

# VARCHART XGantt

ASP.NET Edition 4.3  
User's and  
Reference Guide



# **VARCHART XGantt ASP.NET Edition**

**Version 4.3**

**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](mailto:sales@netronic.com)  
[www.netronic.com](http://www.netronic.com)

© Copyright 2009 NETRONIC Software GmbH  
All rights reserved.

Information in this document is subject to change without notice and does not represent a commitment on the part of NETRONIC Software GmbH. The software described in this document is furnished under a license agreement. The software may be used or copied only in accordance with the terms of the agreement. It is against the law to copy documentation on magnetic tape, disk, or any other medium for any purpose other than the purchaser's personal use.

Microsoft Windows, Microsoft Explorer, Microsoft Visual Basic and Microsoft Visual Studio are trademarks of MICROSOFT Corp., USA.

Last Revision: 1. September 2009

# Table of Contents

---

<b>1</b>	<b>Introduction</b>	<b>11</b>
1.1	VARCHART XGantt at a Glance	11
1.2	Installation	13
1.3	Licensing	14
1.4	Delivery	16
1.5	Usage of the German version	18
1.6	Support and Advice	20

---

<b>2</b>	<b>Tutorial</b>	<b>21</b>
2.1	Overview	21
2.2	Placing the Control on a Web Page	22
2.3	Supplying Data	23
2.4	Calculating End Dates	29
2.5	Marking Non-working Intervals in Activities	33
2.6	Interactions in the VARCHART XGantt Web Server Control	35
2.7	Using Layers	39
2.8	Using Filters	42
2.9	Creating Histograms	45
2.10	Exporting a Diagram	61
2.11	Saving the Configuration	62

---

<b>3</b>	<b>Important Concepts</b>	<b>63</b>
3.1	AJAX Extensions	63
3.2	Boxes	65
3.3	Data Tables	69
3.4	Date Lines	76
3.5	Dates and Daylight Saving Time	77
3.6	Events	79
3.7	Filters	80

## 4 Table of Contents

3.8	Graphics Formats	82
3.9	Grouping	86
3.10	Hierarchical Order	90
3.11	Histograms	92
3.12	How to Use a Calendar	98
3.13	Layers	99
3.14	Link Appearance	102
3.15	Links	103
3.16	Localization of Text Output	107
3.17	Maps	109
3.18	MultiState Fields	114
3.19	Node (Activity)	116
3.20	Resource Scheduler	117
3.21	Schedule	121
3.22	Sorting	124
3.23	Table	131
3.24	Time Scale	133
3.25	Tooltips during Runtime	139
3.26	Unicode	140
3.27	Viewer Metafile (*.vmf)	141
3.28	Writing PDF files	142

---

## **4 Property Pages and Dialog Boxes 145**

4.1	General Information	145
4.2	The "General" Property Page	146
4.3	The "Border Area" Property Page	152
4.4	The "Nodes" Property Page	154
4.5	The "Layout" Property Page	158
4.6	The "Objects" Property Page	162
4.7	The "Links" Property Page	164
4.8	The "Schedule" Property Page	166
4.9	The "Administrate Data Tables" Dialog Box	168
4.10	The "Specify Bar Appearance" Dialog Box	171
4.11	The "Edit Layer" Dialog Box	175

4.12	The "Edit Layer Format" Dialog Box	180
4.13	The "Administrate Filters" Dialog Box	184
4.14	The "Edit Filter" Dialog Box	186
4.15	The "Administrate Line formats" Dialog Box	190
4.16	The "Edit Line format" Dialog Box	192
4.17	The "Grouping" Dialog Box	195
4.18	The "Administrate Calendar grids" Dialog Box	203
4.19	The "Administrate Line grids" Dialog Box	205
4.20	The "Administrate Maps" Dialog Box	208
4.21	The "Edit Map" Dialog Box	210
4.22	The "Configure Mapping" Dialog Box	212
4.23	The "Administer Boxes" Dialog Box	213
4.24	The "Edit Box" Dialog Box	216
4.25	The "Administrate Box Formats" Dialog Box	217
4.26	The "Edit Box Format" Dialog Box	219
4.27	The "Administrate Link Appearances" Dialog Box	222
4.28	The "Specify Table" Dialog Box	226
4.29	The "Edit Table" Dialog Box	228
4.30	The "Edit Table Format" Dialog Box	230
4.31	The "Edit Line Attributes" Dialog Box	235
4.32	The "Edit Pattern Attributes" Dialog Box	236
4.33	The "Specify Calendars" Dialog Box	237
4.34	The "Administrate Intervals" Dialog Box (Calendar)	239
4.35	The "Administrate Calendar Profiles" Dialog Box	241
4.36	The "Administrate Intervals" Dialog Box (Calendar Profiles, Profile Type <Day Profile>)	242
4.37	The "Administrate Intervals" Dialog Box (Calendar Profiles, Profile Type <Week Profile>)	243
4.38	The "Administrate Intervals" Dialog Box (Calendar Profiles, Profile Type <Variable Profile>)	244
4.39	The "Administrate Intervals" Dialog Box (Calendar Profiles, Profile Type <Year Profile>)	246
4.40	The "Specify Time Scale" Dialog Box	248
4.41	The "Edit Time Scale Section" Dialog Box	251
4.42	The "Administrate Histograms" Dialog Box	257

## 6 Table of Contents

4.43	The "Edit Histogram" Dialog Box	259
4.44	The "Select Curve Data Source" Dialog Box	263
4.45	The "Select Ribbon Type" Dialog Box	264
4.46	The "Specify Date Lines" Dialog Box	266
4.47	The "Edit Date Line" Dialog Box	268
4.48	The "Specification of Texts, Graphics and Legend" Dialog Box	270
4.49	The "Legend Attributes Dialog Box"	273
4.50	The "Licensing" Dialog Box	275
4.51	The "Request License Information" Dialog Box	277

---

## 5 Frequently Asked Questions **279**

5.1	What Needs to be Done if the .NET-Framework was Installed, but still an Error Message Occurs Saying that the Wrong Version of .NET was Installed?	279
5.2	Does VARCHART XGantt ASP.NET Require the Session State?	280
5.3	What Type of Storage of the Session State does VARCHART XGantt ASP.NET Require?	281
5.4	What Is the Relation Between Sessions and Cookies ?	282
5.5	What Needs to be Considered When Loading Data?	283
5.6	What Can I do if in the Web Application or the Programming Environment a Fatal Error Occurs?	284
5.7	How can I Move a Bar into the Visible Area by Clicking on the Table?	285
5.8	How can I Make Overlapping Activities in a Group Visible?	286
5.9	How can I Save and Reload the Order of Activities?	287
5.10	How can I Improve the Performance?	288
5.11	What to do if the Control Does Not Work With a User Account of a Computer	290
5.12	Can All Fonts be Used?	291
5.13	How to find the Name of the IIS Worker Process	292
5.14	What if an ASPX Page is Displayed as a HTML Page?	293
5.15	How can Jittering and "Scrolling Away" be Avoided When Clicking on the Control?	294
5.16	Why Does the Message "Page not found" Occur When Invoking a HTML or an ASPX Page, Although the Page Does Exist?	295
5.17	How Can a URL be Put on a Node?	296

5.18	How Can a Reload of a Page be Initialized When a Session Timeout Occurs?	297
5.19	How to Use XGantt Best Together With Microsoft's ASP .NET AJAX ?	298

---

## **6 API Reference 299**

6.1	Object Types	299
6.2	VcBorderArea	301
6.3	VcBorderBox	303
6.4	VcBox	311
6.5	VcBoxCollection	321
6.6	VcBoxFormat	328
6.7	VcBoxFormatCollection	333
6.8	VcBoxFormatField	340
6.9	VcCalendar	349
6.10	VcCalendarCollection	358
6.11	VcCalendarGrid	365
6.12	VcCalendarProfile	376
6.13	VcCurve	379
6.14	VcCurveCollection	411
6.15	VcDataDefinition	418
6.16	VcDataDefinitionField	420
6.17	VcDataDefinitionTable	425
6.18	VcDataRecord	431
6.19	VcDataRecordCollection	437
6.20	VcDataTable	445
6.21	VcDataTableCollection	448
6.22	VcDataTableField	454
6.23	VcDataTableFieldCollection	461
6.24	VcDateLine	467
6.25	VcDateLineCollection	475
6.26	VcDateLineGrid	480
6.27	VcFilter	489
6.28	VcFilterCollection	495



## 8 Table of Contents

6.29	VcFilterSubCondition	501
6.30	VcGantt	506
6.31	VcGroup	639
6.32	VcGroupCollection	650
6.33	VcGroupLevelLayout	654
6.34	VcGroupLevelLayoutCollection	669
6.35	VcHistogram	674
6.36	VcHistogramCollection	681
6.37	VcInterval	686
6.38	VcIntervalCollection	699
6.39	VcLayer	704
6.40	VcLayerCollection	734
6.41	VcLayerFormat	740
6.42	VcLayerFormatField	743
6.43	VcLineFormat	750
6.44	VcLineFormatCollection	753
6.45	VcLineFormatField	759
6.46	VcLink	767
6.47	VcLinkAppearance	773
6.48	VcLinkAppearanceCollection	782
6.49	VcLinkCollection	789
6.50	VcMap	793
6.51	VcMapCollection	799
6.52	VcMapEntry	806
6.53	VcNode	816
6.54	VcNodeCollection	827
6.55	VcNodeLevelLayout	831
6.56	VcNumericScale	839
6.57	VcNumericScaleCollection	848
6.58	VcPrinter	853
6.59	VcRect	872
6.60	VcResourceScheduler2	876
6.61	VcRibbon	937
6.62	VcScheduler	947

6.63	VcSection	955
6.64	VcTable	960
6.65	VcTableCollection	964
6.66	VcTableFormat	965
6.67	VcTableFormatCollection	972
6.68	VcTableFormatField	976
6.69	VcTimeScale	988
6.70	VcTimeScaleCollection	993

---

**7 Index 997**



---

---

# 1 Introduction

---

## 1.1 VARCHART XGantt at a Glance

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

VARCHART XGantt is an interactive graphical 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 can meet individual graphical demands.

VARCHART XGantt ASP.NET is a web server control which was completely syntonized to the Microsoft .NET framework.

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

- Creating, deleting or shifting of nodes
- Creating and deleting of links (linking nodes)
- 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
- Exchange of the application data via files or the programming interface
- Various design options for histograms
- Easy customization of properties via the property pages
- Powerful programming interface

## 12 Introduction

**Note:** The source code samples of this documentation are written in VB.NET and C#.

---

## 1.2 Installation

To develop an application on the basis of .NET you need a developing environment such as Microsoft Visual Studio 2005 that supports the .Net framework 2.0 at least and is compatible with mixed-mode components. As operating system only the 32bit or 64bit (x64) editions of Windows from XP upwards can be used.



To install the VARCHART XGantt ASP.NET control on your computer, please start the setup program and follow the instructions.

By default, the control and its associated files will be stored in a folder located at

**c:\Program Files\VARCHART\XGantt ASP.NET .**

After installing you should add the control to the toolbox of your developing environment.

We give an example of how to proceed in Microsoft Visual Studio 2005; in other development environments the procedure is similar:

1. In Visual Studio please create a new project of the type **ASP.NET Web Application**. It doesn't matter which language you choose, but please mind that the toolbox be visible. If it is not, click on **View > Toolbox**.
2. Open the context menu by a right mouse click on the toolbox and select **Choose Items....**
3. By clicking on **Browse** of the tab **.NET Framework Components** you can choose the assembly **NETRONIC.XGantt.Web.dll** from the installation directory. After confirming by **OK**, the icon of VARCHART XGantt ASP.NET  is added to the toolbox.
4. To add the symbol for the **UpdatePanel**  to your toolbox, choose the assembly **NETRONIC.Web.dll**.

---

## 1.3 Licensing

For licensing the VARCHART XGantt.ASP.Net control please click the icon  and draw the control onto the ASP.Net page.

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 licensing information will open.

Four items are needed for the registration:

- the name of the company
- the name of a staff member
- the registration number
- the hardware identification

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 containing the required information by yourself. Please send all enquiries concerning the licensing to [registration@netronic.com](mailto:registration@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 **NETRONIC.XGantt.Web.dll**).

### 1.3.1 Integration of a License File Into an ASP.NET Website Project of Visual Studio 2005

In a website project, Visual Studio 2005 stores the license information for XGantt ASP.NET to the file `app_licenses.dll`. When building a project (Build > Rebuild all), the file will not be updated. After receiving a new license file, the acceptance of the new license needs to be enforced which requires to go through the steps described below:

1. Open the project in VS 2005
2. In the project explorer, click on the file `NETRONIC.XGantt.Web.dll`, which you can find in the directory **bin**. In the field **Auto-refresh path**

the property window indicates the folder from which the file **NETRONIC.XGantt.Web.dll** is updated automatically.

3. Please close VS 2005.
4. Please copy the license file **NETRONIC.XGantt.Web.VcGantt.lic** to the folder mentioned in step 2.
5. Please open the project in VS 2005.
6. Please rebuild the project by clicking on **Build > Rebuild all**. The license file is now copied to the **bin** folder of the project.
7. Please close VS 2005 and re-open the project in VS 2005.
8. In the project explorer, please click on the file **licenses.licx** by using the right mouse button and select **build runtime licenses** from the pop-up menu. The file **app\_licenses.dll** will be rebuilt and the new license information will be copied from the file **NETRONIC.XGantt.Web.-VcGantt.lic** to the DLL file.
9. Please close VS 2005 and re-open the project in VS 2005.
10. Re-build the project by **Build > Rebuild all** once again.

### **1.3.2 Run Time Licenses**

If you have developed a web application with XGantt ASP.NET this application runs on your development computer but not on a server operating system. For running your application on a server you need a special run time license that you can purchase from NETRONIC under specification of the IP address of the server on which the application is to be installed and the developing license number. You will receive an **rlic** file (**NETRONIC.XGantt.Web.VcGantt.rlic**). Please copy this file into the directory containing the ASPX page with the XGantt control.

**Tip:** Please note that a run time license is always assigned to a developer license. If you own more than one developer licenses the application has to be always created with the developer license for which you have obtained the run time license.



---

### 1.4 Delivery

If you wish to deliver to a customer an application developed by yourself having used VARCHART XGantt ASP.NET, the below files have to be included:

*NETRONIC.XGantt.Web.dll*

*NETRONIC.XGanttd.Web.dll* (if you want to use the German version)

*MFC80u.dll*

*msvcp80.dll*

*msvcr80.dll*

*gdiplus.dll*

*ImageServer.aspx*

*NETRONIC.Web.dll* (from version 4.2 onward)

*NETRONIC.XGantt.Web.VcGantt.rlic* (will only be made available by NETRONIC when purchasing a run time license)

*opsaps.dll* (only if your application uses the resource scheduling module)

All other files belonging to VARCHART XGantt ASP.NET are only used during the phase of development and must **not** be passed on to your customers.

Please ensure to own a correct run time license. VARCHART XGantt ASP.NET only runs on a server if the run time license file **NETRONIC.XGantt.Web.VcGantt.rlic** is in the folder of the web application. The run time license refers to the IP address of the server computer.

In order to install the three libraries *mfc80u.dll*, *msvcp80.dll* and *msvcr80.dll* please use the setup file *vc redistrib\_vs2005sp1\_x86.exe*. If the x\_64 version of VARCHART XGantt

is applied, the setup file *vc redistrib\_vs2005sp1\_x64.exe* has to be used instead. You should only use the version which is included in our setup. You will find it in the installation folder of XGanttASP.Net in the subfolder **redist**. For further information please see: [msdn2.microsoft.com/en-us/library/ms235285\(VS.80\).aspx](http://msdn2.microsoft.com/en-us/library/ms235285(VS.80).aspx).

When using Windows NT 4.0 or 2000 the library *gdiplus.dll* should be installed locally with your application (recommendation of Microsoft). If you use Windows XP or Server 2003 or later, this library is already installed as

part of the operating system and must not be overwritten. For information about this, please see:

[msdn.microsoft.com/library/default.asp?url=/library/en-us/gdicpp/GDIPlus/GDIPlus.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/gdicpp/GDIPlus/GDIPlus.asp)

VARCHART XGantt ASP.NET can be run on the platforms:

- on the server side:

- Windows Server 2003
- Windows Vista
- Windows XP SP2 or later
- Windows 2000 SP3 or later

using the .Net framework 2.0 (for further information, see

[msdn.microsoft.com/netframework/technologyinfo/sysreqs/default.aspx](http://msdn.microsoft.com/netframework/technologyinfo/sysreqs/default.aspx))

- Internet Information Services 5.0 or higher

- on the client side:

- JavaScript compatible HTML browser

**Tips:**

- How to check which .NET Framework is already installed:

In the **Control Panel** double click on the **Software** icon and look for 'Microsoft .NET Framework' in the list of applications.

- To be able to use the AJAX extensions, a browser with activated JavaScript is required on the client side.

---

## 1.5 Usage of the German version

The VARCHART XGanttASP .NET Edition is available in German and in English. When installing the German version, the resource assembly NETRONIC.XGanttD.Web.dll is copied to the installation directory in addition to the control assembly NETRONIC.XGantt.Web.dll.

### Usage at design time

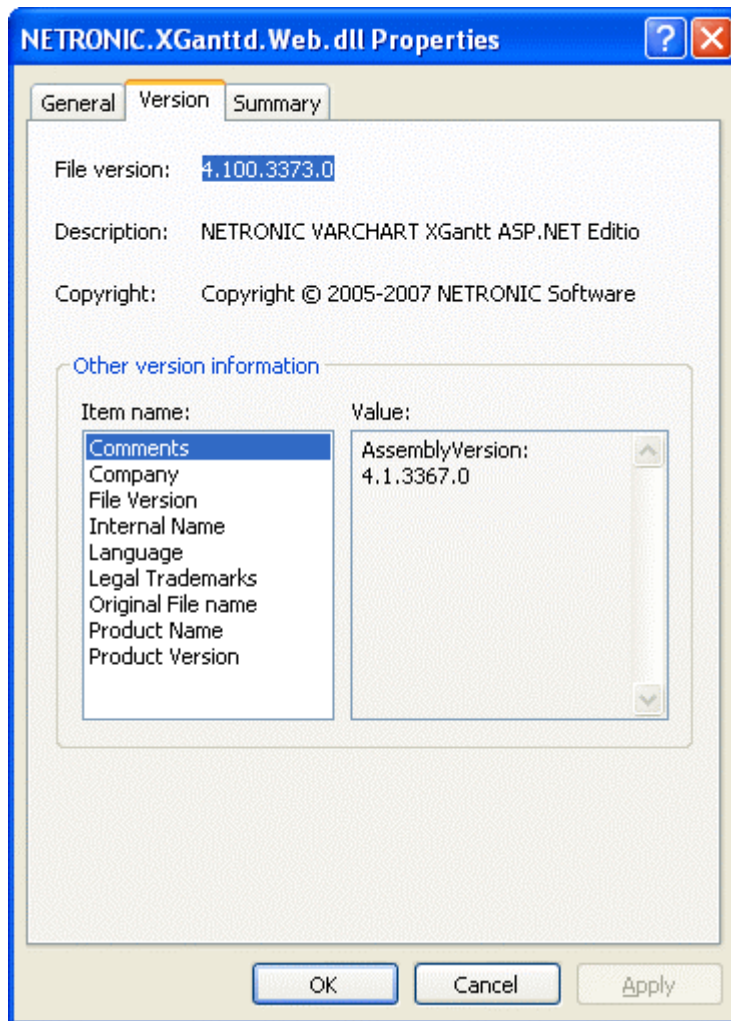
If the **Regional Options** (Control Panel, Regional and Language Options) were set to **German**, the resource assembly is loaded from the installation directory and the German dialogs and property pages are available at design time.

### Usage at run time

If you want to make sure that the resource assembly is used at run time as well and German dialogs are available you have to copy the resource assembly to the application directory. For this, a reference to the assembly has to be added in the project ("Add Reference").

**Tip:** Because the development environment sets the parameter "Copy-Local" to **False** by default, you will have to set it to **True** manually. When the solution is rebuilt afterwards, the resource assembly is copied to the according application directory and will be loaded from there.

In case of problems you should check whether the file version numbers of the assemblies match (Windows Explorer, context menu of the file, **Properties**, tab **Version**).



---

## 1.6 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](mailto:support@netronic.com)

[www.netronic.com](http://www.netronic.com)

...by the way: you may order our support and maintenance service that lasts longer than 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 tutorial, we will get you acquainted with the fundamentals of VARCHART XGantt that are essential for integrating a bar chart into your own web application.

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

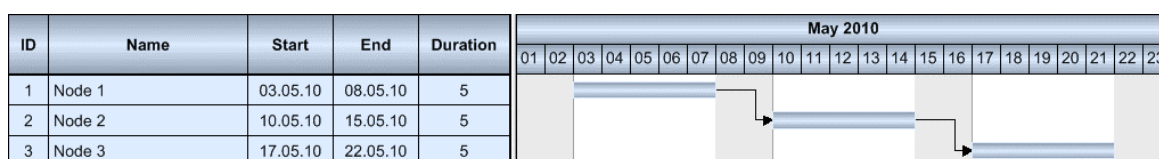
- **Property pages and dialogs**

In this 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 a single code line.

- **API Reference**

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


As the developing environment for the code samples, we use Visual Studio .NET 2005. Our first program sample will show the below result:



The program sample will primarily demonstrate the inbuilt interactions of VARCHART XGantt.

---

## 2.2 Placing the Control on a Web Page

To place the VARCHART XGantt control on the web page, please select it  in the toolbox, then draw a frame by mouse on the page at the position where you want it to appear.

Beside, the ASP.NET page **ImageServer.aspx** needs to be made available on the web site. VARCHART XGantt needs it to send image data as a stream from the server to the client. The file is stored to the the installation folder during the installation process. Please copy it to the folder of your web application.

**Tip:**

A "name space" instruction at the beginning of the program will save you the detailed reference indication when using data types and "enum" elements.

VB: Imports NETRONIC.XGantt.Web

C#: using NETRONIC.XGantt.Web

For example instead of **NETRONIC.XGantt.Web.VcNodeCollection** you only need to write **VcNodeCollection**.

► **XGantt ASP.NET and Windows SharePoint Services**

Currently the only way to integrate VARCHART XGantt ASP.NET into Windows SharePoint Services is to provide a web application which implements the required Gantt functionality inside an ASP.NET web page. Subsequent to that a link to the web page containing the XGantt control has to be added to SharePoint using the Page Viewer Web Part.

The web application may be placed anywhere on the web server the SharePoint system resides on or it may be installed on any other web server.

## 2.3 Supplying Data

For activities and links to be displayed, VARCHART XGantt needs the supply of data. By default, for the communication associated two tables are used:

1. NodeTable (also called Maindata)
2. LinkTable (also called Relations)

When placing a VARCHART XGantt on a web page, the basic fields were already provided in advance.

