object will be returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
Parameter:		
⇨ updateBehaviorName	String **Possible Values:**	Name of the update behavior to be copied Name of the color map
⇨ newUpdateBehaviorName	String **Possible Values:**	Name of the new update behavior Name of the color map
Return value	VcUpdateBehavior	Update behavior object

Example Code

```
Dim updBeh As VcUpdateBehavior
Dim updBehCltn As VcUpdateBehaviorCollection
Dim updBeh As VcUpdateBehavior

Set updBeh = VcGantt1.UpdateBehaviorCollection.FirstUpdateBehavior
Set updBehCltn = histogram.UpdateBehaviorCollection
Set updBeh = updBehCltn.Copy("CurrentUpdateBehavior", "NewUpdateBehavior")
```

FirstUpdateBehavior

Method of VcUpdateBehaviorCollection

This method can be used to access the initial value, i.e. the first update behavior of a update behavior collection and then to continue in a forward iteration loop by the method **NextUpdateBehavior** for the update behaviors following. If there is no update behavior in the update behavior collection, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcUpdateBehavior	First update behavior

Example Code

```
Dim updBeh As VcUpdateBehavior

Set updBeh = VcGantt1.UpdateBehaviorCollection.FirstUpdateBehavior
```

NextUpdateBehavior

Method of VcUpdateBehaviorCollection

This method can be used in a forward iteration loop to retrieve subsequent update behavior from a update behavior collection after initializing the loop by the method **FirstUpdateBehavior**. If there is no update behavior left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcUpdateBehavior	Subsequent update behavior

Example Code

```
Dim updBehCltn As VcUpdateBehaviorCollection
Dim updBeh As VcUpdateBehavior

Set updBehCltn = VcGantt1.UpdateBehaviorCollection
Set updBeh = updBehCltn.FirstUpdateBehavior

While Not updBeh Is Nothing
     List1.AddItem updBeh.Name
     Set updBeh = updBehCltn.NextUpdateBehavior
Wend
```

Remove

Method of VcUpdateBehaviorCollection

This method lets you delete an update behavior. If the update behavior is used in another object, it cannot be deleted. In that case, False will be returned, otherwise True.

	Data Type	Explanation
Parameter: ⇨ updateBehaviorName	String **Possible Values:**	Update behavior name Name of the color map
Return value	Boolean	Update behavior deleted (True)/not deleted (False)

Example Code

```
Dim histogram As VcHistogram
Dim updBehCltn As VcUpdateBehaviorCollection

Set histogram = VcGantt1.HistogramCollection.FirstHistogram
Set updBehCltn = histogram.UpdateBehaviorCollection
updBehCltn.Remove ("CurrentUpdateBehavior")
```

UpdateBehaviorByIndex

Method of VcUpdateBehaviorCollection

This method lets you access a update behavior by its index. If an update behavior of the specified index does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter: ⇨ index	Integer **Possible Values:**	Index of the update behavior Data field index
Return value	VcUpdateBehavior	Update behavior object returned

Example Code

```
Dim updBehCltn As VcUpdateBehaviorCollection
Dim updBeh As VcUpdateBehavior

Set updBehCltn = VcGantt1.UpdateBehaviorCollection
Set updBeh = updBehCltn.UpdateBehaviorByIndex(2)
```

UpdateBehaviorByName

Method of VcUpdateBehaviorCollection

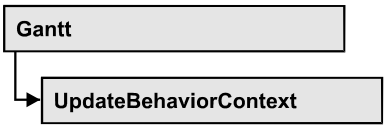
By this method you can retrieve a update behavior by its name. If an update behavior of the specified name does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter: ⇨ updateBehaviorName	String **Possible Values:**	Name of the update behavior Name of the color map
Return value	VcUpdateBehavior	Update behavior

Example Code

```
Dim updBehCltn As VcUpdateBehaviorCollection
Dim updBeh As VcUpdateBehavior

Set updBehCltn = VcGantt1.UpdateBehaviorCollection
Set updBeh = updBehCltn.UpdateBehaviorByName("Standard")
```

7.83 VcUpdateBehaviorContext



An object of the type **VcUpdateBehaviorContext** comprises the context of the update behavior, that is, the behavior of all other objects that are affected by a live update and that can be configured by a user.

Properties

- DelayTime
- IsEditable
- Type
- UpdateMode

Properties

DelayTime

Property of VcUpdateBehaviorContext

This property lets you set the delay time after which the modified objects of the live update visually are to appear while the mouse cursor is moving. Setting this property makes sense only if the property **UpdateMode** was set to **OnPauseWhileMouseMoving**.

	Data Type	Explanation
Property value	Integer	Number of milliseconds **Default value:** 500
	Possible Values:	Data field index

Example Code

```
Dim updBehCtx As VcUpdateBehaviorContext
Dim delTim As Integer

delTim = VcGantt1.updBehCtx.DelayTime
```

IsEditable

Property of VcUpdateBehaviorContext

This property lets you set or retrieve whether the context of the update behavior should be editable at run time.

	Data Type	Explanation
Property value	Boolean	Update behavior context editable (True) / not editable (False) **Default value:** True
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
Dim updBehCtx As VcUpdateBehaviorContext

updBehCtx.Editable = False
```

Type

Read Only Property of VcUpdateBehaviorContext

This property lets you retrieve defined areas (context types) that are affected by the live update and to which the properties **Editable**, **UpdateMode** und **DelayTime** can be applied.

	Data Type	Explanation
Property value	VcUpdateBehaviorContextType	Availabe update areas (types):
	Possible Values:	
	vcHistogramsLayerSourceCurvesCalculations 1101	Curve calculation from layer data in histograms
	vcLinksChangeSuccessorNode 402	Links change their successor node
	vcNodeLevelLayoutsChangeNodesSortingOrder 201	Node level layouts change the sorting order of nodes
	vcNodesAutoScheduling 105	Nodes are automatically scheduled
	vcNodesChangeDatesDuration 101	Nodes change their dates or their duration
	vcNodesFiltering 102	Nodes are filtered
	vcNodesGrouping 104	Nodes are grouped
	vcNumericScalesChangeUnitWidth 1201	Numeric scales change unit width
	vcSashesChangePosition 1501	Sashes change position
	vcTablesChangeColumnWidth 901	Tables change column width
	vcTimeScalesChangeSectionStartDate 1002	Time scales change the start date of a section
	vcTimeScalesChangeUnitWidth 1001	Time scales change unit width

UpdateMode

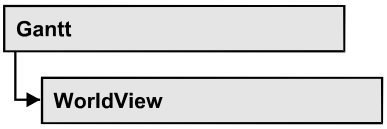
Property of VcUpdateBehaviorContext

In a self-created update behavior this property lets you set or retrieve a cursor action on which the live update is to take place. If this property was set to **OnPauseWhileMouseMoving**, you can set the desired delay time

by the **DelayTime** property.

	Data Type	Explanation
Property value	VcUpdateMode	Available actions of the cursor: **Default value:** vcOnMouseMove
	Possible Values:	
	vcOnMouseMove 1	The update is displayed when the mouse cursor moves
	vcOnMouseUp 0	The update is displayed when the left mouse button is released
	vcOnPauseWhileMouseMoving 2	The update is displayed when a pause occurs during the movements of the mouse cursor

7.84 VcWorldView



An object of the type **VcWorldView** designates the world view window.

Properties

- Border
- BorderColor
- Height
- HeightActualValue
- Left
- LeftActualValue
- MarkingColor
- Mode
- ParentHWnd
- ScrollBarMode
- Top
- TopActualValue
- UpdateBehaviorName
- Visible
- Width
- WidthActualValue

Properties

Border

Property of VcWorldView

This property lets you set or retrieve whether the world view should have a frame (not valid for **vcPopupWindow** mode). The color of the frame is **Color.Black**. This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	Boolean **Possible Values:**	World view with a border line (True)/without border line (False) **Default value:** True Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.WorldView.Mode = vcNotFixed
VcGantt1.WorldView.Border = True
```

BorderColor

Property of VcWorldView

This property lets you set/retrieve the color of the frame that may be visible.

	Data Type	Explanation
Property value	Color RGB ({0...255},{0...255},{0...255})	RGB color values ({0...255},{0...255},{0...255}) **Default value:** 0,0,0

Height

Property of VcWorldView

This property lets you retrieve the vertical extent of the world view. In the modes **vcFixedAtTop**, **vcFixedAtBottom**, **vcNotFixed** and **vcPopupWindow** of the property **Mode** it can also be set.