### Fields of the Maindata data table:

Index	Name	Primary key	Type	DateFormat	Editable	Hidden
0	ID	True	Integer		True	False
1	Name	False	String		False	False
2	Start	False	DateTime	DD.MM.YYYY	False	False
3	End	False	DateTime	DD.MM.YYYY	True	False
4	Duration	False	Integer		False	False

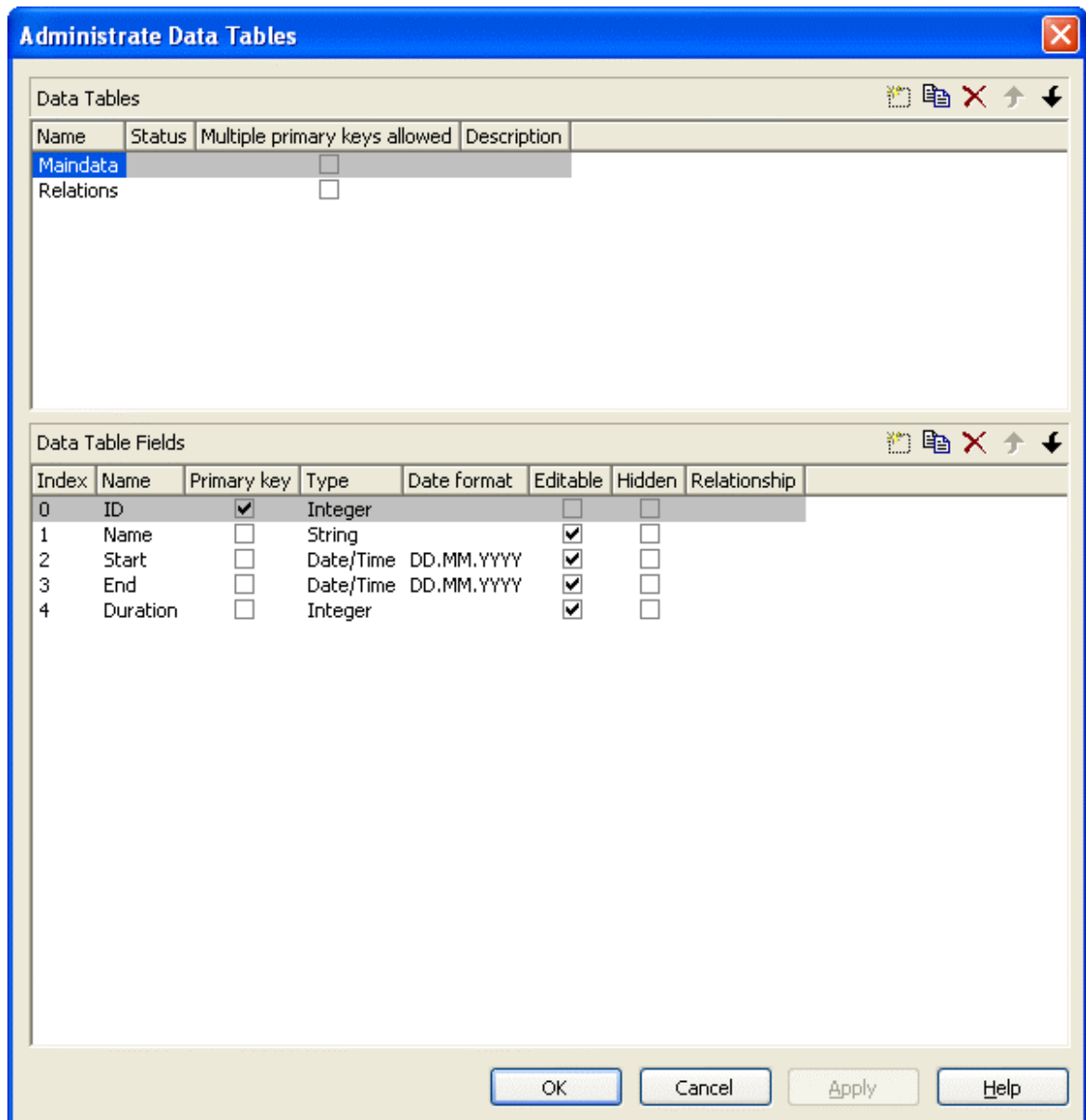
### Fields of the Relations data table:

Index	Name	Primary key	Type	Editable	Hidden
0	Link ID	True	String	False	False
1	Predecessor Node ID	False	String	True	False
2	Successor Node ID	False	String	True	False

Further fields required need to be defined manually. You can do this at design time by the dialog **Administrative Data Tables** (lower section) or at run time by the method **Add(...)** of the object **VcDataTableFieldCollection**.

If you need more tables than the ones defined by default you can create them in the upper section of the dialog box **Administrative Data Tables** after having clicked **Extended data tables enabled** on the property page **General**.





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 VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecCltn As VcDataRecordCollection

VcGantt1.ExtendedDataTablesEnabled = True
```

```

dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
dataRecCltn = dataTable.DataRecordCollection
dataRecCltn.Add("1;Node 1;07.05.2010;;5")
dataRecCltn.Add("2;Node 2;14.05.2010;;5")
dataRecCltn.Add("3;Node 3;21.05.2010;;5")

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

VcGantt1.EndLoading

```

### Example Code C#

```

vcGantt1.ExtendedDataTablesEnabled = true;
VcDataTable dataTable =
vcGantt1.DataTableCollection.DataTableByName("Maindata");
VcDataRecordCollection dataRecCltn = dataTable.DataRecordCollection;
dataRecCltn.Add("1;Node 1;07.05.2010;;5");
dataRecCltn.Add("2;Node 2;14.05.2010;;5");
dataRecCltn.Add("3;Node 3;21.05.2010;;5");

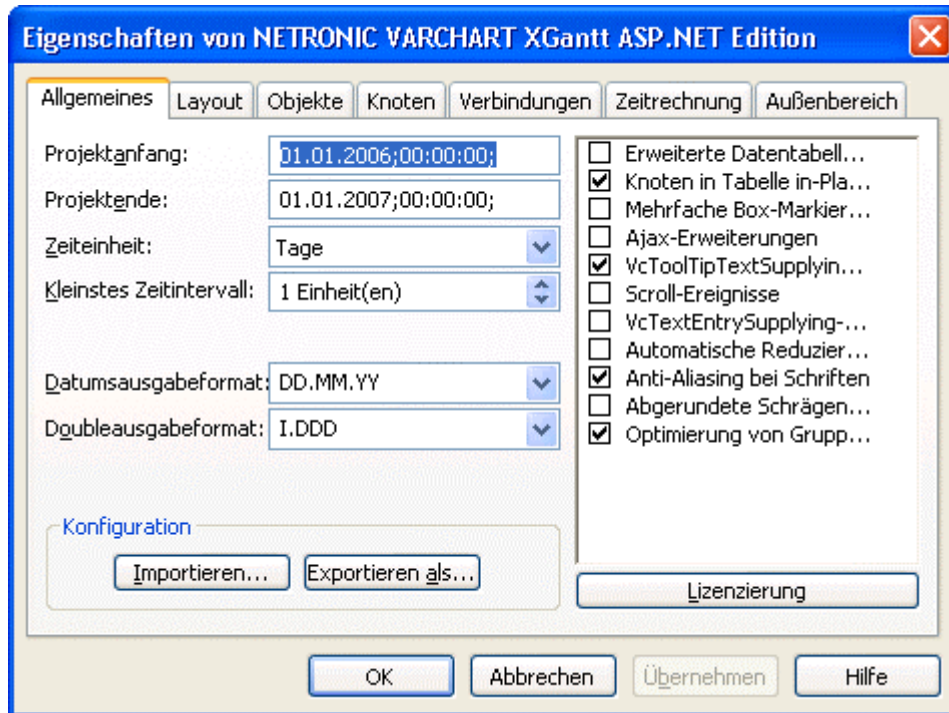
dataTable =
vcGantt1.DataTableCollection.DataTableByName("Relations");
dataRecCltn = dataTable.DataRecordCollection;
dataRecCltn.Add("1;1;2");
dataRecCltn.Add("2;2;3");

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 settings of **Time unit**. It is pre-set to days, which you can modify on the **General** property page.

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



### Loading data from a CSV file

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

#### Example Code

```
1;Node 1;07.05.2010;;5;
2;Node 2;14.05.2010;;5;
3;Node 3;21.05.2010;;5;
****
1;1;2;
2;2;3;
```

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

The records of the Maindata data table are listed first, followed by the records of the Relations data table. Use **\*\*\*\* Table name \*\*\*\*** to mark the beginning of a record group.

If you saved this kind of file for example by the name **intro.csv**, you may import the data as follows:

#### Example Code VB.NET

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

**Example Code C#**

```
vcGantt1.Open(@"c:\intro.csv");
```

**Specifying the period of time which is represented**

Until now, you will see no activities, because the time scale has not been adjusted to the corresponding period. The displayed range of the time scale can be defined via the properties **TimeScaleStart** and **TimeScaleEnd** or determined from the data by the method **OptimizeTimeScaleStartEnd(...)** of the object **VcGantt**.

**Example Code VB.NET**

```
VcGantt1.TimeScaleEnd = New DateTime(2011, 1, 1)
VcGantt1.TimeScaleStart = New DateTime(2010, 5, 4)
```

**Example Code C#**

```
vcGantt1.TimeScaleEnd = new DateTime(2011,1,1);
vcGantt1.TimeScaleStart =new DateTime(2010,5,4);
```

Below you can find the code which you will need for our starter sample.

**Example Code VB.NET**

```
Private Sub Page_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

    Dim dataTable As VcDataTable
    Dim dataRecCltn As VcDataRecordCollection

    vcGantt1.ExtendedDataTablesEnabled = True
    dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
    dataRecCltn = dataTable.DataRecordCollection
    dataRecCltn.Add("1;Node 1;03.05.2010;;5")
    dataRecCltn.Add("2;Node 2;08.05.2010;;5")
    dataRecCltn.Add("3;Node 3;15.05.2010;;5")

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

    VcGantt1.EndLoading()

    VcGantt1.OptimizeTimeScaleStartEnd(3)
End Sub
```

**Example Code C#**

```
private void Page_Load(object sender, System.EventArgs e)
{
```

## 28 Supplying Data

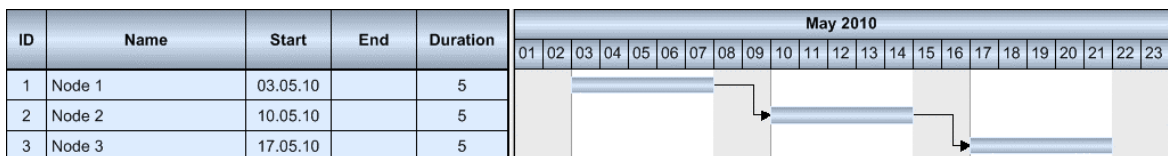
```
vcGantt1.ExtendedDataTablesEnabled = true;
VcDataTable dataTable =
vcGantt1.DataTableCollection.DataTableByName("Maindata");
VcDataRecordCollection dataRecCltn = dataTable.DataRecordCollection;
dataRecCltn.Add("1;Node 1;03.05.2010;;5");
dataRecCltn.Add("2;Node 2;08.05.2010;;5");
dataRecCltn.Add("3;Node 3;15.05.2010;;5");

dataTable =
vcGantt1.DataTableCollection.DataTableByName("Relations");
dataRecCltn = dataTable.DataRecordCollection;
dataRecCltn.Add("1;1;2");
dataRecCltn.Add("2;2;3");

vcGantt1.EndLoading();

vcGantt1.OptimizeTimeScaleStartEnd(3);
}
```

If you run the program now, the result should be as shown in the below illustration.



## 2.4 Calculating End Dates

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

In the default calendar, the weekdays (Monday to Friday) are pre-defined as active times and the weekends (Saturday and Sunday) are defined as non-active times.

You can recognize the non-active times in the diagram by their gray background. The calendar may be switched off by deactivating the option **Nodes use calendar** on the **Nodes** property page.

Please note the difference in calculating when using and when not using a calendar:

An activity which starts on Friday and lasts for 3 days will end on Tuesday if the calendar is activated. Without a calendar, the activity will finish on Sunday already.

The end date can be calculated by using the method **AddDuration(...)** of the object **VcCalendar**. This requires the **start** and the **duration** of each activity. The fields can be accessed via their index. After having set the end date by the method **set\_DataField(...)**, the method **Update()** of **VcNode** needs to be invoked for the modifications to be displayed.

### Example Code VB.NET

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

### Example Code C#

```
VcCalendar tmpCal = vcGantt1.CalendarCollection.Active;
DateTime tmpDate = tmpCal.AddDuration((DateTime)node.get_DataField(2),
                                     Convert.ToInt32(node.get_DataField(4)));
node.set_DataField(3, tmpDate);
node.Update();
```

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

## 30 Calculating End Dates

ID	Name	Start	End	Duration															
					01	02	03	04	05	06	07	08	09						
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															
					01	02	03	04	05	06	07	08	09						
1	Node 1	03.05.10	08.05.10	5															

Dates that were generated by calculation are always placed 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 VB.NET

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

### Example Code C#

```
tmpDate = tmpCal.AddDuration((DateTime)node.get_DataField(3), (-1) *  
Convert.ToInt32(node.get_DataField(4)));  
node.set_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 code was resumed in a separate method called **SetNodeEndDate(...)**.

### Example Code VB.NET

```
Private Sub SetNodeEndDate(ByVal node As VcNode)  
    'Avoid empty duration 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  
        tmpCal = VcGantt1.CalendarCollection.Active  
        Dim tmpDate As DateTime  
        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.Update()
    End If
End Sub

```

### Example Code C#

```

private void SetNodeEndDate(VcNode node)
{
    // Avoid empty duration or negative duration
    if ((string) node.get_DataField(4) == "" ||
        Convert.ToInt32(node.get_DataField(4)) < 0)
        node.set_DataField(4, "0");

    // Start Date empty then end date should also be empty
    if (node.get_DataField(2).ToString() == "31.12.1899 00:00:00")
        node.set_DataField(3, "");
    else
    {
        // Precondition in property page nodes
        // "Assign calendar to nodes" must be true
        VcCalendar tmpCal = vcGantt1.CalendarCollection.Active;

        DateTime tmpDate = tmpCal.AddDuration(
            (DateTime)node.get_DataField(2),
            Convert.ToInt32(node.get_DataField(4)));
        node.set_DataField(3, tmpDate);

        // start date only in active times
        tmpDate = tmpCal.AddDuration((DateTime)node.get_DataField(3),
            (-1) * Convert.ToInt32(node.get_DataField(4)));
        node.set_DataField(2, tmpDate);
        node.Update();
    }
}

```

The calculation of dates is required

1. after activities were loaded
2. after the values of activities were modified via the API

A computational loop for all nodes can be established via the property **NodeCollection** of **VcGantt**. The code will be added to the event **Page\_Load(...)** at the end.

### Example Code VB.NET

```

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

```

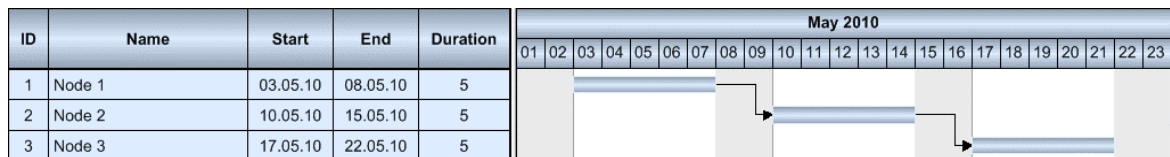


## 32 Calculating End Dates

### Example Code C#

```
// Calculate end date for all nodes
foreach (VcNode node in vcGantt1.NodeCollection)
{
    SetNodeEndDate(node);
}
```

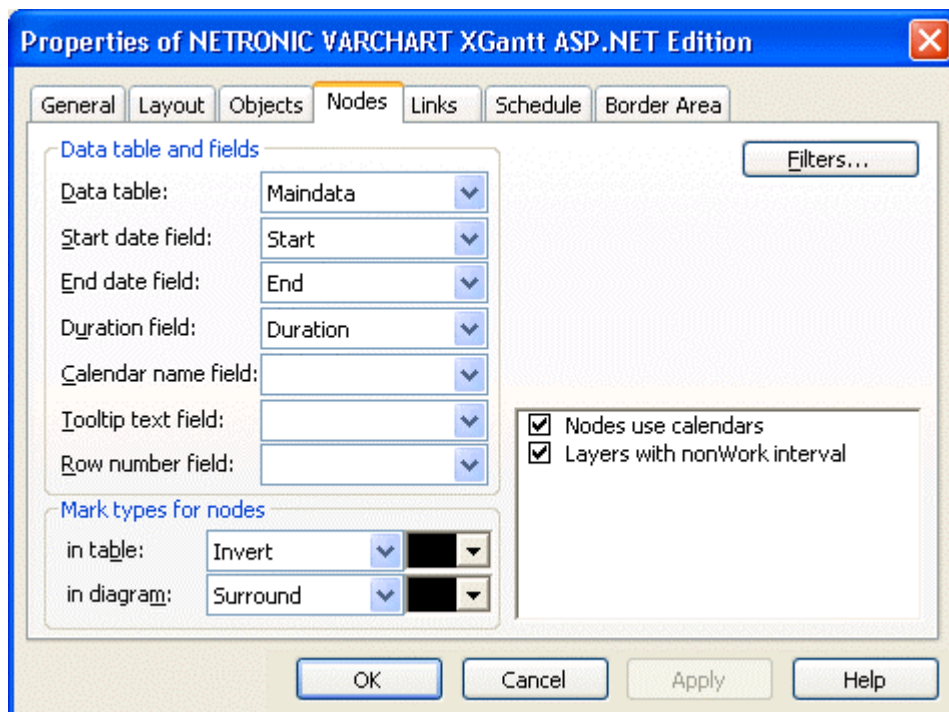
If values of data were altered by the API, the method **SetNodeEndDate(...)** has to be invoked explicitly.



## 2.5 Marking Non-working Intervals in Activities

The visual interruption of the activities by non working intervals can be displayed by setting the option **Layers with nonWork interval**. The option only shows if the activities depend on a calendar. To link nodes to a calendar, you can set the option **Nodes use calendars**.

The option can be activated during run time or during design time. At design time, on the property page **Nodes** you can activate the option **Layers with nonWork interval**.



At runtime you can set the property **LayersWithNonWorkInterval** of the object **VcGantt**.

May 2010															
04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	20
[A blue bar representing a non-working interval spans from the 6th to the 18th of May 2010.]															

*LayersWithNonWorkInterval = false*

### 34 Marking Non-working Intervals in Activities

May 2010																
04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20

*LayersWithNonWorkInterval = true*

## 2.6 Interactions in the VARCHART XGantt Web Server Control

VARCHART XGantt ASP.NET being a web server control is run on the server while on the client only a browser is required. On the client, no programming code is active except for scripts generated by the .NET framework. Therefore, compared to the .NET edition, interactions of the ASP.NET edition are significantly reduced. Only simple clicks by the left mouse button can be recognized and are accepted. This results in the below list of interactions:

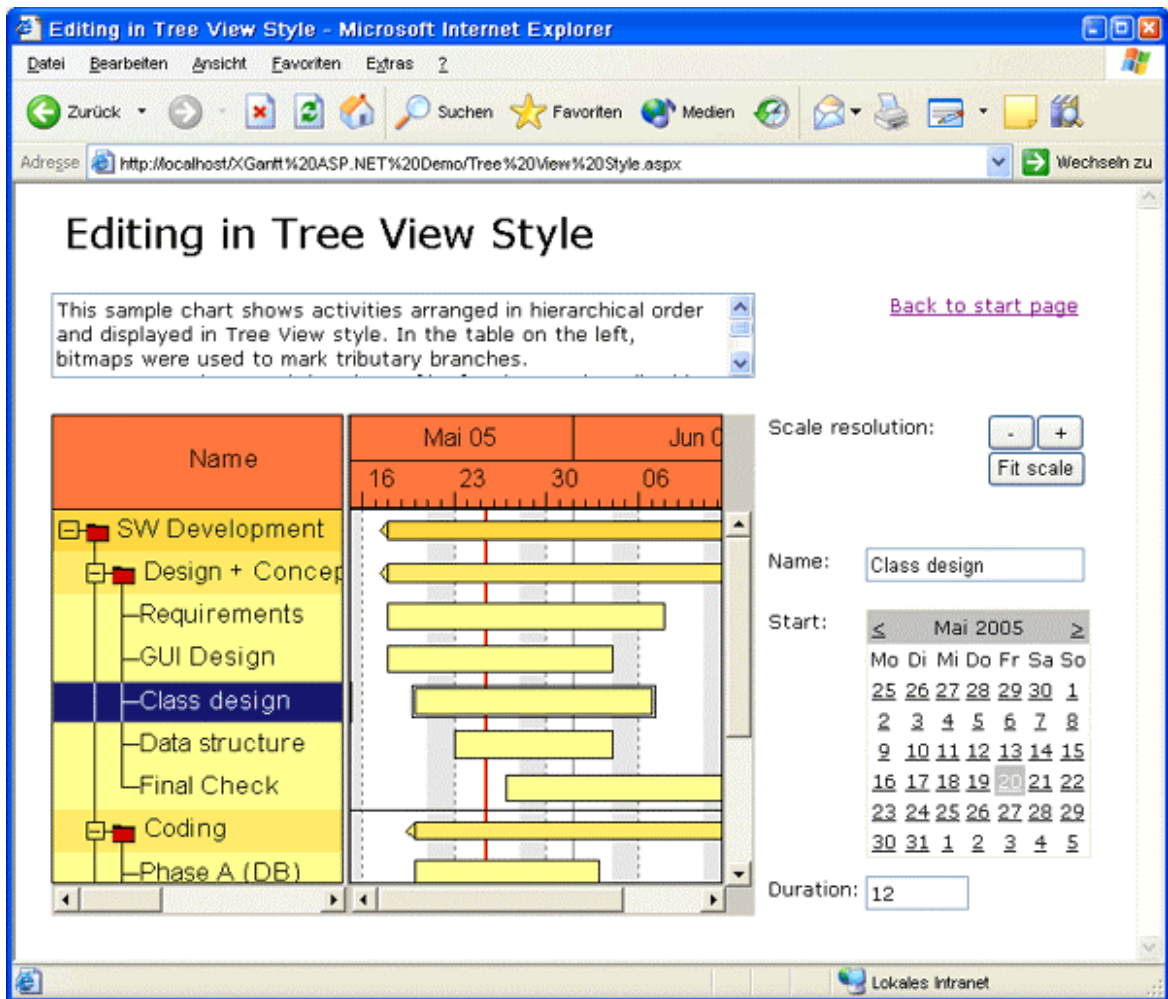
- Selecting activities and table fields
- Scrolling parts of the diagram by clicking on the scroll bar buttons (scrolling by small steps) or clicking on the neutral parts of of the scroll bar or dragging the schroll bar with the mouse respectively (scrolling by large steps)
- Collapsing and expanding groups of hierarchy levels by clicking on the +/- symbols in the table.
- Modifying the table/diagram ratio by dragging the splitter bar with the mouse

Name	Duration			
		04	05	06
Node 1	5	+	+	
Node 2	5			

- Display of a tool tip, if the mouse remains for some time on an object of the chart. This requires the AJAX extensions to have been enabled and, on the side of the client, a browser where JavaScript was activated

The above interactions have been implemented in the VARCHART XGantt control, so the developer does not need to invest time or work into them.

The picture below shows a sample application, where names and durations of activities can be modified by text boxes, start dates can be entered by the calendar control and the time scale resolution can be modified by buttons.



Data that do not belong to the description of an activity but refer to the VARCHART XGantt control as a whole, such as the time scale resolution, can most easily be modified. The below sample code for zooming by buttons demonstrates this:

#### Example Code VB.NET