Please note that the pixel coordinates are system coordinates, i. e. in Visual Basic you have to perform a conversion from/to Twips by the properties **App.TwipsPerPixelX** and **App.TwipsPerPixelY**.

This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	Long	Height of the world view {0, ...} **Default value:** 100

Example Code

```
VcGantt1.WorldView.Height = 100
```

HeightActualValue

Read Only Property of VcWorldView

This property lets you retrieve the vertical extension of the world view which actually is displayed. In the modes b!vcLVFixedAtBottom, **vcLVFixedAtLeft**, **vcLVFixedAtRight**, **vcLVFixedAtTop** the actual value may differ from the one that was set because in these modes either the height or the width is preset.

Please note that the pixel coordinates are system coordinates, i. e. in Visual Basic you have to perform a conversion from/in Twips by the properties **App.TwipsPerPixelX** and **App.TwipsPerPixelY**.

	Data Type	Explanation
Property value	Long	Actual height of the world view {0, ...} **Default value:** 100

Example Code

```
VcGantt1.LegendView.Height = 300
```

Left

Property of VcWorldView

This property lets you retrieve the left position of the Additional Views. In the modes **vcNotFixed** and **vcPopupWindow** of the property **Mode** it can also be set.

Please note that the pixel coordinates are system coordinates, i. e. in Visual Basic you have to perform a conversion from/to Twips by the properties **App.TwipsPerPixelX** and **App.TwipsPerPixelY**.

This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	Long	Left position of the world view **Default value:** 0

Example Code

```
VcGantt1.WorldView.Left = 200
```

LeftActualValue

Read Only Property of VcWorldView

This property lets you retrieve the left position of the world view which actually ist displayed. In the modes b!vcLVFixedAtBottom, **vcLVFixedAtLeft**, **vcLVFixedAtRight**, **vcLVFixedAtTop** the actual value may differ from the one that was set because in these modes either the height or the width is preset.

Please note that the pixel coordinates are system coordinates, i. e. in Visual Basic you have to perform a conversion from/to Twips by the properties **App.TwipsPerPixelX** and **App.TwipsPerPixelY**.

	Data Type	Explanation
Property value	Long	Actual left position of the world view **Default value:** 0

Example Code

```
VcGantt1.LegendView.LeftActualValue = 150
```

MarkingColor

Property of VcWorldView

This property lets you enquire/set the line color of the rectangle that indicates in the Additional Views the currently selected section. This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	Color	RGB color values **Default value:** RGB(0, 0, 255)

Example Code

```
VcGantt1.WorldView.MarkingColor = RGB(255, 0, 0)
```

Mode

Property of VcWorldView

This property lets you enquire/set the Additional Views mode. This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	WorldViewModeEnum	Mode of the world view **Default value:** vcPopupWindow
	Possible Values:	
	vcFixedAtBottom 4	The world view is displayed on the bottom of the control window. The reference system of the coordinates is the control. With this value set, the height can be specified, whereas the position and the width are fixed.
	vcFixedAtLeft 1	The world view is displayed on the left side of the control window. The reference system of the coordinates is the control. With this value set, the width can be specified, whereas the position and the height are fixed.
	vcFixedAtRight 2	The world view is displayed on the right side of the control window. The reference system of the coordinates is the control. With this value set, the width can be specified, whereas the position and the height are fixed.
	vcFixedAtTop 3	The world view is displayed on the top of the control window. The reference system of the coordinates is the control. With this value set, the height can be specified, whereas the position and the width are fixed.
	vcNotFixed 5	The world view is a child window of the current parent window of the control. It can be positioned at any position with any extension. The reference system of the coordinates is the parent window. The child window does not have a frame of its own and cannot be moved interactively by the user. The parent window can be modified by the property **VcWorldView.ParentHWnd**.
	vcPopupWindow 6	The world view is a popup window with its own frame. The reference system of the coordinates is the screen. The user can modify its position and extension, open it by the default context menu and close it by the **Close** button in the frame.