```
Private Sub ZoomInButton_Click(ByVal sender As System.Object, ByVal e_
As System.EventArgs) Handles ZoomInButton.Click
    Dim uw = VcGanttASP1.TimeScaleCollection.Active.Section(0).UnitWidth
    uw += 30
    VcGanttASP1.TimeScaleCollection.Active.Section(0).UnitWidth = uw
End Sub
```

```
Private Sub ZoomOutButton_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles ZoomOutButton.Click
    Dim uw = VcGanttASP1.TimeScaleCollection.Active.Section(0).UnitWidth
    uw -= 30
    If (uw < 30) Then
        uw = 30
    End If
    VcGanttASP1.TimeScaleCollection.Active.Section(0).UnitWidth = uw
End Sub
```

**Example Code C#**

```
private void ZoomInButton_Click(object sender, System.EventArgs e)
{
    int uw =
vcGanttASP1.TimeScaleCollection.Active.get_Section(0).UnitWidth;
    uw += 30;
    vcGanttASP1.TimeScaleCollection.Active.get_Section(0).UnitWidth = uw;
}

private void ZoomOutButton_Click(object sender, System.EventArgs e)
{
    int uw =
vcGanttASP1.TimeScaleCollection.Active.get_Section(0).UnitWidth;
    uw -= 30;
    if (uw < 30)
        uw = 30;
    vcGanttASP1.TimeScaleCollection.Active.get_Section(0).UnitWidth = uw;
}
```

Clicks by the right mouse button, double clicks and even mouse movements remain unnoticed. This results in the below list of disabled interactions:

- moving elements in the Gantt chart
- altering the time scale resolution by dragging a time scale section
- modifying the size ratio between the table and the Gantt graph or the histogram by moving splitter bars
- control by the key board, in-place editing of table data
- dialogs, context menus on clicking the right mouse button
- drag & drop

These restrictions can partly be compensated by using the standard web server controls provided by the .NET framework or by using own controls. Apart from buttons and text boxes there is also a calendar control to edit dates.

If data of single activities are to be edited, the user has to select an activity first. The selection is to be stored for the HTTP request following. Due to the absent memory of the HTTP protocol, the selection will not be known any more on the server, if the user - for example - modifies the name of the activity in a text box and thus triggers another HTTP request. A way to save the selection is to store it to a session variable:

**Example Code**

**VB.NET:**

## 38 Interactions in the VARCHART XGantt Web Server Control

```
Private Sub VcGanttASP1_VcNodeLeftClicking(ByVal sender As Object,
ByVal _ e As NETRONIC.XGantt.Web.VcNodeClickingEventArgs) Handles
VcGanttASP1.VcNodeLeftClicking
    Session.Remove("ClickedObject")
    Session.Add("ClickedObject", e.Node)
    If (e.Node.GetType().ToString() = "NETRONIC.XGantt.Web.VcNode") Then
        NameTextBox.Text = e.Node.DataField(1)
    End If
End Sub
```

```
Private Sub NameTextBox_TextChanged(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles NameTextBox.TextChanged
    Dim obj As NETRONIC.XGantt.Web.VcNode

    obj = Session.Item("ClickedObject")
    If (Not obj Is Nothing) Then
        obj.DataField(1) = NameTextBox.Text
        obj.Update()
    End If
End Sub
```

### Sample C#:

```
private void vcGanttASP1_VcNodeLeftClicking(object sender,
NETRONIC.XGantt.Web.VcNodeClickingEventArgs e)
{
    Session.Remove("ClickedObject");
    Session.Add("ClickedObject", obj);
    if (obj.GetType().ToString()=="NETRONIC.XGantt.Web.VcNode")
        NameTextBox.Text=(String)((VcNode)obj).get_DataField(1);
}

private void NameTextBox_TextChanged(object sender, System.EventArgs e)
{
    Object obj = Session["ClickedObject"];

    if (obj!=null)
        if (obj.GetType().ToString()=="NETRONIC.XGantt.Web.VcNode")
            {
                ((VcNode)obj).set_DataField(1, NameTextBox.Text);
                ((VcNode)obj).Update();
            }
}
}
```

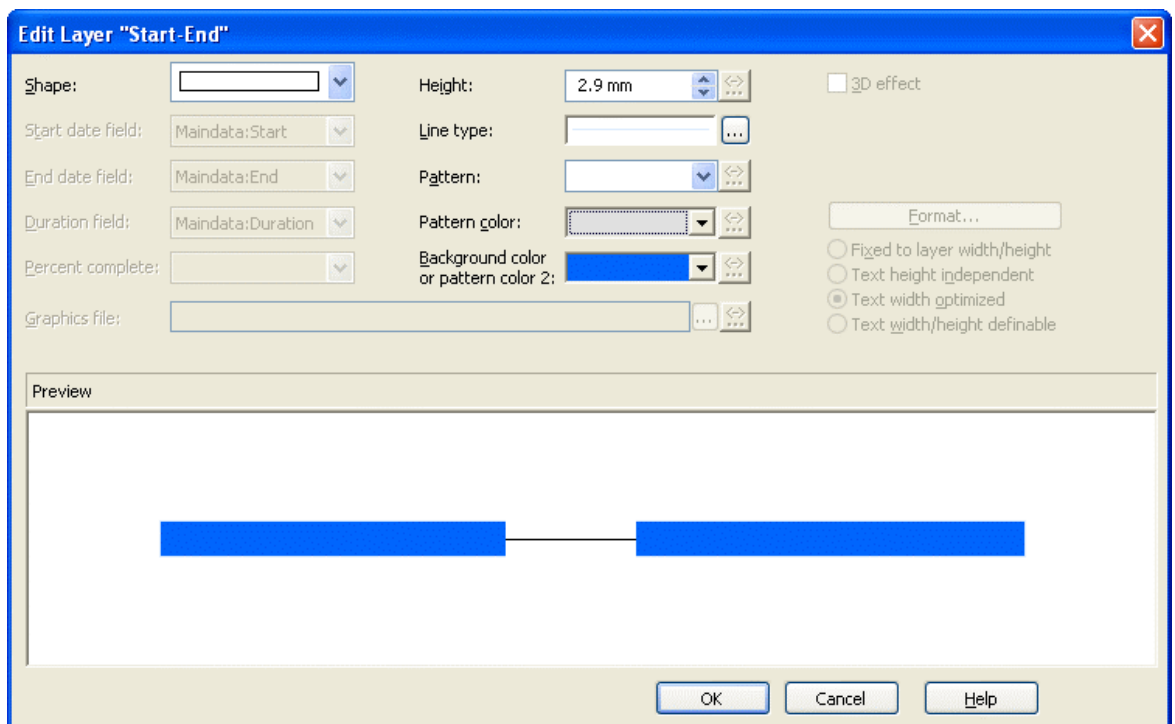
As an alternative, the VcNodeCollection object also offers a solution. It requires to identify the activity affected in the collection, since several activities may have been selected by the user.



## 2.7 Using Layers

A layer is the graphical representation of a pair of dates. In addition, the same pair of dates can be displayed by different layers. Logically, the different layers stack up to a pile.

In our example, we are now going to create a second, different looking layer.

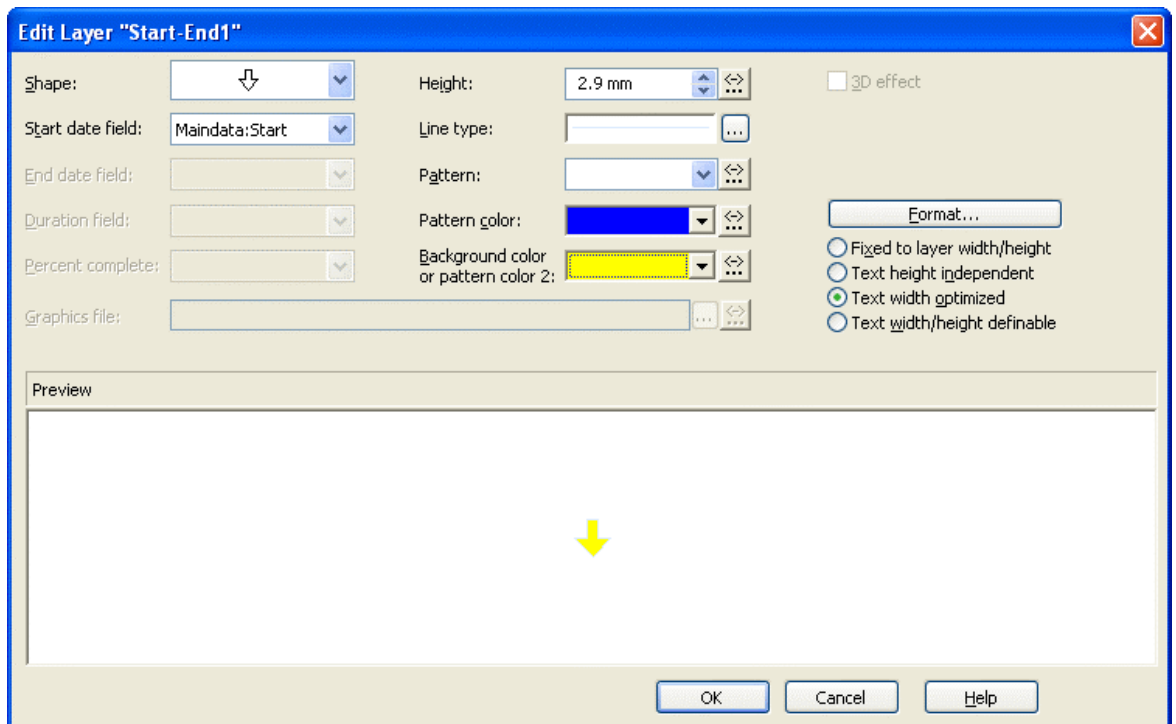
1. On the **Objects** property page, please select **Layers...**, which will pop up the dialog **Specify bar appearance**. Please note that the layer **Start-End** was already defined before.



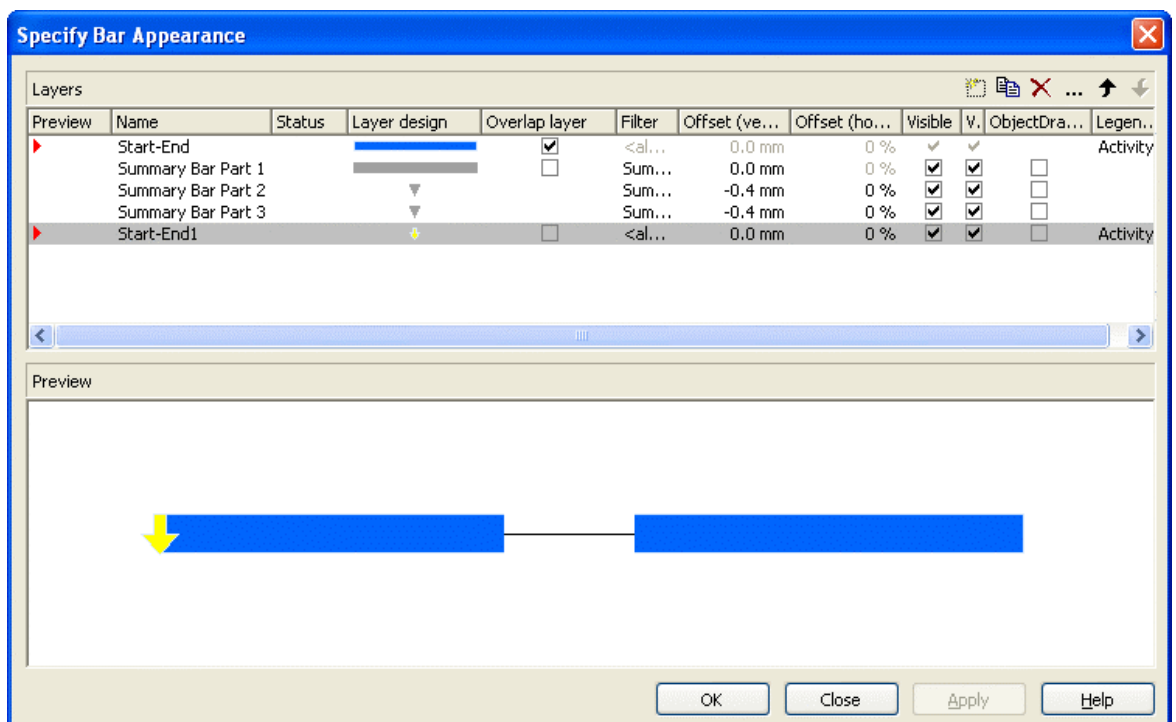
2. Please copy the definition of the layer **Start-End** by clicking on the **Copy layer** button .
3. You can now edit the new layer **Start-End1** by clicking on the button **Edit layer** .
4. Please change the **Background color** to yellow and the **Shape** to an arrow pointing downward.



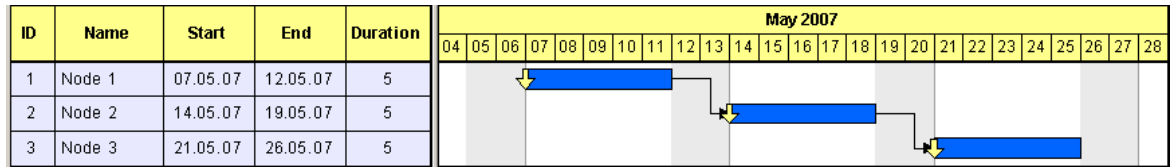
## 40 Using Layers



5. By clicking on OK, you will return to the dialog **Specify bar appearance**.
6. The layers will be displayed in the preview if marked the column **Preview**. The red triangle appearing indicates the display of the layer in the preview window below.



7. Our modification of the definition results in the below layer shape:




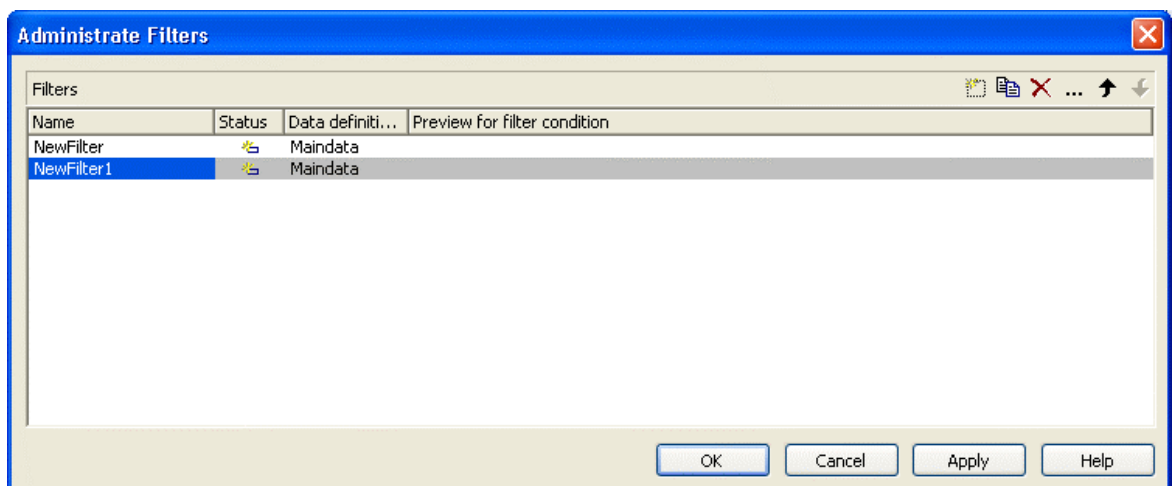
## 2.8 Using Filters


Next, we would like to have the red arrow appear only if the node is a milestone i.e., if the duration of the activity equals 0.

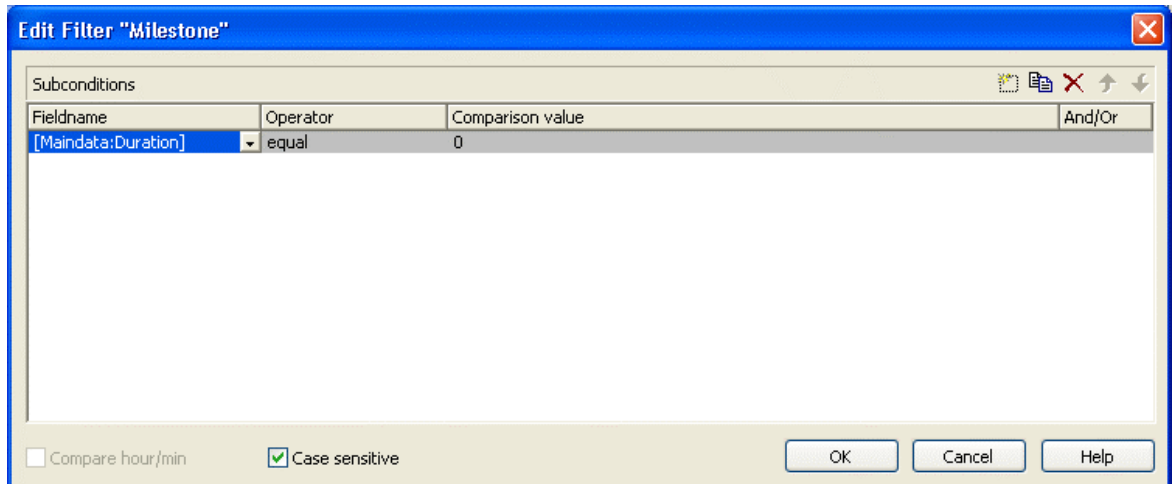
This problem can be solved easily by using filters. A filter consists of a series of linked conditions which result in a logical Yes/No statement.

Layers are always linked to filters. The corresponding layer becomes visible only if the evaluation of the filter conditions results in "Yes". The filter <always>, which is assigned to a layer by default, always returns "Yes". For our example, two filters are required that contain one condition each:

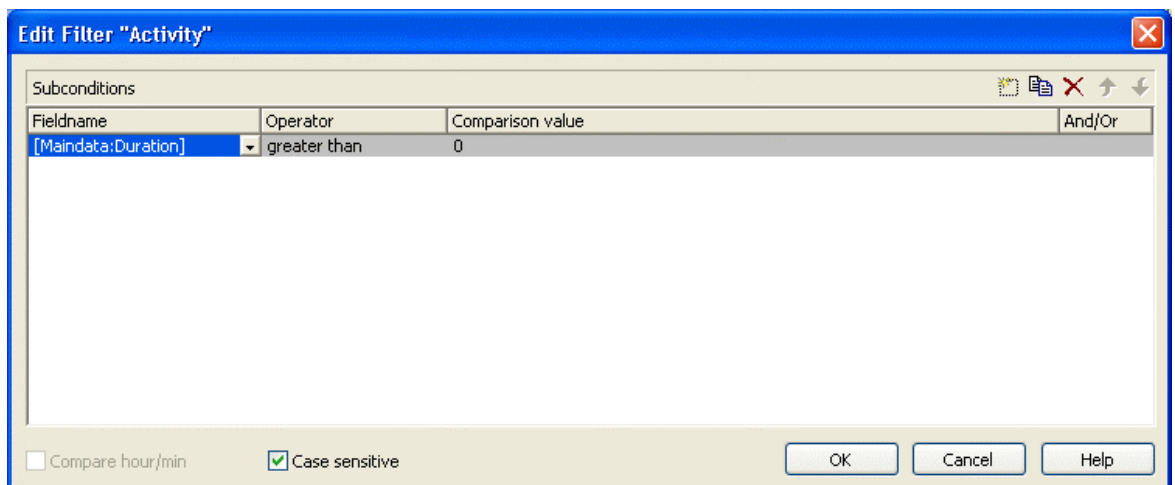
- The red arrow shall appear if the duration = 0
  - The blue bar shall appear if the duration > 0
1. On the property pages **Objects** please click on the button **Filters**, which will pop up the dialog **Administrare Filters**.
  2. Now please create two new filters by clicking on the button .



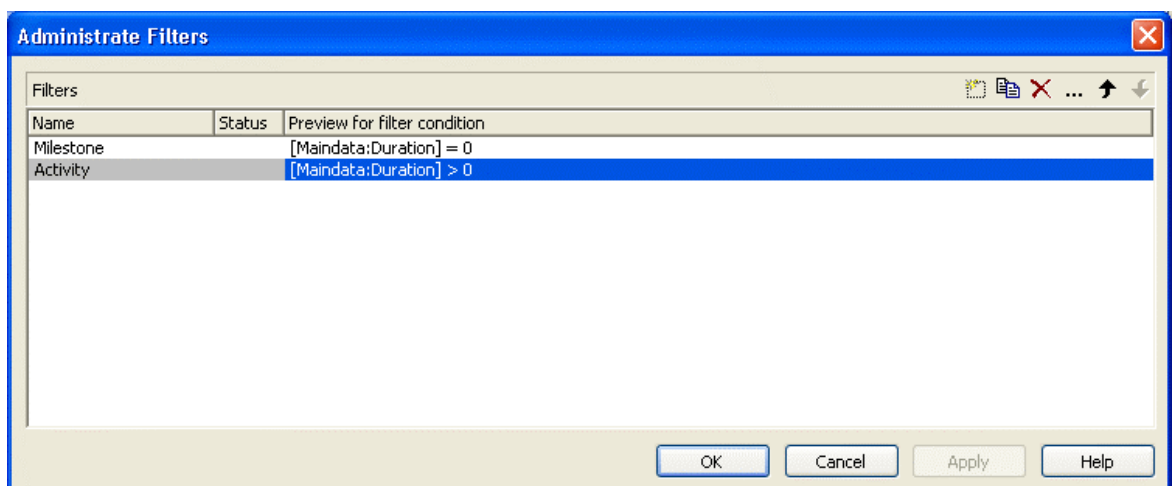
3. In the column **Name** rename "NewFilter" and "NewFilter1" into "Milestone" and "Activity".
4. Please confirm the modifications by clicking on **Apply**.
5. Select the filter "Milestone" and open the dialog **Edit Filter** by clicking on .
6. Select "Duration" as **Fieldname**, as **Operator** "equal" and as **Comparison value** 0.



7. Leave the dialog by clicking on **OK**.
8. Select "Activity" and by clicking **...** go again to the **Edit Filter** dialog.
9. Select "Duration" as **Fieldname**, for the **Operator** "greater than" and for the **Comparison value** 0.

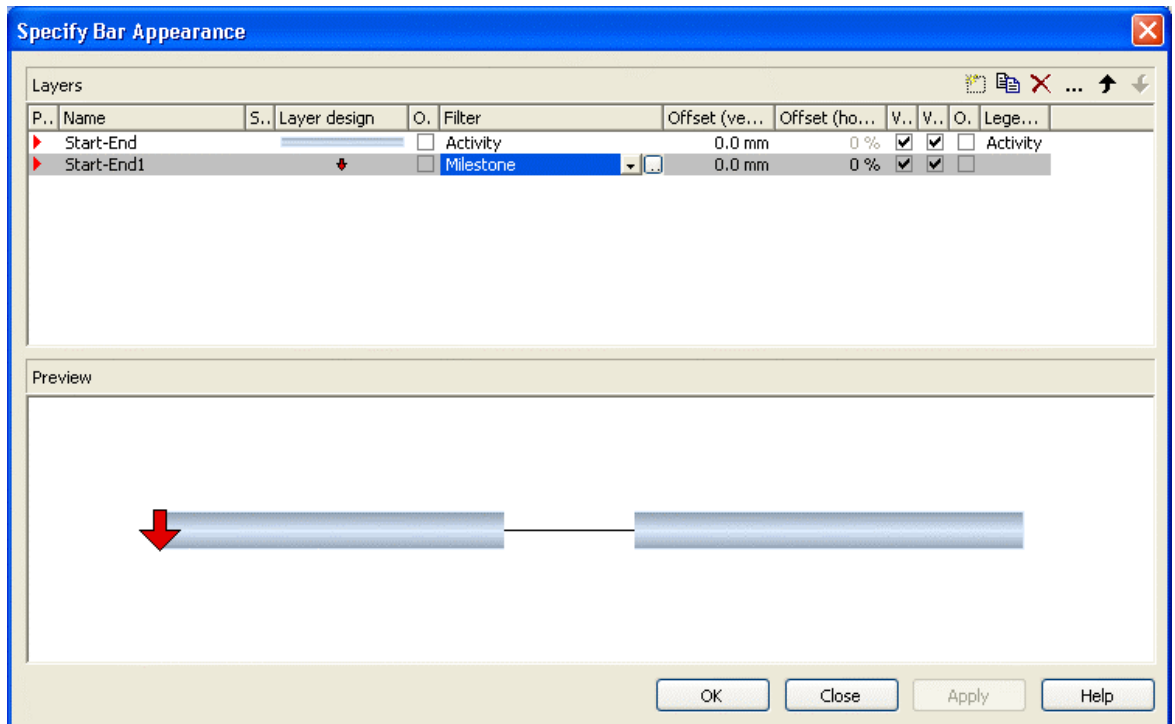


10. Leave the dialog by clicking on **OK**.

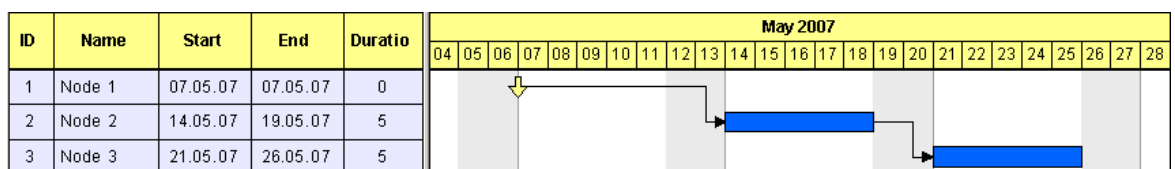


## 44 Using Filters

11. Click on **OK** again to return to the property pages.
12. 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**.



13. Please run the program now and set the duration of the first activity to 0. The result is shown in the below illustration:





## 46 Creating Histograms

**Step 1:** Displaying a histogram in the Gantt chart is switched on.

**Step 2:** Marked activities shall appear inverted in the table while in the Gantt graph, they shall be crosshatched. As a first partial step, their markebility is switched off for the Gantt graph.

**Step 3:** To distinguish between selected and non-selected nodes, a data field named "Selected" is created, that stores the actual selection state of a node.

**Step 4:** A value is assigned to the data field, that represents the marking state.

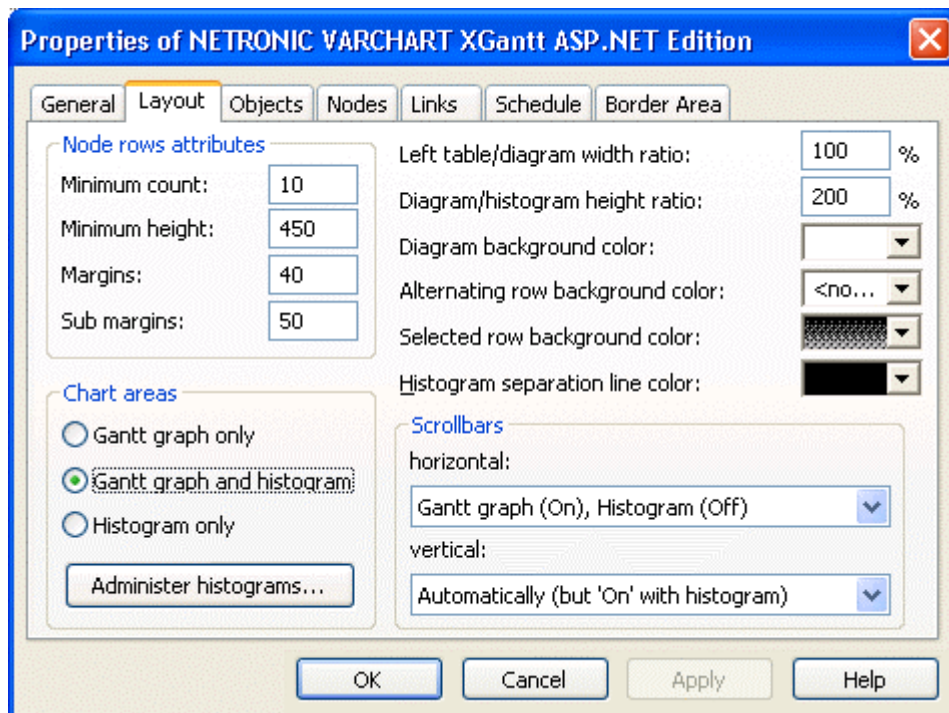
**Step 5:** Two different filters are created that separator selected and unselected activities.

**Step 6:** Two different appearances are defined for selected and unselected nodes. They are combined with the filters.

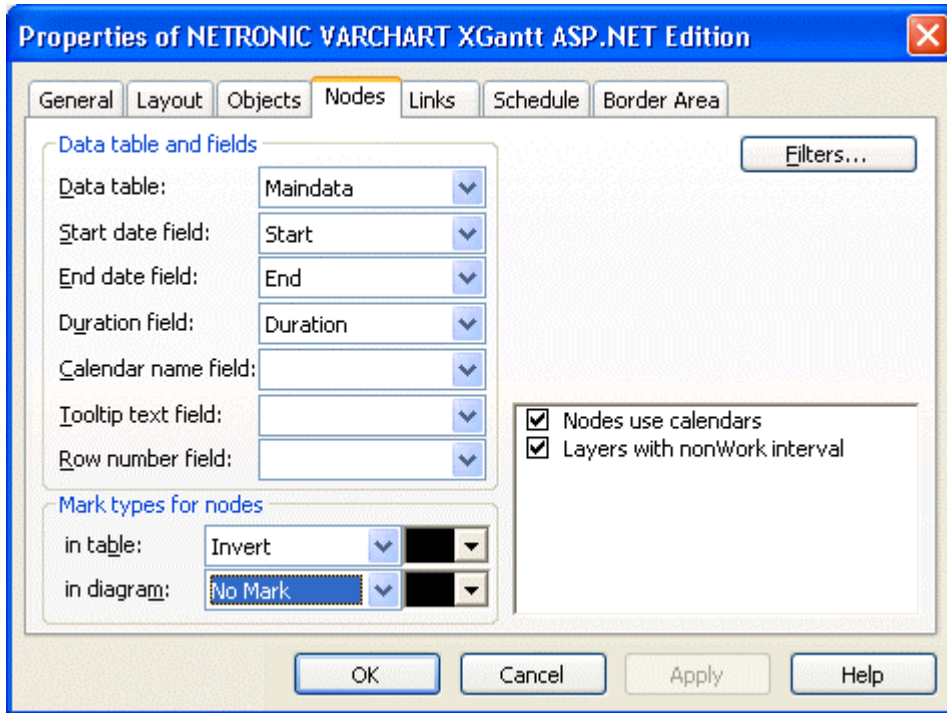
**Step 7:** Four curves are created for the histogram: the capacity curve, the curve of unmarked activities, the curve of marked activities and an auxiliary curve to fill an area. To the areas between the curves, colors and patterns are assigned.

**Step 8:** Finally, the values of the capacity curve are defined.

**Step 1:** First, please switch on the display of histograms in the Gantt diagram. Please invoke the property page **Layout** and find the tab section **Chart areas**, where you can set the option **Gantt graph and histogram**.



**Step 2:** Since marked nodes in the Gantt graph shall show a crosshatch pattern of their own, their markeability is switched off now for the Gantt graph. Please invoke the property page **Nodes**, find the tab section **Mark Types** and set the field **for nodes in diagram** to **No Mark**.



**Step 3:** To distinguish between selected and non-selected nodes, a data field named "Selected" is created, that stores the actual selection state of a node. Please invoke the dialog **Administrative Data Tables** by clicking **Data tables** on the property page **Objects** and select the Maindata data table. In the lower section, please add a field of the type **Integer** and name it "Selected". On the contents of this field it will depend later on what the node is going to look like when displayed.



## 48 Creating Histograms

**Administrate Data Tables**

Data Tables

Name	Status	Multiple primary keys allowed	Description
Maindata		<input type="checkbox"/>	
Relations		<input type="checkbox"/>	

Data Table Fields

Index	Name	Primary key	Type	Date format	Editable	Hidden	Relationship
0	ID	<input checked="" type="checkbox"/>	Integer		<input type="checkbox"/>	<input type="checkbox"/>	
1	Name	<input type="checkbox"/>	String		<input checked="" type="checkbox"/>	<input type="checkbox"/>	
2	Start	<input type="checkbox"/>	Date/Time	DD.MM.YYYY	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
3	End	<input type="checkbox"/>	Date/Time	DD.MM.YYYY	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
4	Duration	<input type="checkbox"/>	Integer		<input checked="" type="checkbox"/>	<input type="checkbox"/>	
5	Selected	<input type="checkbox"/>	Integer		<input checked="" type="checkbox"/>	<input type="checkbox"/>	

OK Cancel Apply Help

### Example Code VB.NET

```
Private Sub VcGantt1_VcNodesMarked(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.Web.VcNodesMarkedEventArgs) Handles
VcGantt1.VcNodesMarked
    Dim node As VcNode
    For Each node In VcGantt1.NodeCollection
        If node.Marked = True Then
            node.DataField(5) = 1
        Else
            node.DataField(5) = 0
        End If
        node.Update()
    Next
End Sub
```

**Example Code C#**

```
private void vcGantt1_VcNodesMarked(object sender,
NETRONIC.XGantt.Web.VcNodesMarkedEventArgs e)
{
    foreach (VcNode node in vcGantt1.NodeCollection)
    {
        if (node.Marked == true)
            node.set_DataField(5,1);
        else
            node.set_DataField(5,0);
        node.Update();
    }
}
```

In the event **VcNodeCreated** the below code prevents a node from appearing marked when created. Because all previously selected nodes will be unmarked when a new node is created, the field contents of "Selected" needs to be updated.

**Example Code VB.NET**

```
Private Sub VcGantt1_VcNodeCreated(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.Web.VcNodeCreatedEventArgs) Handles
VcGantt1.VcNodeCreated
    e.Node.DataField(1) = "Node " + e.Node.DataField(0)
    e.Node.Marked = False
    e.Node.Update()

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

**Example Code C#**

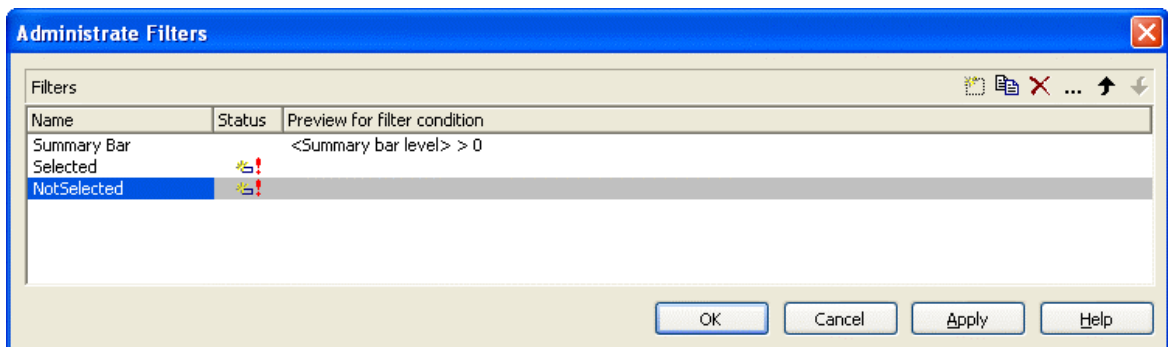
```
private void vcGantt1_VcNodeCreated(object sender,
NETRONIC.XGantt.Web.VcNodeCreatedEventArgs e)
{
    e.Node.set_DataField(1, "Node " + e.Node.get_DataField(0));
    e.Node.Marked = false;
    e.Node.Update();


    foreach (VcNode node in vcGantt1.NodeCollection)
    {
        node.set_DataField(5,0);
        node.Update();
    }
}
```

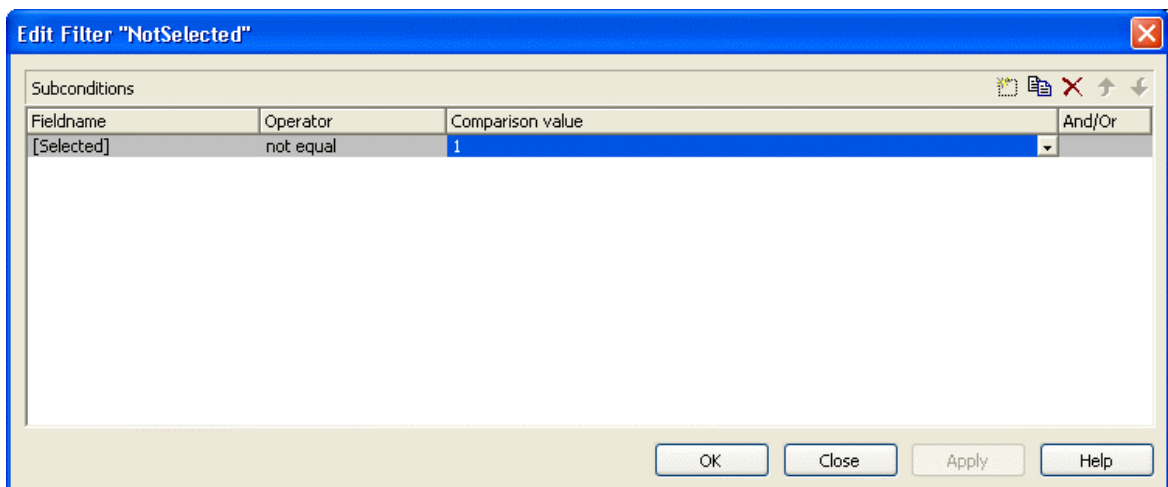
**Step 5:** In this step two different filters are created that separate selected from unselected activities. Please invoke the property page **Objects** click on the button **Filter...** to get to the dialog **Administrative Filters**. Please create two

## 50 Creating Histograms

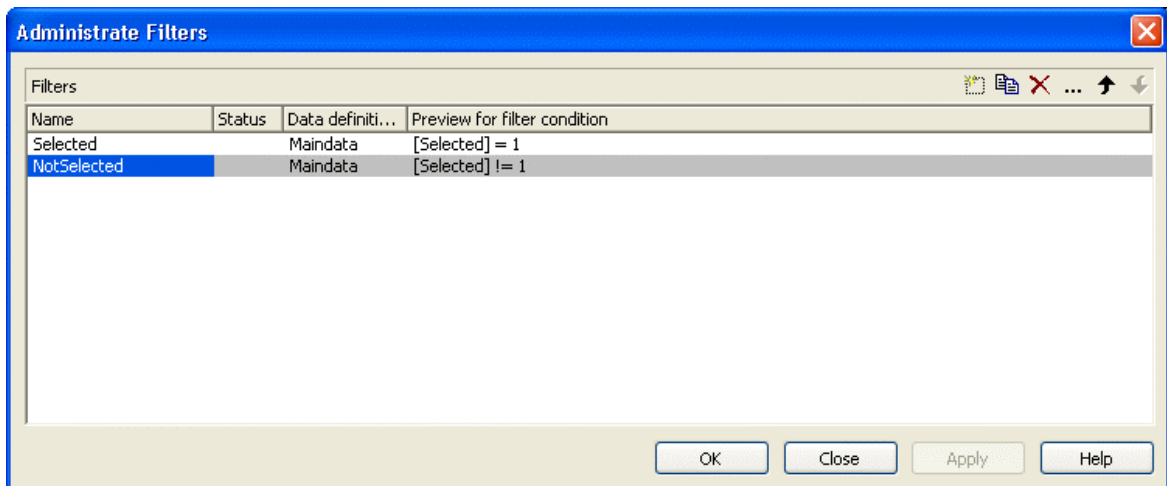
new filters by clicking on the button  and name them "Selected" and "NotSelected".




Now, please set the filter conditions. To the filter "Not Selected", please assign the condition "Selected not equal 1". Due to this condition, only unselected nodes will be filtered. Now please mark the filter **Not Selected** and click on the  button right-hand at the top of the dialog. It will invoke the **Edit Filter** dialog. In the column **Fieldname** please choose the field **Selected**, in the column **Operator** please choose **not equal** and in the column **Comparison value** please enter the value 1. Quit the dialog by **OK**.

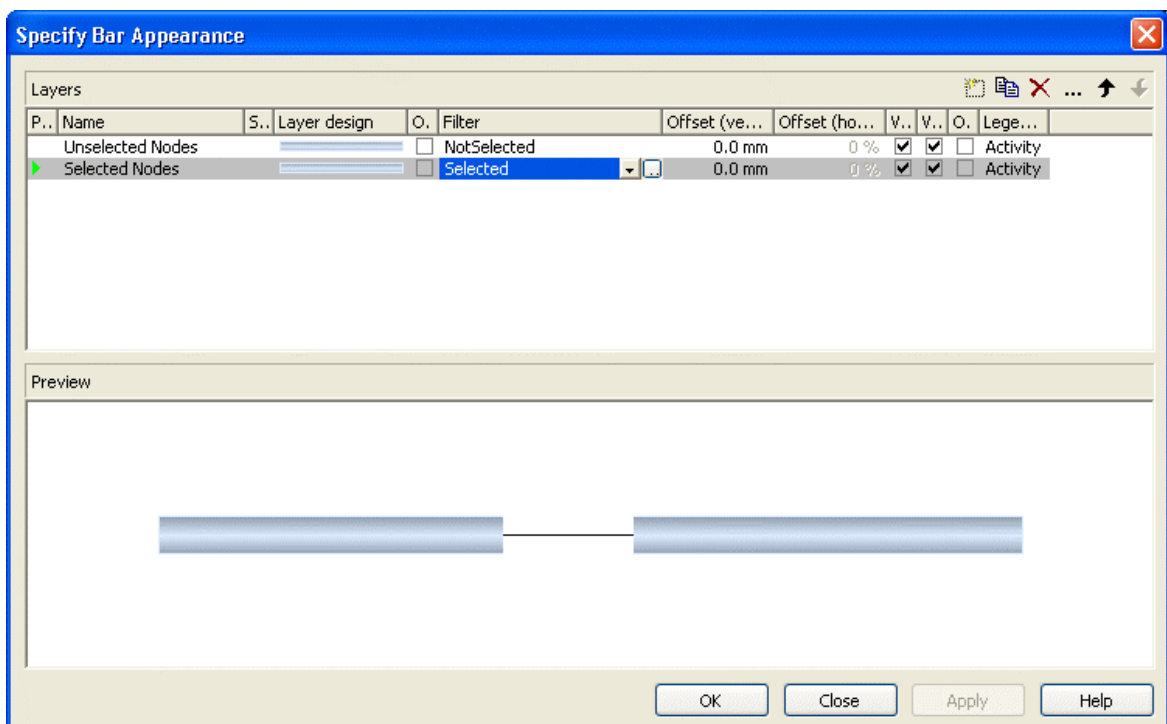


Now, in the same way please assign to the filter "Selected" the condition "Selected equal 1".



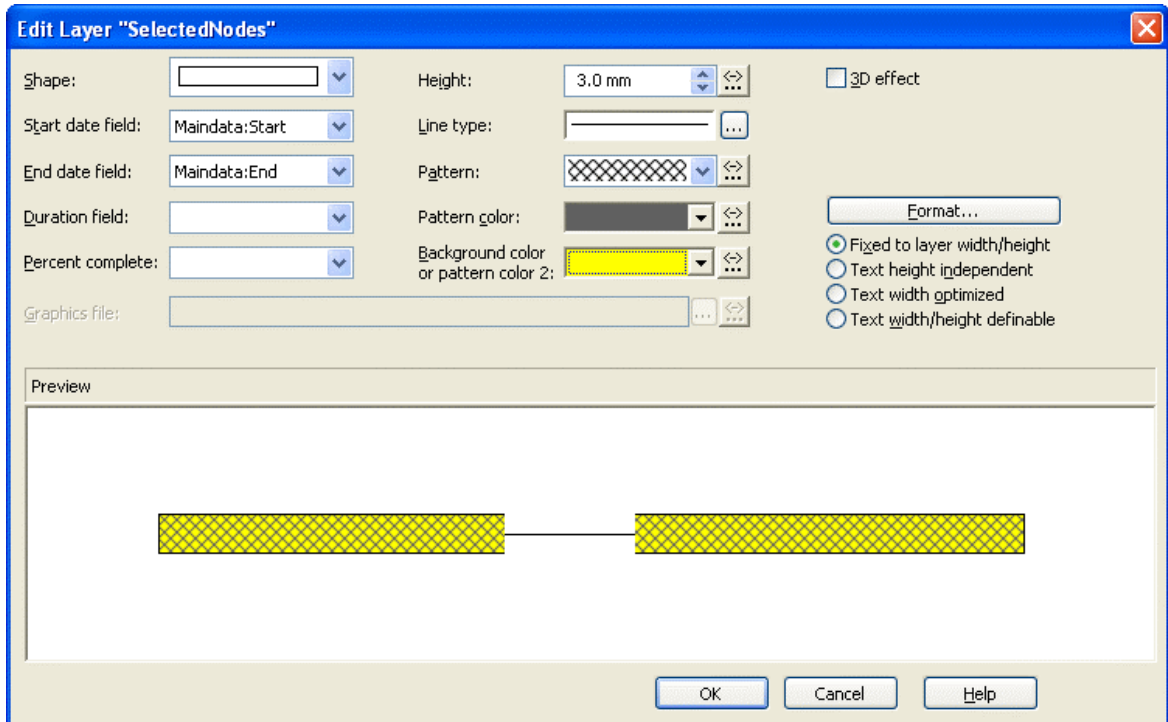
**Step 6:** In this step, we will define two different appearances for selected and unselected nodes to be combined with the filters.

Please select the property page **Objects** and click on the object **Layers...**. This will pop up the dialog **Specify Bar Appearance**. Please rename the layer "Start-End" into "Unselected Nodes" by entering the new name directly into the field in the column **Name**. Please find the column **Filter** and assign the filter "Not Selected" to the Layer. Copy the layer by clicking on the button  and name the copy "Selected Nodes". Assign the Filter "Selected" to the layer.