Example Code

```
VcGantt1.WorldView.Mode = vcFixedAtBottom
```

ParentHWnd

Property of VcWorldView

In the **vcNotFixed** mode, this property lets you set the HWnd handle of the parent window, for example, if the world view is to appear in a frame window implemented by your own. By default, the frame window is

positioned on the HWnd handle of the parent window of the VARCHART ActiveX main window. This property can be used only at run time.

	Data Type	Explanation
Property value	OLE_HANDLE	Handle

Example Code

```
MsgBox (VcGantt1.worldview.ParentHWnd)
```

ScrollBarMode

Property of VcWorldView

This property lets you set or retrieve the scroll bar mode of the world view. This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	WorldViewScrollBarModeEnum	Scrollbarmode **Default value:** NoScrollBar
	Possible Values:	
	vcAutomaticScrollBar 3	Display of a horizontal or vertical scrollbar if required.
	vcHorizontalScrollBar 1	Display of a horizontal scrollbar if required.
	vcNoScrollBar 0	The complete chart is displayed without scrollbars.
	vcVerticalScrollBar 2	Display of a vertical scrollbar if required.

Example Code

```
VcGantt1.WorldView.ScrollBarMode = vcAutomaticScrollBar
```

Top

Property of VcWorldView

This property lets you retrieve the top position of the world view. n the modes **vcNotFixed** und **vcPopupWindow** of the property **Mode** it also can be set.

Please note that the pixel coordinates are system coordinates, i. e. in Visual Basic you have to perform a conversion from/to Twips by the properties **App.TwipsPerPixelX** and **App.TwipsPerPixelY**.

This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	Long	Top position of the world view

Example Code

```
VcGantt1.WorldView.Top = 20
```

TopActualValue

Read Only Property of VcWorldView

This property lets you enquire the top position of the world view which actually is displayed. In the modes b!vcLVFixedAtBottom, **vcLVFixedAtLeft**, **vcLVFixedAtRight**, **vcLVFixedAtTop** the actual value may differ from the one that was set because in these modes either the height or the width is preset.

Please note that the pixel coordinates are system coordinates, i. e. in Visual Basic you have to perform a conversion from/to Twips by the properties **App.TwipsPerPixelX** and **App.TwipsPerPixelY**.

	Data Type	Explanation
Property value	Long	Actual top position of the world view **Default value:** 0

Example Code

```
VcGantt1.LegendView.TopActualValue = 40
```

UpdateBehaviorName

Property of VcWorldView

This property lets you set or retrieve the name of the UpdateBehavior.

	Data Type	Explanation
Property value	String **Possible Values:**	Name of the UpdateBehavior Name of the color map

Visible

Property of VcWorldView

This property lets you enquire/set whether the worldview is visible or not. This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	Boolean	World view visible (True)/not visible (False) **Default value:** False
	Possible Values:	Group invisible/visible group nodes are/are not visible

Example Code

```
VcGantt1.WorldView.Visible = True
```

Width

Property of VcWorldView

This property lets you retrieve the horizontal extent of the world view. In the modes **vcFixedAtLeft**, **vcFixedAtRight**, **vcNotFixed** and **vcPopupWindow** of the property **Mode** it also can be set.

Please note that the pixel coordinates are system coordinates, i. e. in Visual Basic you have to perform a conversion from/to Twips by the properties **App.TwipsPerPixelX** and **App.TwipsPerPixelY**.

This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	Long	Horizontal extension of the world view {0, ...} **Default value:** 100

Example Code

```
VcGantt1.WorldView.Width = 200
```

WidthActualValue

Read Only Property of VcWorldView

This property lets you retrieve the horizontal extent of the legend view which actually is displayed. In the modes b!vcLVFixedAtBottom, **vcLVFixedAtLeft**, **vcLVFixedAtRight**, **vcLVFixedAtTop** the actual value may differ from the one that was set because in these modes either the height or width is preset.

Please note that the pixel coordinates are system coordinates, i. e. in Visual Basic you have to perform a conversion from/to Twips by the properties **App.TwipsPerPixelX** and **App.TwipsPerPixelY**.

	Data Type	Explanation
Property value	Long	Actual horizontal extension of the world view {0, ...} **Default value:** 100

Example Code

```
VcGantt1.LegendView.WidthActualValue = 600
```

8 Index

B

C

D

E

F

G

I

K

M

O

P

R

S

T

U

V