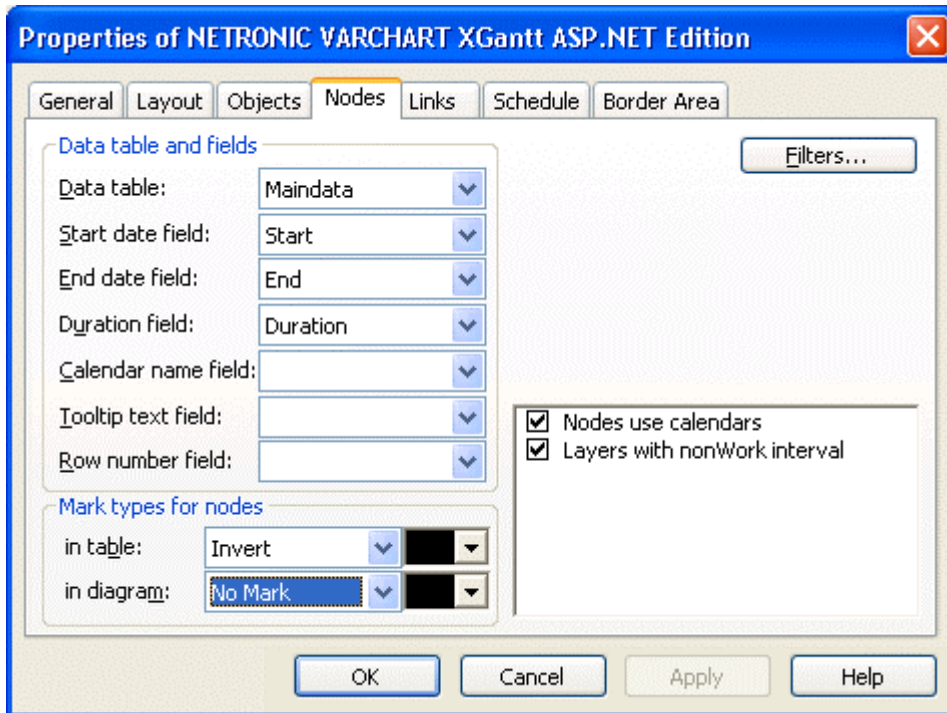


## 52 Creating Histograms

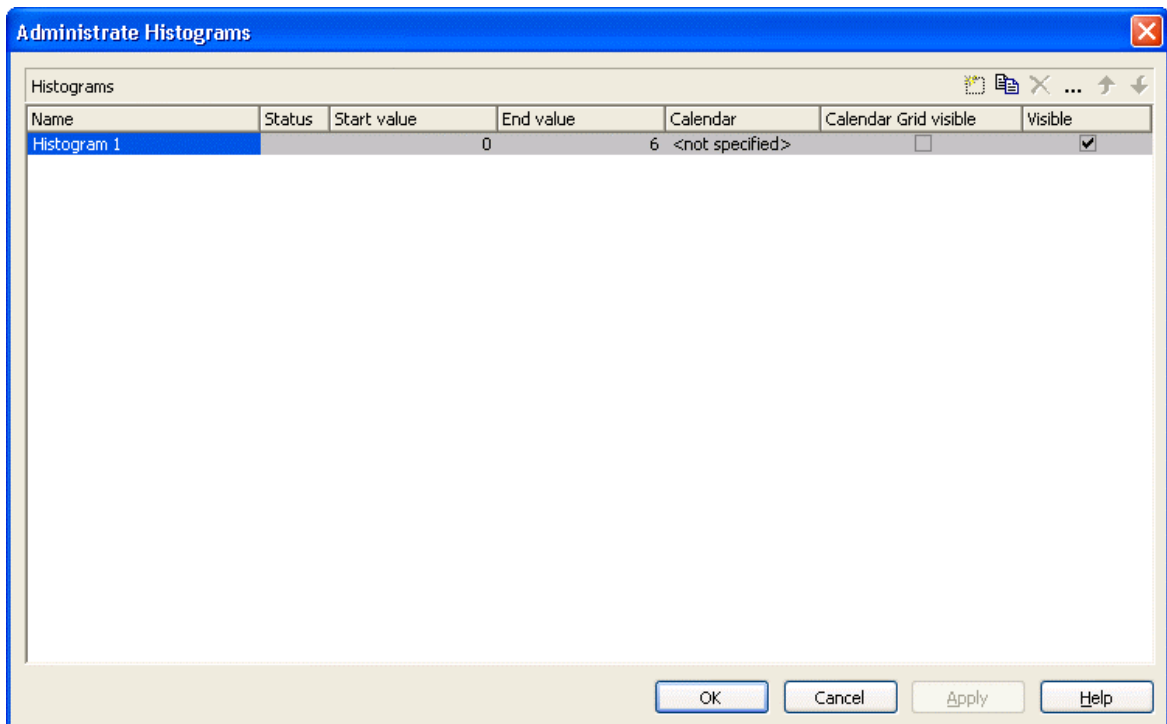
Both layers still look alike. You can modify the design of the layer "Selected Nodes" by double-clicking in the corresponding field of the column **Layer design**. The dialog **Edit-Layer** will pop up. Please select a cross hatch **Pattern**, a yellow **Background color** and a black **Pattern color**.



To ensure that weekends in non work intervals are indicated by a line instead of a bar, one more option needs to be set on the **Nodes** property page. There please activate the option **Layers with nonWork interval**.



**Step 7:** In this step, four curves will be created for the histogram: the capacity curve, the curve of unmarked activities, the curve of marked activities and an auxiliary curve to fill an area. Click on **Administer histograms...** on the property page **Layout** to invoke the corresponding dialog.




Several histograms may be present in a Gantt chart at the same time. Each of

## 54 Creating Histograms

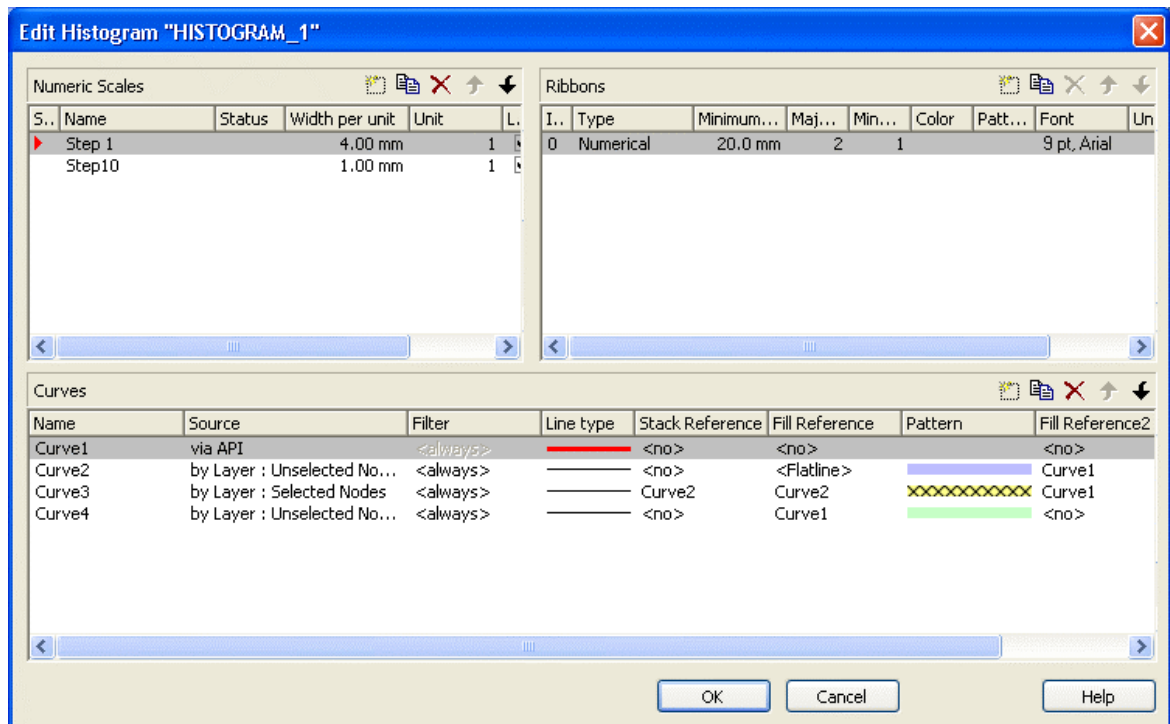
the histograms has a numeric scale of its own and contains its own curves.

Please now define the start and end values of the numeric scale. Click in the **Histogram\_1** field of the **End value** column and enter 6.

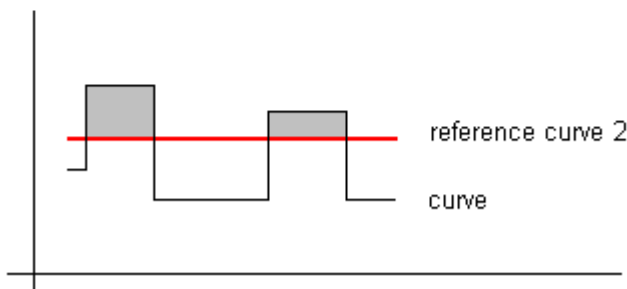
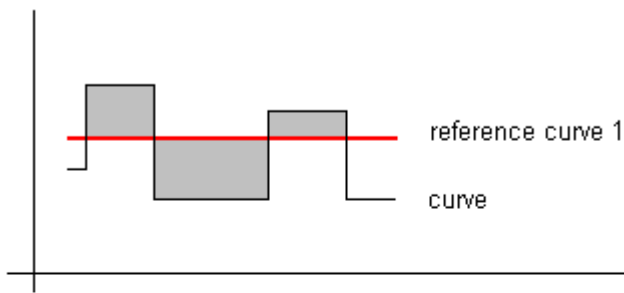
Now, please edit the histogram. For this, please click on the **Edit** button  right-hand at the top of the dialog.

"Curve 1" shall represent the capacity curve (in red). "Curve 2" shall summarize the marked nodes while "Curve 3" shall represent the unmarked nodes. "Curve 4" is an auxiliary curve to provide the green background of the shortfall areas.

One curve already exists. Please create three further curves and define their properties according to the illustration.



To a curve, two reference curves can be assigned at maximum. A curve forms areas with its reference curves, to which colors and patterns can be assigned (see sketches below). Of the first reference curve, all parts that form above and below the curve add up to the area (top sketch). Of the second reference curve, only those parts add to the area that form below the curve, that is, of which the Y-values are smaller than those of the original curve (bottom sketch). In addition, areas formed by the second reference curve are displayed at higher priority. We will see below, what the consequences will be in the histogram.



The capacity curve (Curve 1) will receive its values from a list, that we are going to provide later on by programming code. Therefore, please set its data source in the field **Source** to **via API**. Because of this, an additional filter for nodes from which data can be taken is not needed.

Please set the **Line type** to a thick red line. The values of this curve shall not be added to the values of another curve; therefore the field **Stack reference** remains empty. Also, the capacity curve is not intended to form an area with another curve, therefore the two fill references and their fill patterns remain empty. Please create Curve 1 as described by clicking on the corresponding fields in the dialog.

Curve 2 represents the nodes not selected and composes of values from the layers named "Unselected Nodes". A filter for further selection is not needed. Please choose a blue line color for the curve line. The curve values will not be added to the values of another curve, so the **Stack Reference** remains empty. The curve is supposed to form an area with the X-axis, so please in the field **Fill Reference** select the value **Flatline**.

This curve, consisting of the non-selected nodes, should also indicate in a special way, where it exceeds the capacity curve in order to mark the bottlenecks of the production system. Therefore, as soon as its Y-values exceed those of Curve 1, the area below shall be hatched. So please set Curve 1 as its second reference curve and select a hatched fill pattern.

"Curve 3" shall represent the selected nodes. So please, as its data source, assign the layers named "Selected Nodes". A filter is not needed. Please assign a light gray line color. Since the selected nodes shall be displayed



above the non-selected nodes, their values have to be added to the ones of the non-selected nodes. So please choose Curve 2 as the **Stack Reference**. The same curve also serves as the first reference curve, since the selected nodes visually shall differ from the non-selected ones. As a fill pattern, please select a gray cross hatch pattern on a yellow background.

The area formed will be visible above and below Curve 2. In addition, it shall appear above the capacity curve; therefore please assign Curve 1 as the second reference curve and fill the area with the same color and pattern. If selected nodes rise above the capacity curve, they will appear in the same color and pattern as below the capacity curve (you could distinguish between selected nodes above and below the capacity curve by assigning e.g. a red color here).

By curve 4 we are going to define an area that represents the light green background between the capacity curve and the node piles below. It indicates available resources of the production system. It is limited at its bottom by the unselected nodes, so please choose them as the data source. At the top, the area is limited by the capacity curve, which you please set as the first reference curve.

Question: Why does the area of Curve 4 not hide the selected nodes?  
 Answer: Because there is a priority in the list of curves presented by this dialog. The curves listed at the bottom have a lower priority than those listed at the top. This is why areas of curve 3 are displayed on top of areas of curve 4. You can modify the priority by the arrows right-hand at the top of the window.

### Step 8:

In the final step, we are going to provide the values of the capacity curve. For this, please modify the code in the **Load** event as shown below:

#### Example Code VB.NET

```
Private Sub Page_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
    VcGantt1.Width = ClientSize.Width - VcGantt1.Left
    VcGantt1.Height = ClientSize.Height - VcGantt1.Top

    VcGantt1.InsertNodeRecord("1;Node 1;07.05.2007;;5")
    VcGantt1.InsertNodeRecord("2;Node 2;09.05.2007;;5")
    VcGantt1.InsertNodeRecord("3;Node 3;10.05.2007;;6")
    VcGantt1.InsertNodeRecord("4;Node 4;17.05.2007;;10")
    VcGantt1.InsertNodeRecord("5;Node 5;22.05.2007;;3")
    VcGantt1.InsertNodeRecord("6;Node 6;23.05.2007;;1")

    VcGantt1.EndLoading()
```

```

VcGantt1.OptimizeTimeScaleStartEnd(3)

'calculate end date
Dim node As VcNode

For Each node In VcGantt1.NodeCollection
    setNodeEndDate(node)
Next

Dim histogram As VcHistogram
Dim curve As VcCurve

histogram = VcGantt1.HistogramCollection.FirstHistogram
curve = histogram.CurveCollection.CurveByName(" Curve1 ")
curve.PointsEquidistant = False
curve.SetValues("01.05.2007", "2")
curve.SetValues("05.05.2007", "0")
curve.SetValues("07.05.2007", "2")
curve.SetValues("12.05.2007", "0")
curve.SetValues("14.05.2007", "4")
curve.SetValues("19.05.2007", "0")
curve.SetValues("21.05.2007", "2")
curve.SetValues("26.05.2007", "0")
curve.SetValues("28.05.2007", "2")

End Sub

```

### Example Code C#

```

private void Page_Load(object sender, System.EventArgs e)
{
    vcGantt1.Width = ClientSize.Width - vcGantt1.Left;
    vcGantt1.Height = ClientSize.Height - vcGantt1.Top;

    vcGantt1.InsertNodeRecord("1;Node 1;07.05.2007;;5");
    vcGantt1.InsertNodeRecord("2;Node 2;09.05.2007;;5");
    vcGantt1.InsertNodeRecord("3;Node 3;10.05.2007;;6");
    vcGantt1.InsertNodeRecord("4;Node 4;17.05.2007;;10");
    vcGantt1.InsertNodeRecord("5;Node 5;22.05.2007;;3");
    vcGantt1.InsertNodeRecord("6;Node 6;23.05.2007;;1");

    vcGantt1.EndLoading();

    vcGantt1.OptimizeTimeScaleStartEnd(3);

    // calculate end date
    foreach (VcNode node in vcGantt1.NodeCollection)
    {
        SetNodeEndDate(node);
    }

    VcHistogram histogram =
    vcGantt1.HistogramCollection.FirstHistogram();
    VcCurve curve = histogram.CurveCollection.CurveByName("Curve 1");
    curve.PointsEquidistant = false;
    curve.SetValues(Convert.ToDateTime("01.05.2007"), "2");
    curve.SetValues(Convert.ToDateTime("05.05.2007"), "0");
    curve.SetValues(Convert.ToDateTime("07.05.2007"), "2");
    curve.SetValues(Convert.ToDateTime("12.05.2007"), "0");
    curve.SetValues(Convert.ToDateTime("14.05.2007"), "4");
}

```

## 58 Creating Histograms

```
curve.SetValues(Convert.ToDateTime("19.05.2007"), "0");  
curve.SetValues(Convert.ToDateTime("21.05.2007"), "2");  
curve.SetValues(Convert.ToDateTime("26.05.2007"), "0");  
curve.SetValues(Convert.ToDateTime("28.05.2007"), "2");  
}
```

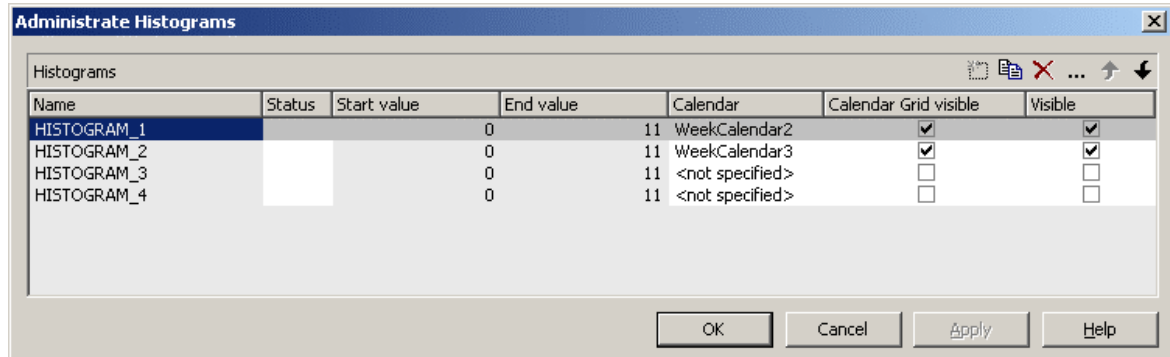
Now, please run the program and mark an activity. You can recognize immediately by the cross hatched section on a yellow background in the histogram, what part the activity occupies in the bulk of the work load displayed.

### Calendar Grids in Histograms

You can assign one or 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*

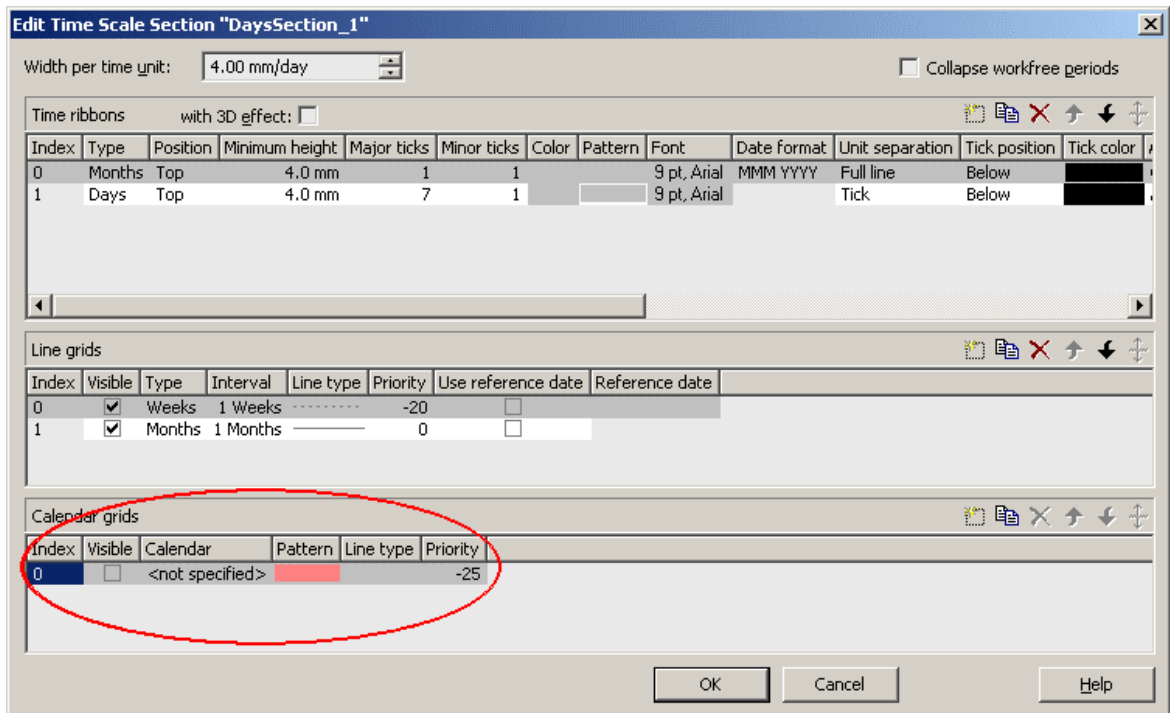
The corresponding API calls are:

#### Example Code VB.NET

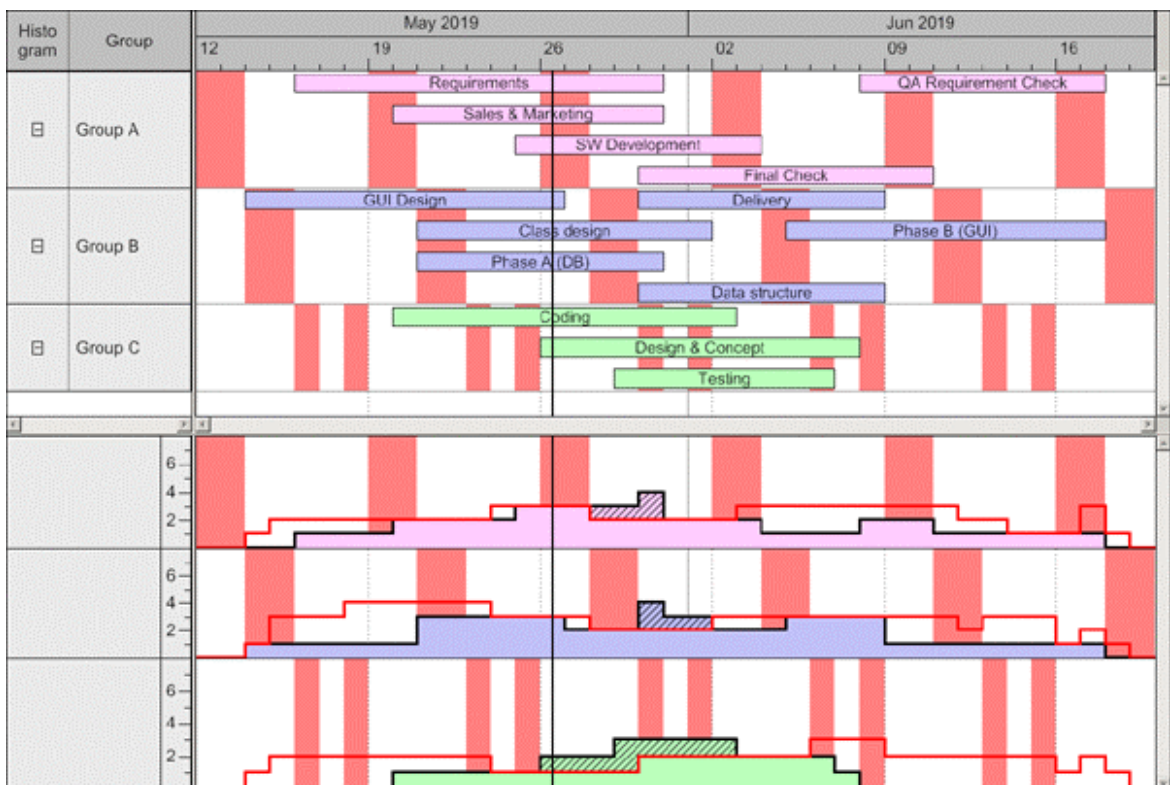
```
// assigning the calendar to the histogram (by the calendar name)  
histogram.calendarName = group.DataField(14)  
// switching the calendar grid on  
histogram.ShowCalendarGrids = True  
// setting the histogram visible  
histogram.Visible = True
```

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:

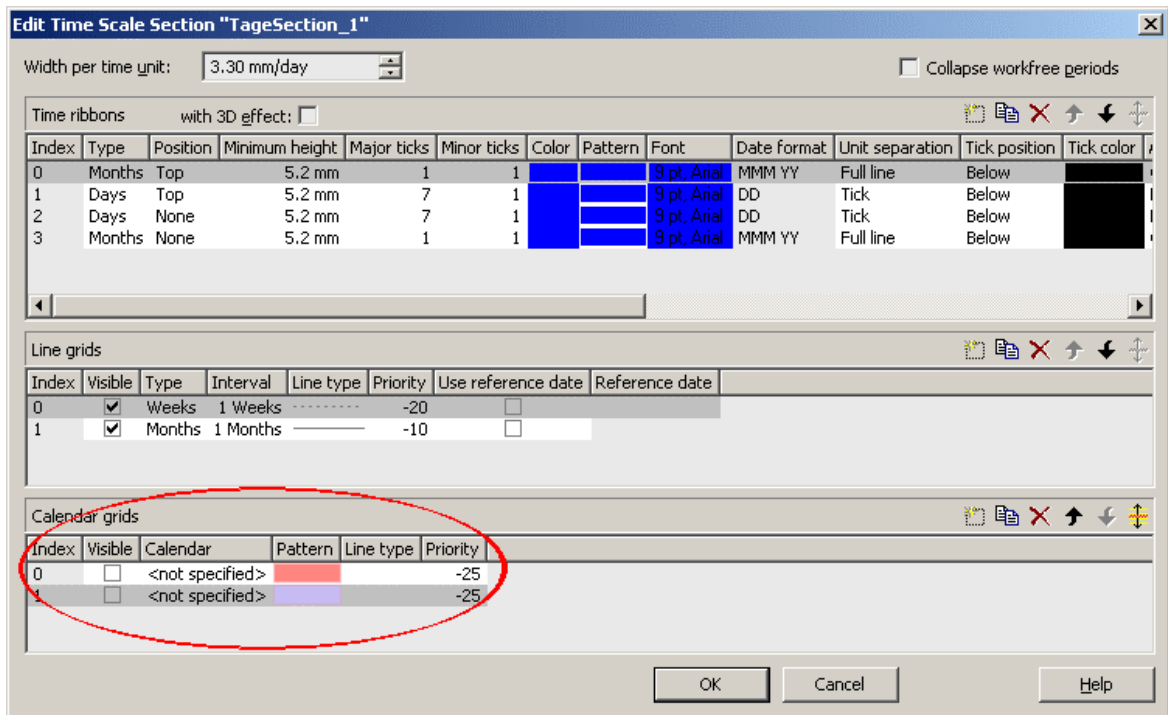


Thus the calendar grid will display the same appearance in the Gantt graph as in the histogram. In the example below it is a calendar grid that shows a different pattern for each group (groupwise calendar grid):

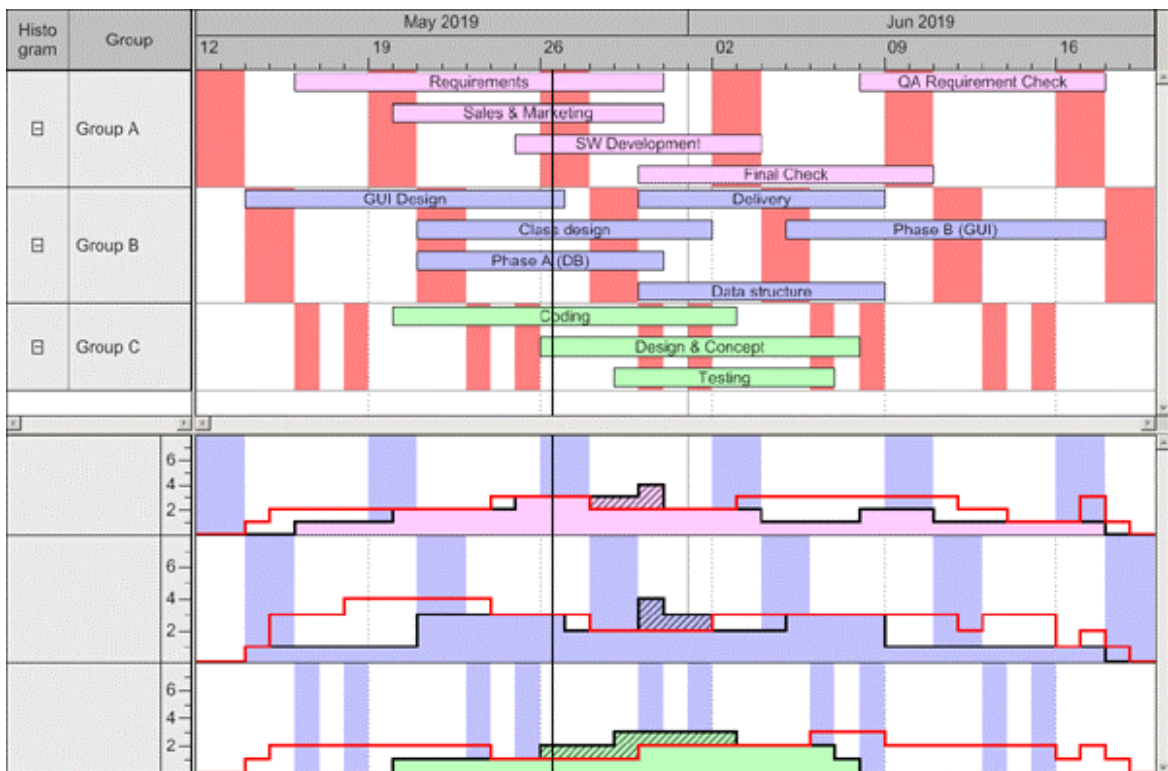


## 60 Creating Histograms

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:



---

## 2.10 Exporting a Diagram

You can export a diagram into a graphics file by using the API method **ExportGraphicsToFile**.

*Available formats:*

- \*.BMP (Microsoft Windows Bitmap)
- \*.EMF (Enhanced Metafile)
- \*.EMF, with EMF+ included
- \*.GIF (Graphics Interchange Format)
- \*.JPG (Joint Photographic Experts Group)
- \*.PNG (Portable Network Graphics)
- \*.TIF (Tagged Image File Format)
- \*.WMF (Microsoft Windows Metafile)
- \*.WMF, with EMF included

Only WMF and EMF are vector formats that allow to store files independent of pixel resolution. All other formats are pixel-oriented and confined to a limited resolution.

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

---

## 2.11 Saving the Configuration

All settings made on the property pages at design time are added to your project as HTML code. Changes come into operation only after saving your project.

**Tip:**

For this reason, you should activate in Microsoft Visual Studio .NET 2003 the Option **Save all changes in Tools Options Environment Projects and Solutions**, so that your settings are automatically saved before compiling.

If you do not select this option, you will have to save your project manually if you want the settings of the property pages to be used in the program.

You can store the settings of the property pages to a configuration outside your project at any time and load them when needed. This is very useful if you want to use previous settings again or if you need the same settings for different projects.

A stored configuration consists of two files of identical names but different extensions, (INI and IFD), that both are indispensable.

**How to save your current configuration:**

On the **General** property page please click on the **Export...** button and enter the name of the INI file. The ifd-file of the same name will be created automatically.

**How to load a stored configuration:**

On the **General** property page please click on the **Import...** button and select the desired file.

---



---

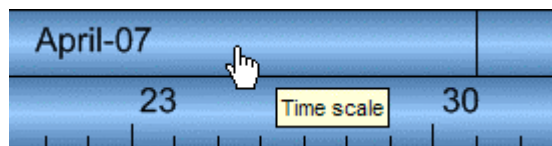
## 3 Important Concepts

---

### 3.1 AJAX Extensions

The VARCHART XGantt ASP.NET Edition contains a new DLL file, **NETRONIC.Web.dll**, which provides all components of the AJAX extensions. After having enabled the extensions, the following features are available for your Gantt chart:


- You can make tooltips available in a very easy way. The tool tips can contain any kind of element formatable in HTML, such as pictures, tables and others. All HTML font formats can be displayed, such as bold or colors.



- After interactions in the Gantt chart only parts of the web page, namely the chart and other sections which you may define, will be updated. Thus there will be neither flicker nor change of scroll position which would occur on re-loading a page.

The sections of the web page defined by you serve for updating other controls, for example a calendar, with ongoing interactions.

For this, you need the new **VcUpdatePanel** control included in the **NETRONIC.Web.dll**. It works as a "shell control" and is invisible to a user on the web page.

To use the **VcUpdatePanel** click on the according icon  in your tool box and draw a frame at the position in the form where you want it to appear. Afterwards, the reference list of the project. automatically shows a reference to the **NETRONIC.Web.dll**.

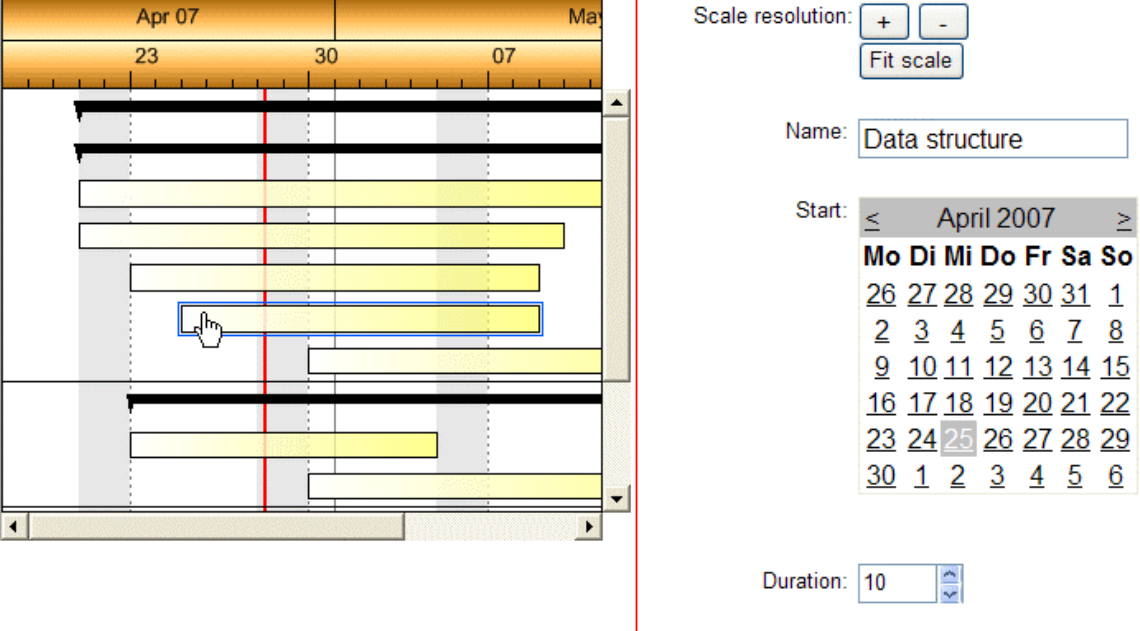
Now drag the control to be updated synchronously with interactions of the **VcGantt** control into the empty frame. You can also drag more than one control into a **VcUpdatePanel**.

The latter invokes graphical updates, visualizing the data that are present on the server also on the client (browser). You **must not** drag a **VcGantt** control onto a **VcUpdatePanel**,



## 64 Important Concepts: AJAX Extensions

The picture below shows an update panel on the right into which a calendar was drawn, the dates of which are synchronized with the dates of the node on which the pointer is positioned.



The image shows a Gantt chart interface with a task bar selected. A calendar popup is displayed on the right, showing the dates of the selected node. The calendar is for April 2007, with the 25th highlighted. The popup includes a 'Scale resolution' section with '+' and '-' buttons and a 'Fit scale' button. The 'Name' field contains 'Data structure'. The 'Start' field shows '< April 2007 >'. The calendar grid shows the following dates:

Mo	Di	Mi	Do	Fr	Sa	So
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	1	2	3	4	5	6

The 'Duration' field is set to 10.

You have to activate the AJAX extensions as well as the use of the tool tips. This may be done either on the property page **General** by ticking the box **Ajax extension** and if necessary **VcToolTipTextSupplying events** or by the properties **Ajax extensions enabled** and **ToolTipTextSupplyingEvent-Enabled**. Accordingly, the event **VcToolTipTextSupplying** will be triggered each time the pointer remains on an object for a certain time.

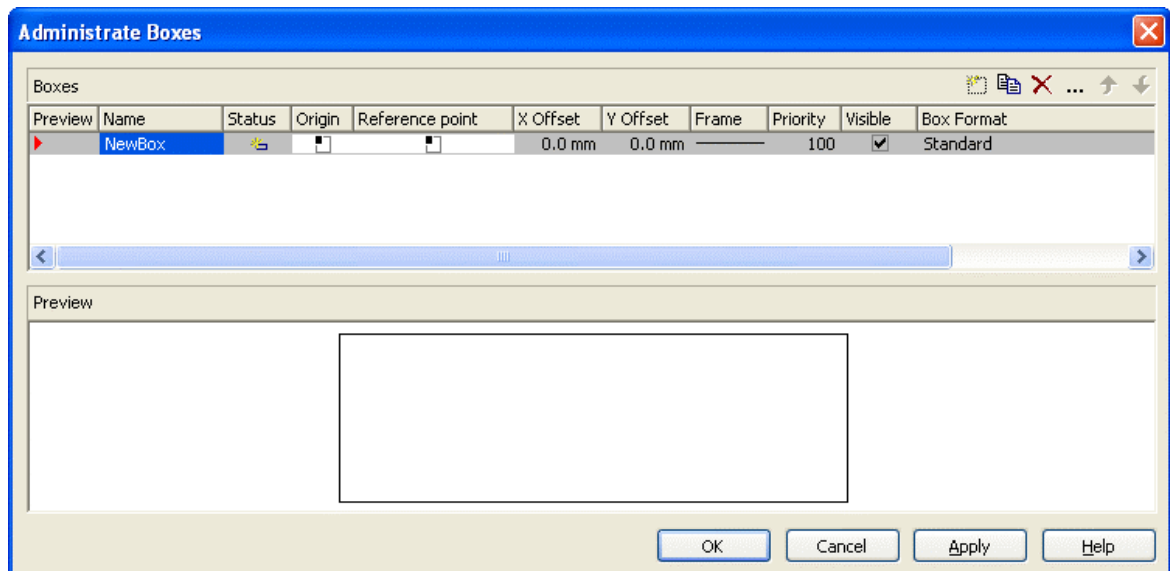
The AJAX extension are switched on by default, the tooltips are not.

### Tips:

- Both properties have to be set before the first loading of a page.
- The AJAX extensions have to be activated in order to use the tool tips.
- You should refrain from using "SmartNavigation" simultaneously with AJAX extensions, since there are incompatibilities.

## 3.2 Boxes

In the diagram area, boxes that contain texts or graphics can be displayed. To generate boxes, please select the property page **Objects** and press the **Boxes...** button. The dialog **Administrate Boxes** will open, where you can add, copy, delete or edit boxes.



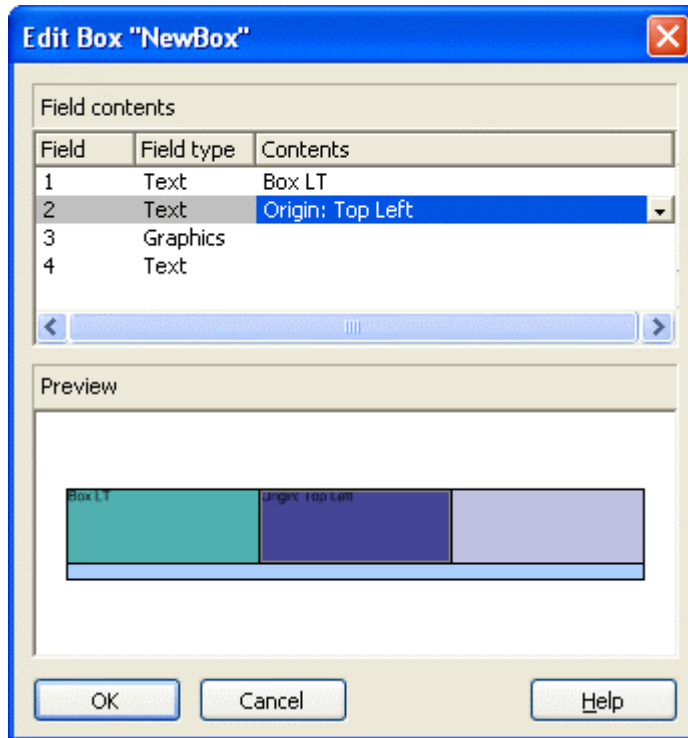
The properties **Origin**, **Reference Point**, **X Offset** and **Y Offset** allow to exactly position a box in the diagram area. The relative position of the boxes does not depend on the current diagram size.

For each box you can specify

- its name
- its point of origin (a point in the diagram to which the reference point refers to form what is called "the offset")
- its reference point, i. e. the complementary point of the box to form the offset
- its X or Y Offset (distance between origin and reference point in x or y direction)
- type, thickness and color of the box frame line
- its priority in comparison to other diagram objects (nodes, grids, etc.)
- whether the box is visible
- its format

The **Edit Box** dialog lets you edit a box. This dialog box will appear at design time when you click on the **Edit box** button in the **Administer Boxes** dialog box.

## 66 Important Concepts: Boxes



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

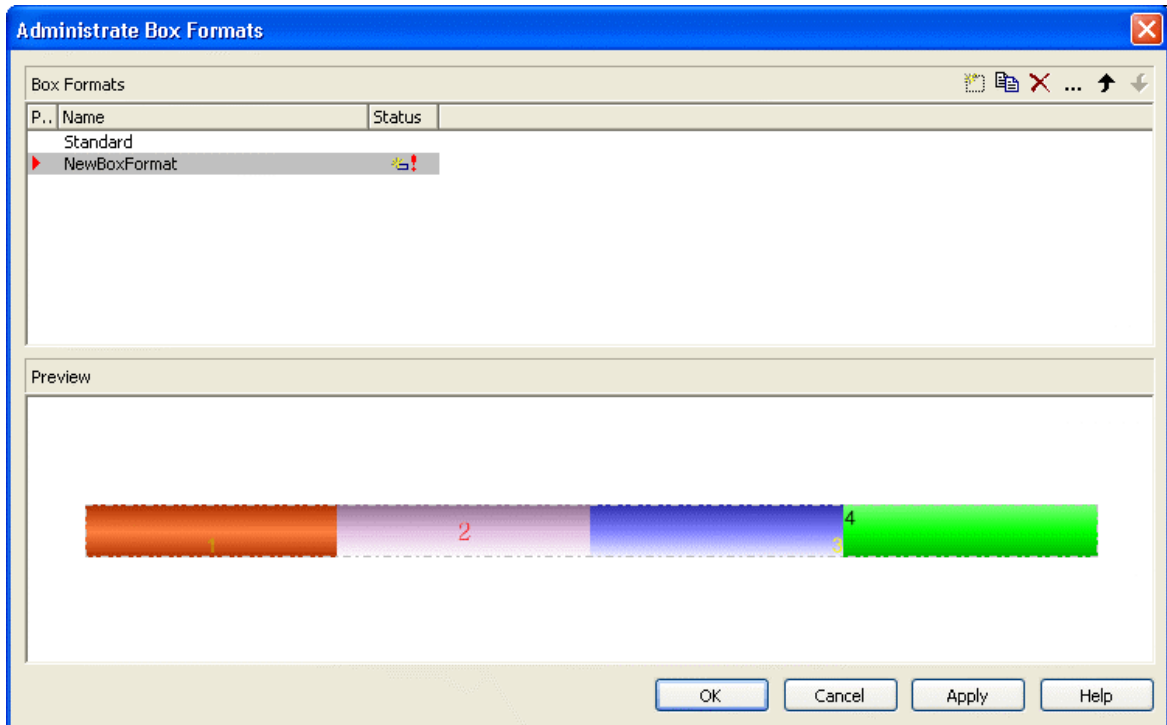
The **Field type** column displays the field types (text or graphics).

You can enter the text of the field or a graphics file name into the **Content** column. 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"). Without the line feed symbol the lines will automatically be separated where blanks occur.

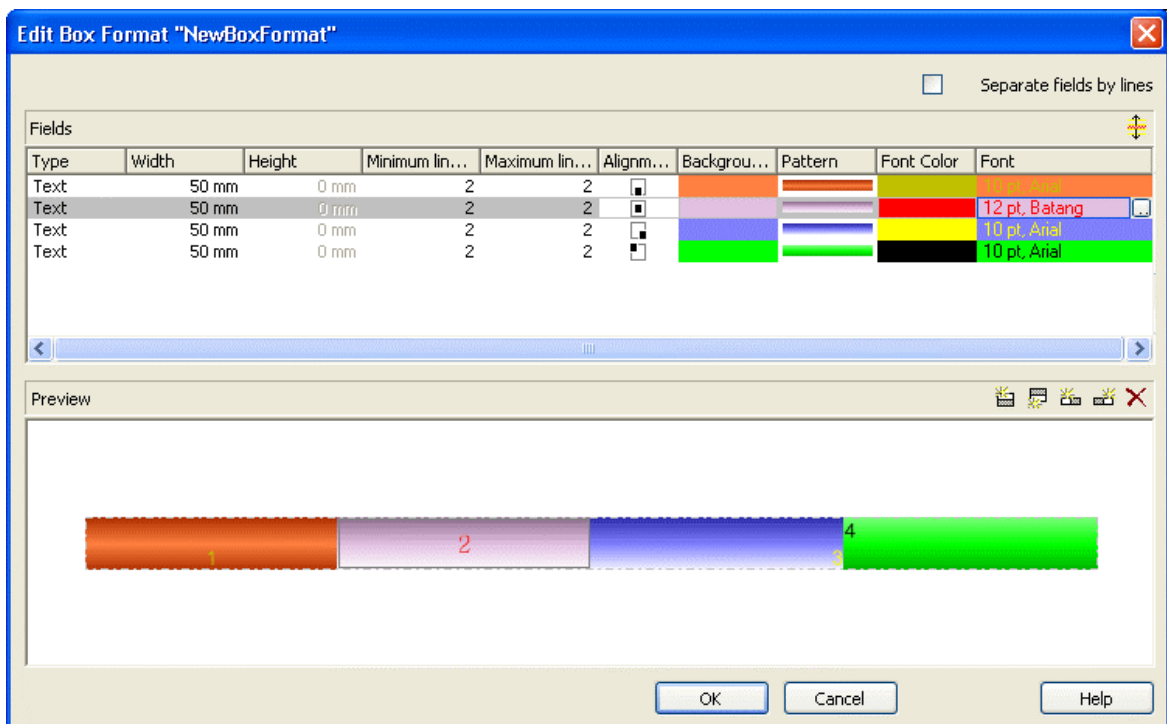
### ► **Box formats**

For each box you can select a box format, and you can specify the box formats.

In the **Administrate Box Formats** dialog box you can add, copy, delete or edit box formats. This dialog box will appear if you click the **Edit** button of the **Box format** field in the **Administrate Boxes** dialog.



In the **Edit Box Format** dialog box you can specify the box format. Click the **...** button in the **Administrate Box Formats** dialog box to open this dialog.



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

Furthermore, the following items can be specified for each box :

- the field type (text or graphics)
- width and height

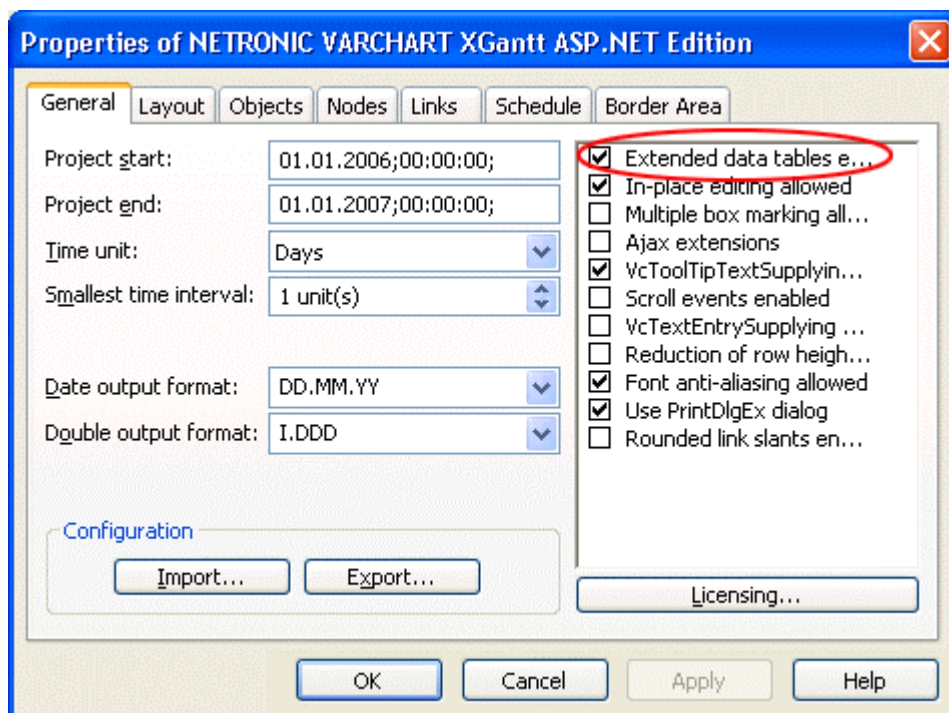
## **68** Important Concepts: Boxes

- how many lines of text can be displayed in the current field
- the alignment
- the background color
- the fill pattern
- the font attributes

### 3.3 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. This helps avoiding redundancies in many cases; it allows to access the main data record by the depending data record and supplies the data required by the resource scheduling module integrated in VARCHART XGantt.

For reasons of compatibility to existing applications VARCHART XGantt continues to operate in the previous mode. 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 enabled** on the property page **General**:







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

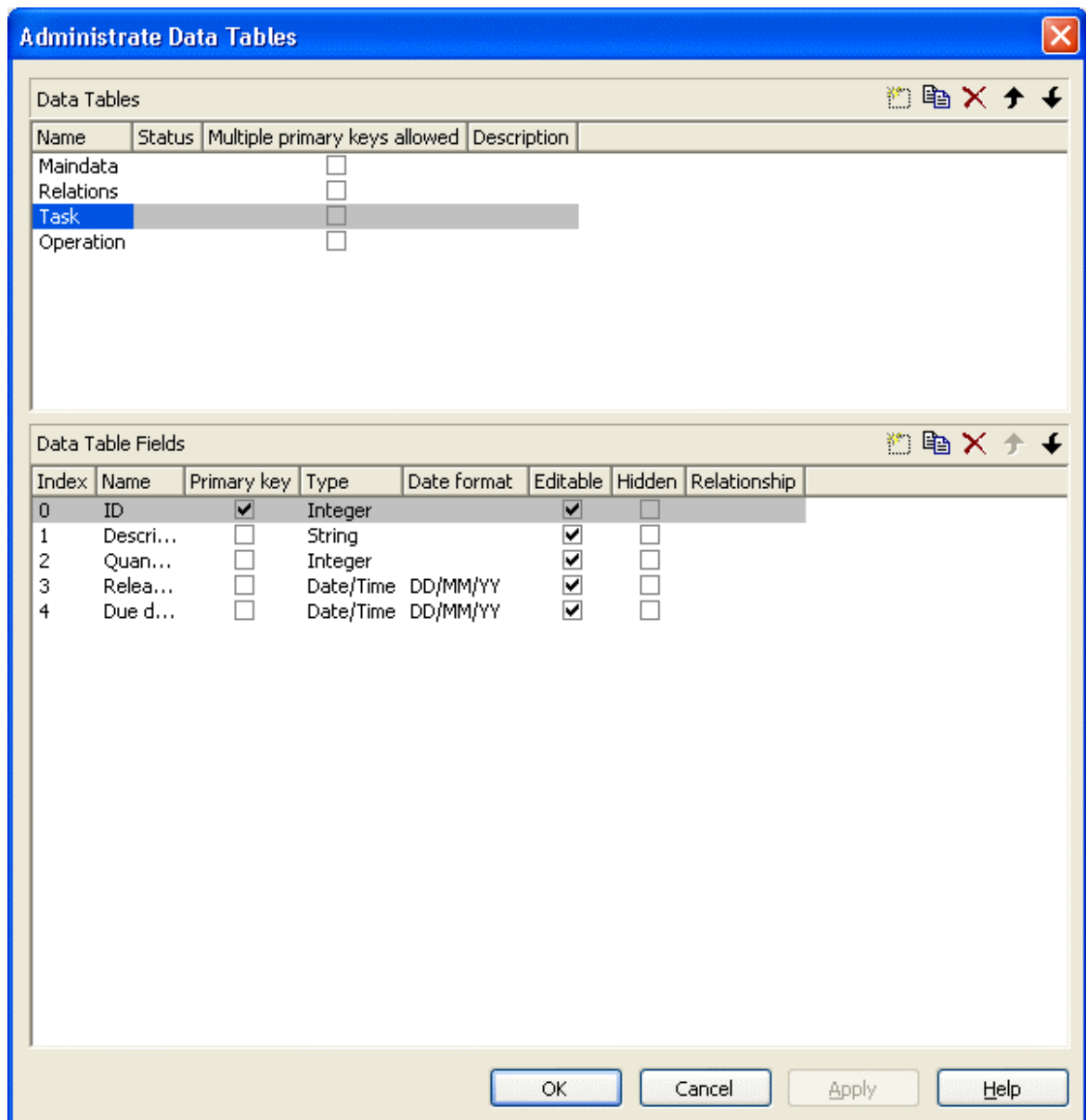
#### ► Handling Data Tables

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 **Administratate Data Tables**. Generating new data tables requires to have

## 70 Important Concepts: Data Tables

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 unique and thus distinguishable. 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**.

For a data table referred to by a relation, selecting a field to be the primary key is compulsory.

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.

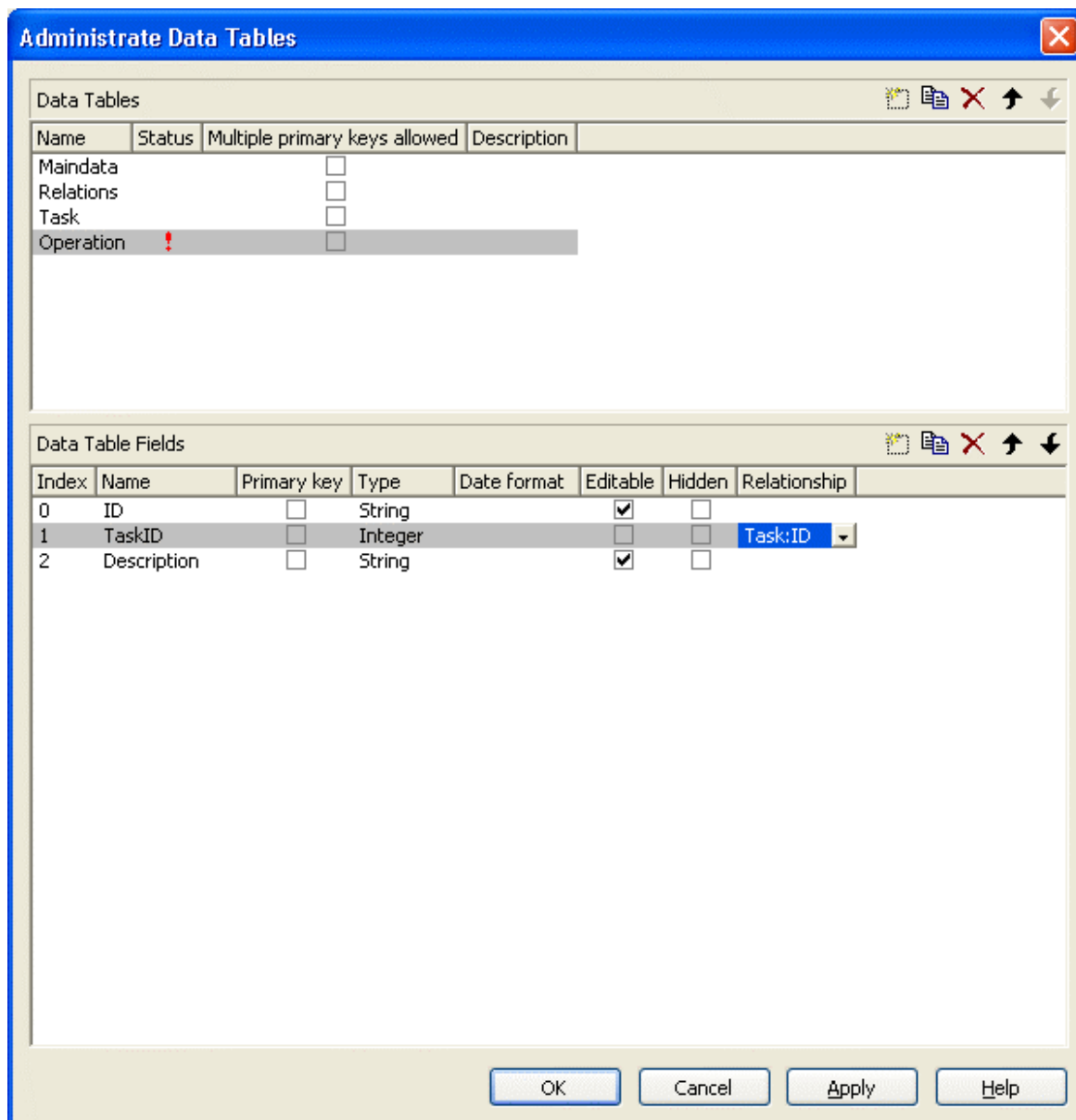
Between two tables A and B at the moment only a single 1:n relationship can be established; a second field of B is not allowed to relate to the primary key of A. Nevertheless, a field of a third table C is allowed to relate to the primary key of table A.

**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 **VcDataRecord-NotFound**.

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



## 72 Important Concepts: Data Tables



### 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 VB.NET**

```

Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable

dataTableCltn = VcGantt1.DataTableCollection
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")

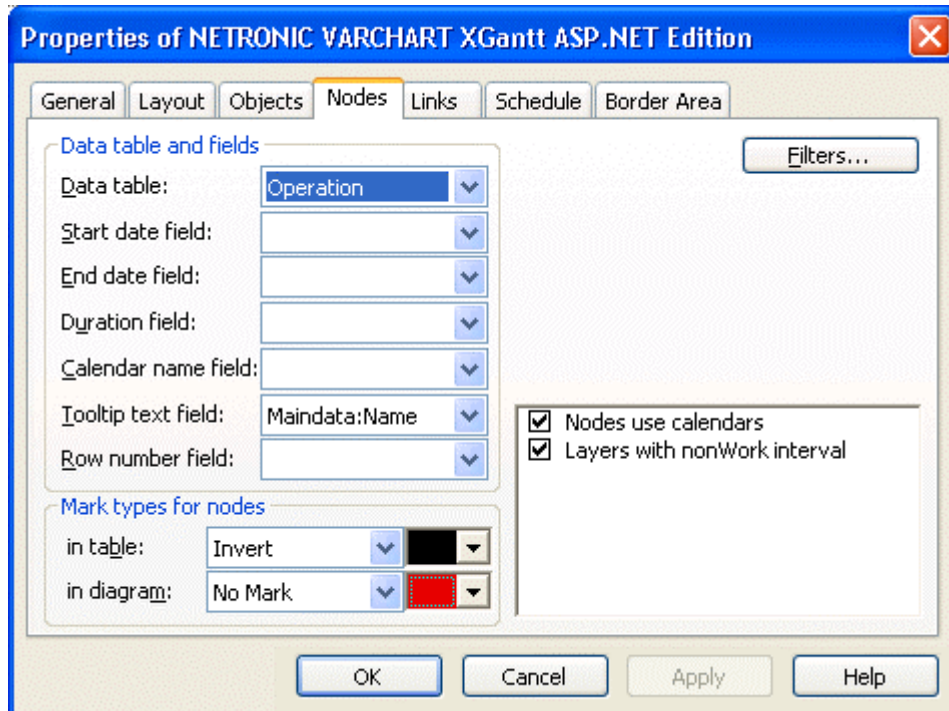
dataTable = dataTableCltn.DataTableByName("Operation")
dataTable.DataRecordCollection.Add("1;1;Operation
1;12.05.2007;14.05.2007")
dataTable.DataRecordCollection.Add("2;1;Operation
2;15.05.2007;19.05.2007")
dataTable.DataRecordCollection.Add("3;2;Operation
3;01.06.2007;05.06.2007")
dataTable.DataRecordCollection.Add("4;2;Operation
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 **Nodes** in the **Data table and fields** 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.

## 74 Important Concepts: Data Tables



This is the result in the table of the Gantt chart if the table **Operation** was taken 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.

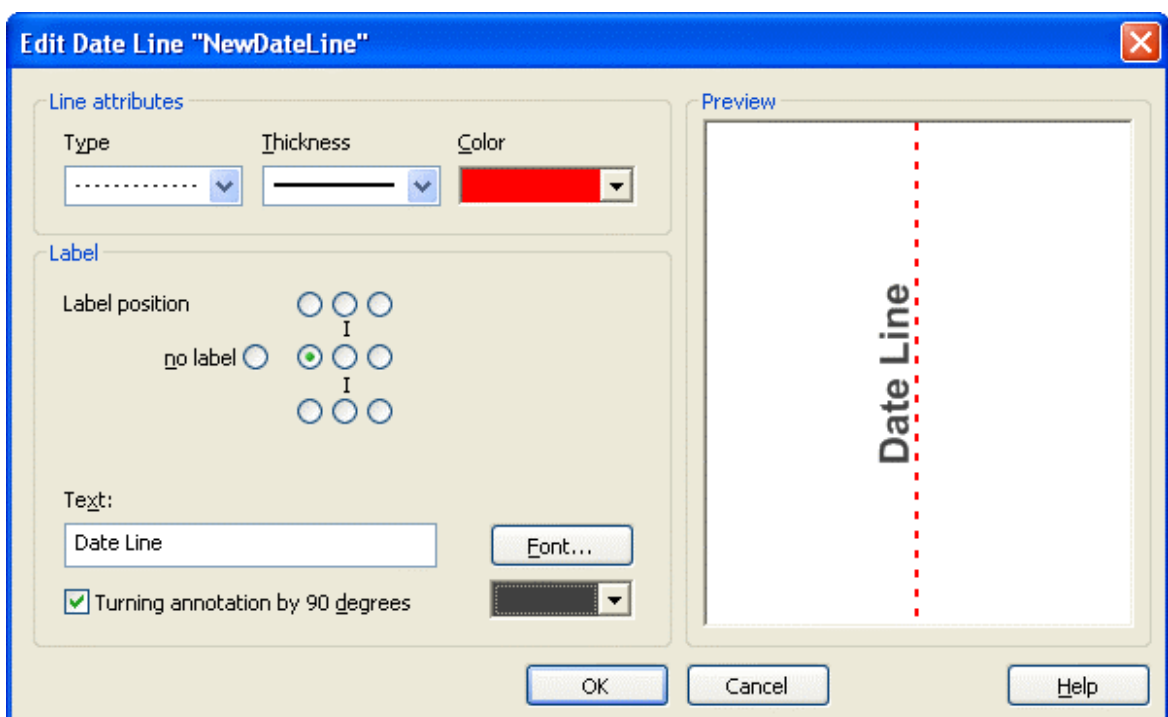
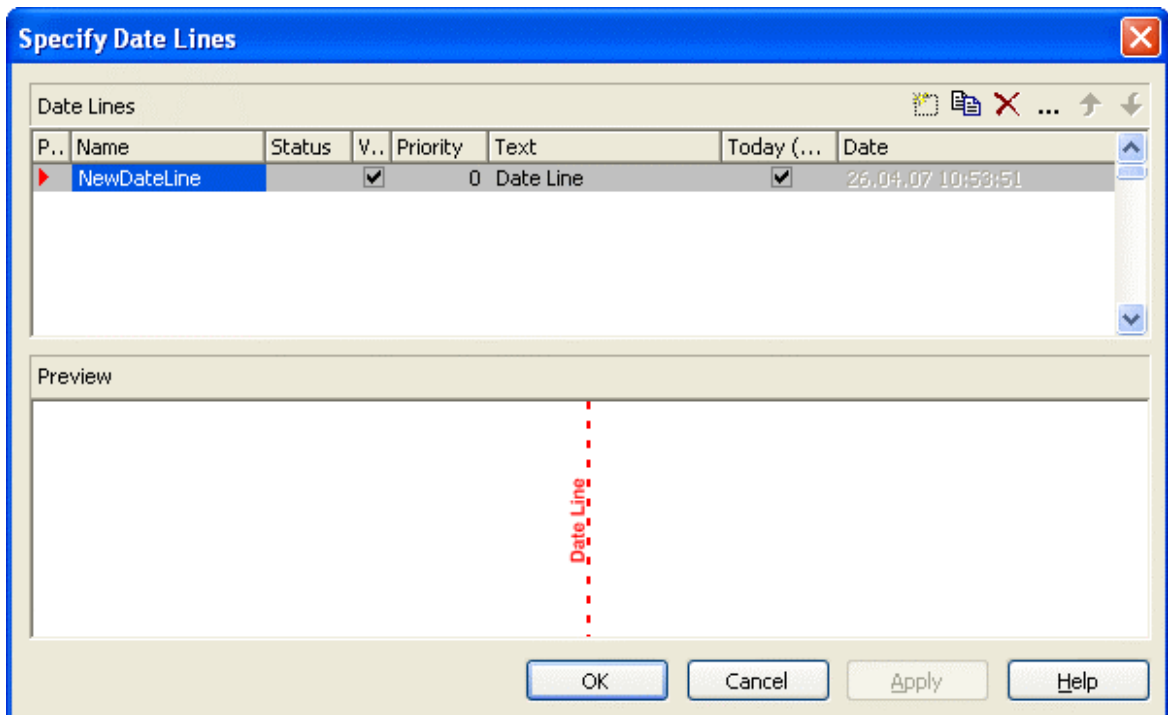
ID	Description	Quantity	Due date	Operation
1	Task 1	10	20.05.07	
2	Task 2	20	15.06.07	

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	VcDataTableCollectionVcDataTable
VcDataDefinitionTable	VcDataTableFieldCollection
VcDefinitionField	VcDataTableField
	VcDataRecordCollection
	VcDataRecord

## 3.4 Date Lines

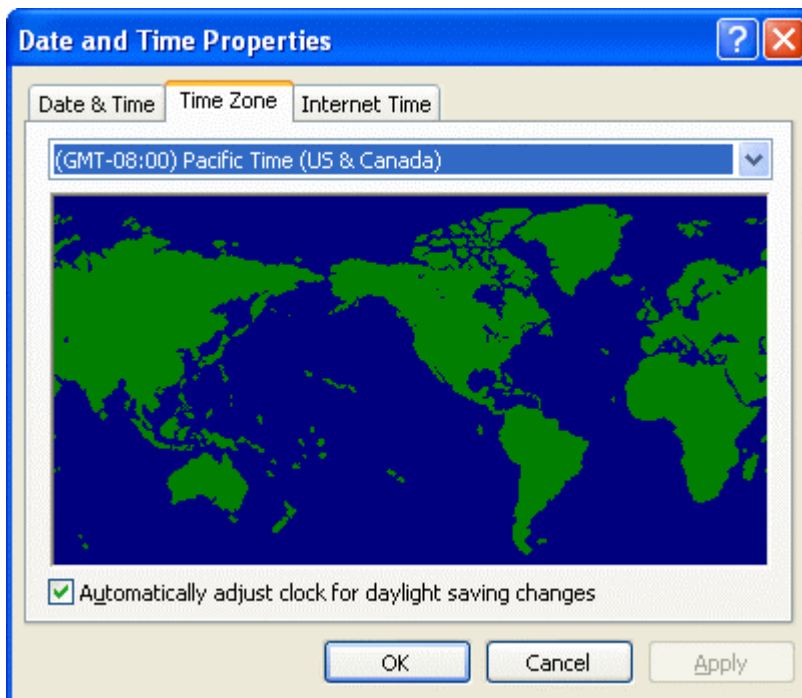
Date lines (vertical lines in the diagram) can highlight certain dates. Their attributes, such as the date, the line type, the priority and whether they are visible can be set in the below "Specify..." and "Edit..." dialogs. You can get to them if you click on **Date lines** button on the **Objects** property page:



## 3.5 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 VARCHART XGantt is running on before they are passed to the VARCHART component. The latter automatically refers to the information on the beginning and the end of daylight saving time which is present in the system.

To make the 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 system by clicking on the button **Start**, then on the menu item **Control Panel**, then on the icon **Date and Time**.



When switching to or back from daylight saving time, 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, dates in those periods consequently will be interpreted in the wrong way.

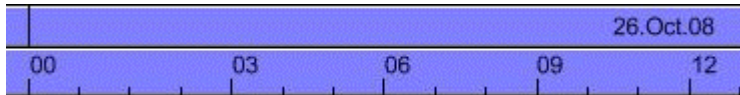
## 78 Important Concepts: Dates and Daylight Saving Time

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 the `VARCHART` components 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 spring (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 **ArgumentOutOfRangeException** 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.

---

## 3.6 Events

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

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

**Note:** By means of the events you can control all interactions and revoke them where required.

### ► Return Status

The below table shows the return status values of VARCHART events:

Constant	value	description
vcRetStatDefault	2	default value
vcRetStatFalse	0	revoking the action



---

## 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 fulfil the criteria defined, e.g. in order to highlight them in the diagram.

When applying a filter, the data of the record is compared to the criteria of the filter. Those layers, curves, links or table formats that fulfil the filter criteria will be selected.

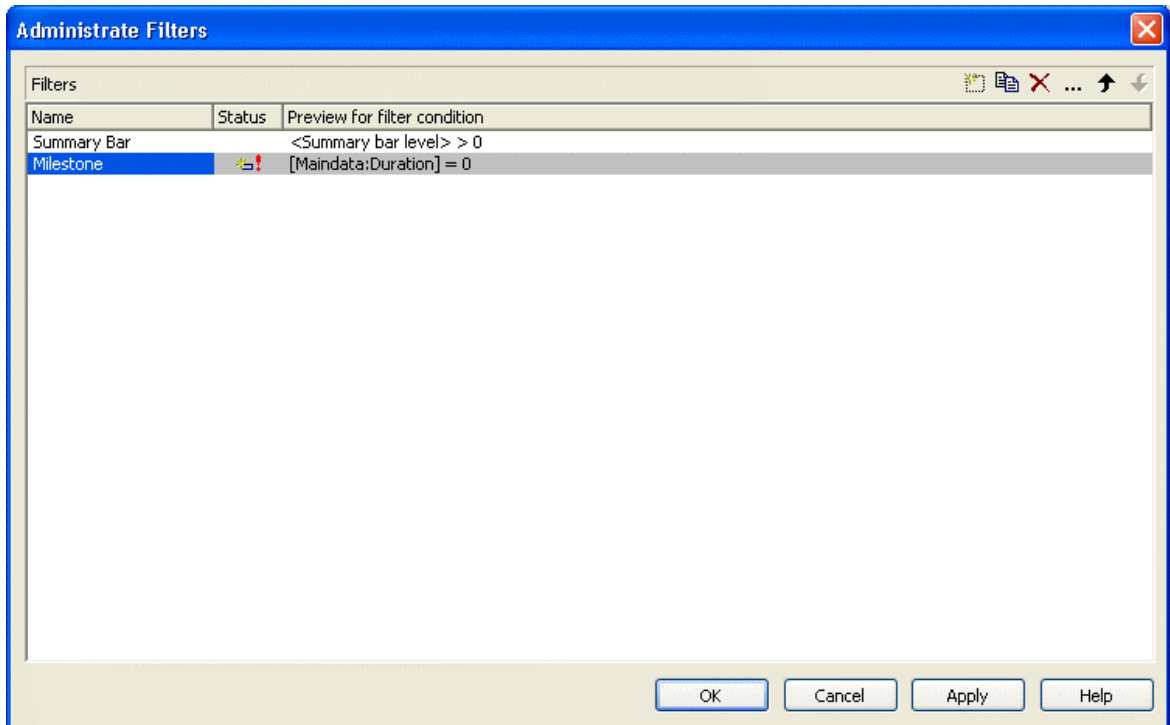
For example, you can define a filter that selects "All activities starting after January 2012".

Filters can only be handled in design mode.

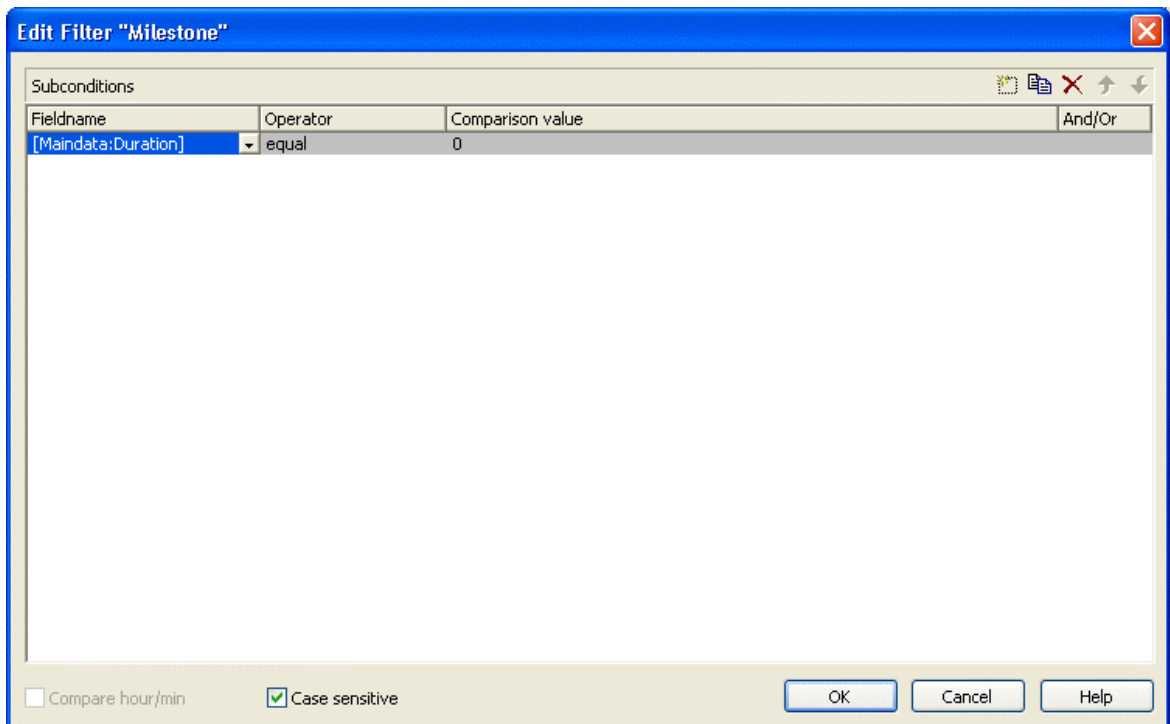
You can reach the **Administrate Filters** dialog box:

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

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



To edit a filter, please click on 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 `VcGanttASP1.ShowGraphics-ExportDialog` and `VcGanttASP1.ExportGraphics`.

#### ► 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 its 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 into an EMF file, discontinuous lines (for example dashed) 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)**

Although the name suggests this format to be an extension of EMF, it is a vector graphics format of its 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 by the VARCHART products for reasons of compatibility.

### ► **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 every display program and internet browser. The format itself is free of copyrights and completely documented.

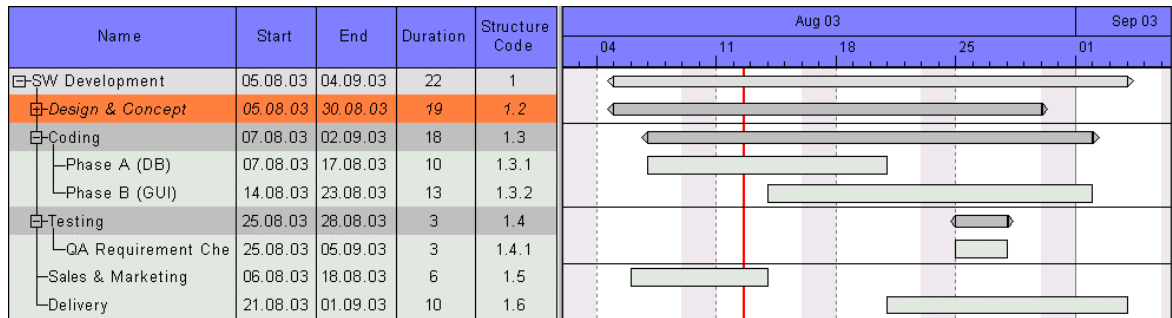
In the VARCHART products from version 4.2 onward the free library **libpng** is used, in order to set a resolution and thus store bitmaps of any size. It has to be taken into account though that very large PNG files may cause problems when loaded, since usually PNG files get completely unpacked in the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

For the format description please see <http://www.libpng.org/pub/png-spec/1.1/PNG-Contents.html>.

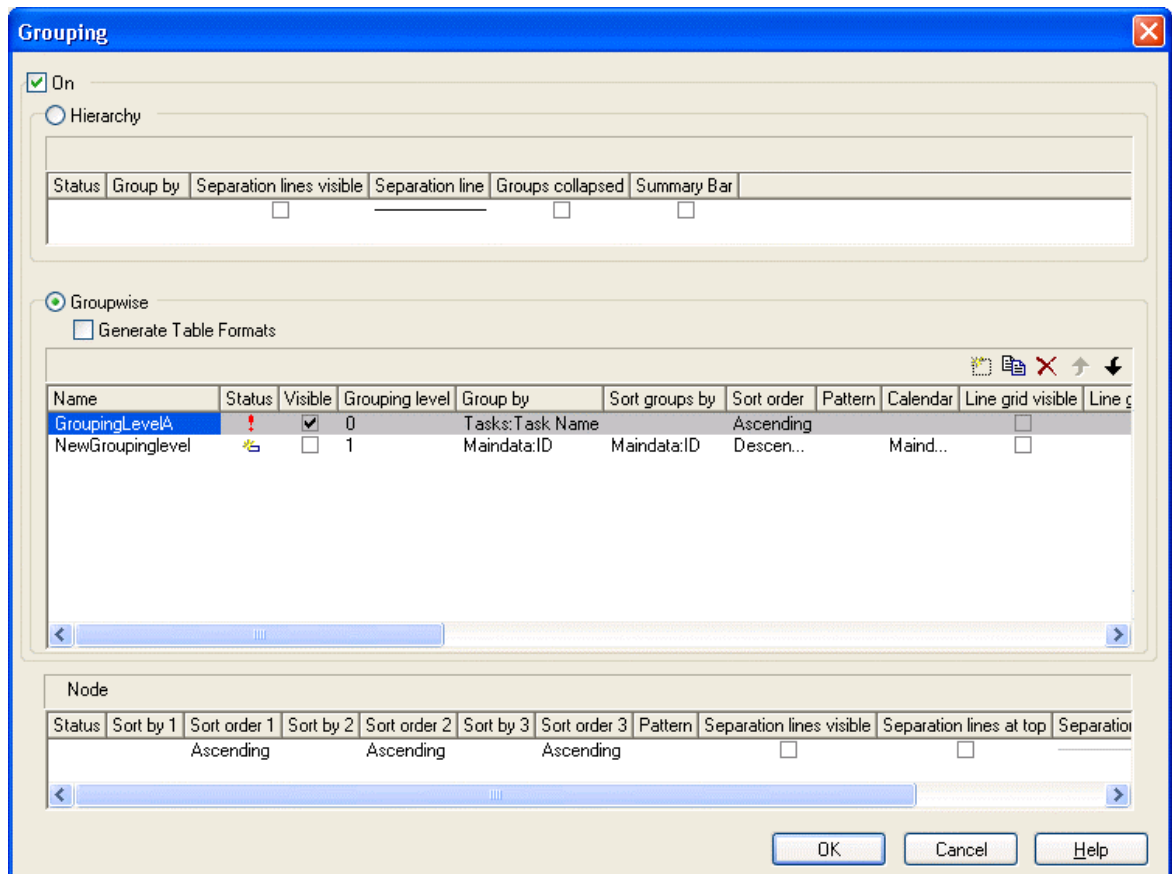
### 3.9 Grouping

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 (tree view style) could look something like this:



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.



In the diagram, an extra row is displayed above a group that contains the group title. The appearance of the group title in the table can be individually defined 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 sign 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 state to the expanded one 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 Bars**)
- whether **Group nodes** are to be displayed

#### ► **Empty Groups**

If you delete all nodes of a group, the title of this group in the table will still remain. If you remove the grouping and apply it again, or if you finish the program and restart it, the titles of all empty groups will disappear.

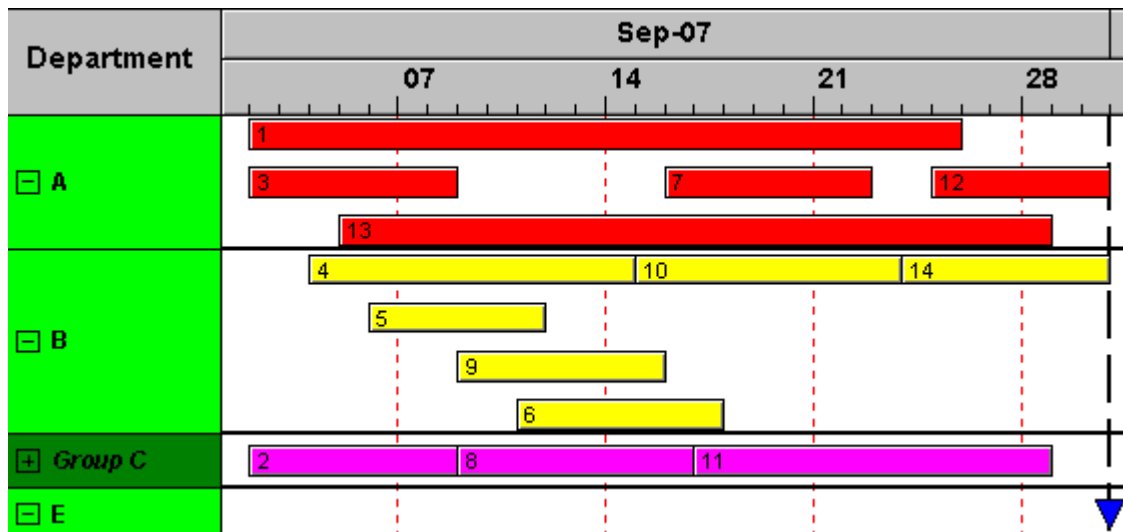
#### ► **Diagram with Grouping Option "Nodes in One Line"**

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



## 88 Important Concepts: Grouping

A diagram with this option enabled looks something like this:



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 Overlapping nodes

If the **Nodes in separate rows** mode was not selected, you can specify via the **sorting order** which nodes lie above the others. The nodes are sorted according to their sorting order, that means that the last node in the sorting order lies above all others and is completely visible.

### ► Summary bars

Summary bars can be displayed in the grouping lines. You can specify whether summary bars are to be displayed and for which grouping levels.

**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** lets you specify/enquire at run time whether summary bars are visible or not. On condition that the grouping is not hierarchically, you can switch on or off the summary bars for each level separately with the help of the parameter **GroupingLevel**.

On the **Layer** property page you can specify the appearance of the summary bars by creating appropriate layers which visualize the summary bars. You may define one layer for all or several levels as well as different layers for each level, e. g. the layer "Summary bar 1" for the first level, "Summary bar 2" for the second level etc.

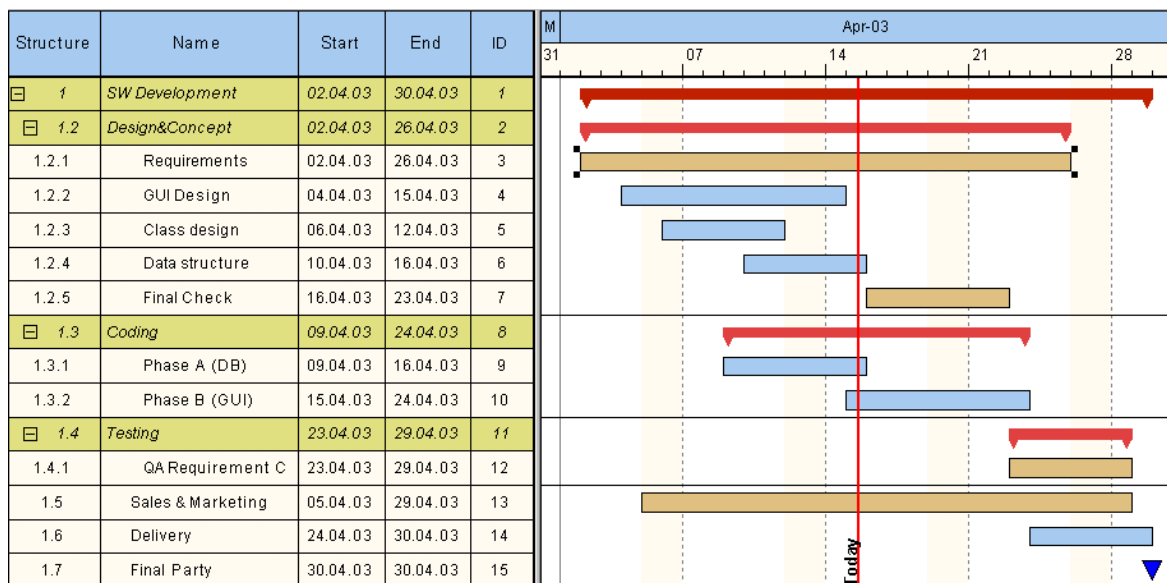
Now you have to assign corresponding filters to the summary bars so that the visualization is carried out at all. Filters can be created in the **Administrate Filters** dialog, e.g. the filter "Summary bar 1" for the first level. In order to specify the appropriate level, in the **Edit Filter** dialog select "<summary bar-level>" under **Field name**, select the right **Operator** (equal, greater or equal, greater than, etc.) and enter the desired level number in the **Comparison** field.

## 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:

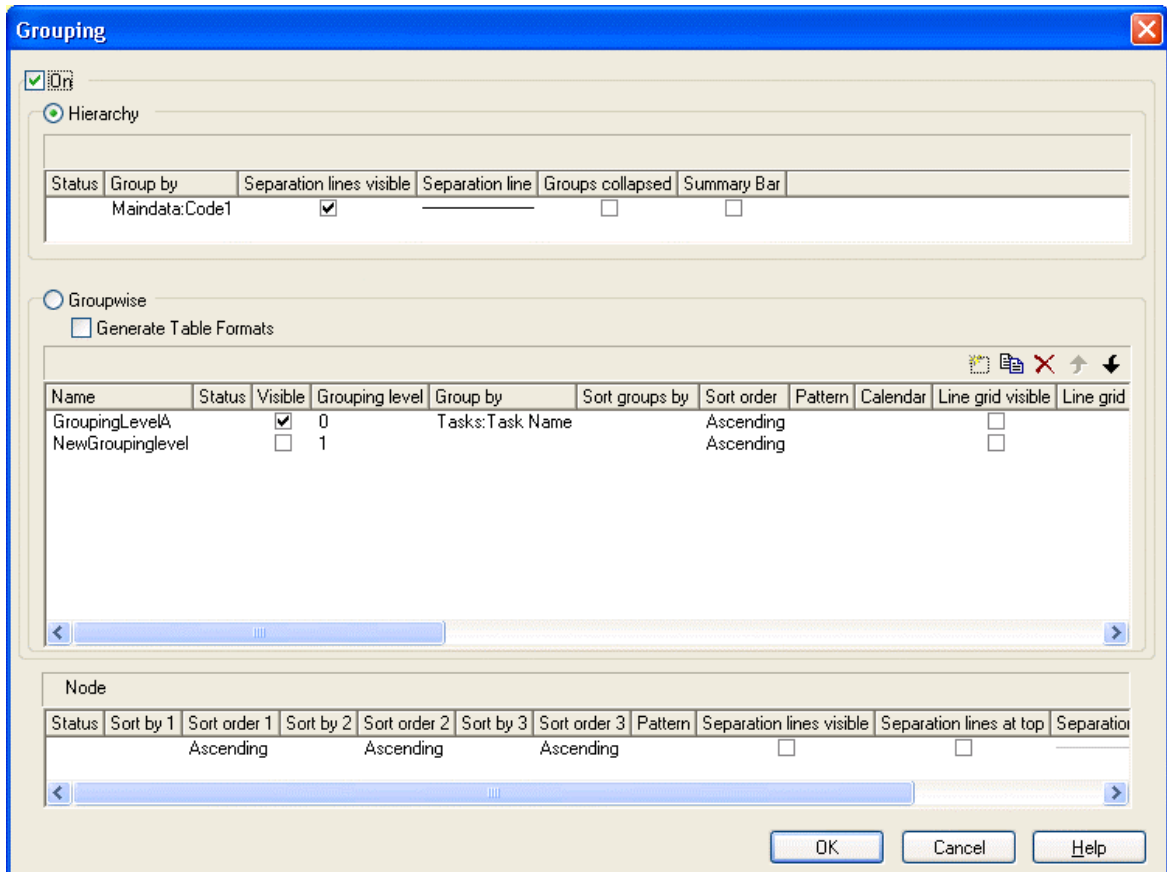


The symbols + and – are automatically displayed in front of the superordinate activities. Sublevels are indented automatically. By clicking on the - symbol, the structure of subordinate activities will fold (collapse); by clicking on the + symbol it will unfold (expand).

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:

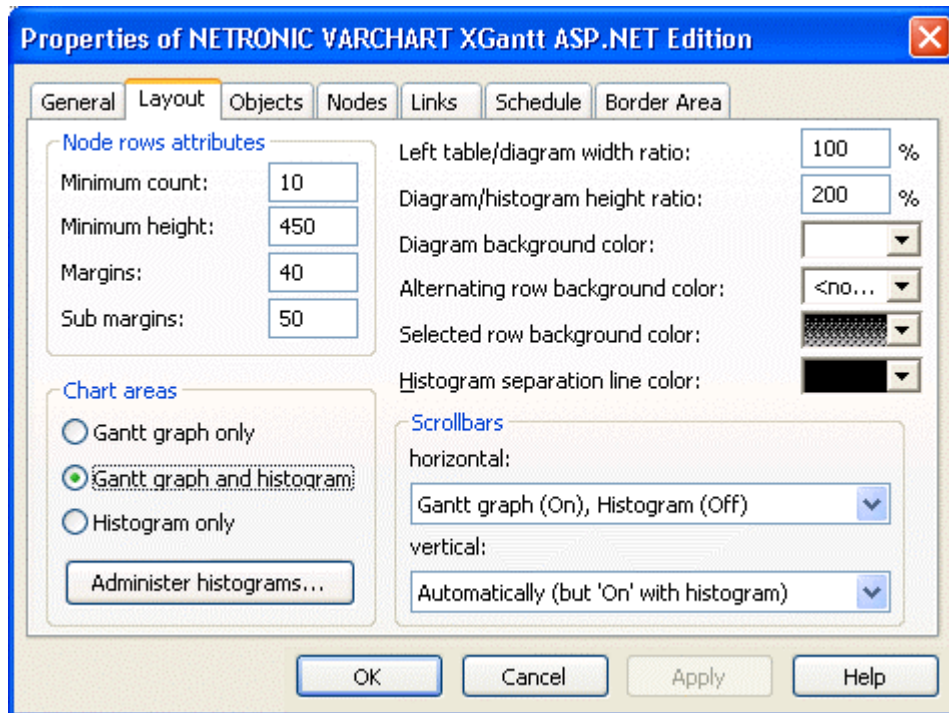
- Display and style of **Separation lines**
- whether the activities should be collapsed on the start of the program (**Groups collapsed**)
- whether summary bars are to be displayed (**Summary Bar**)

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

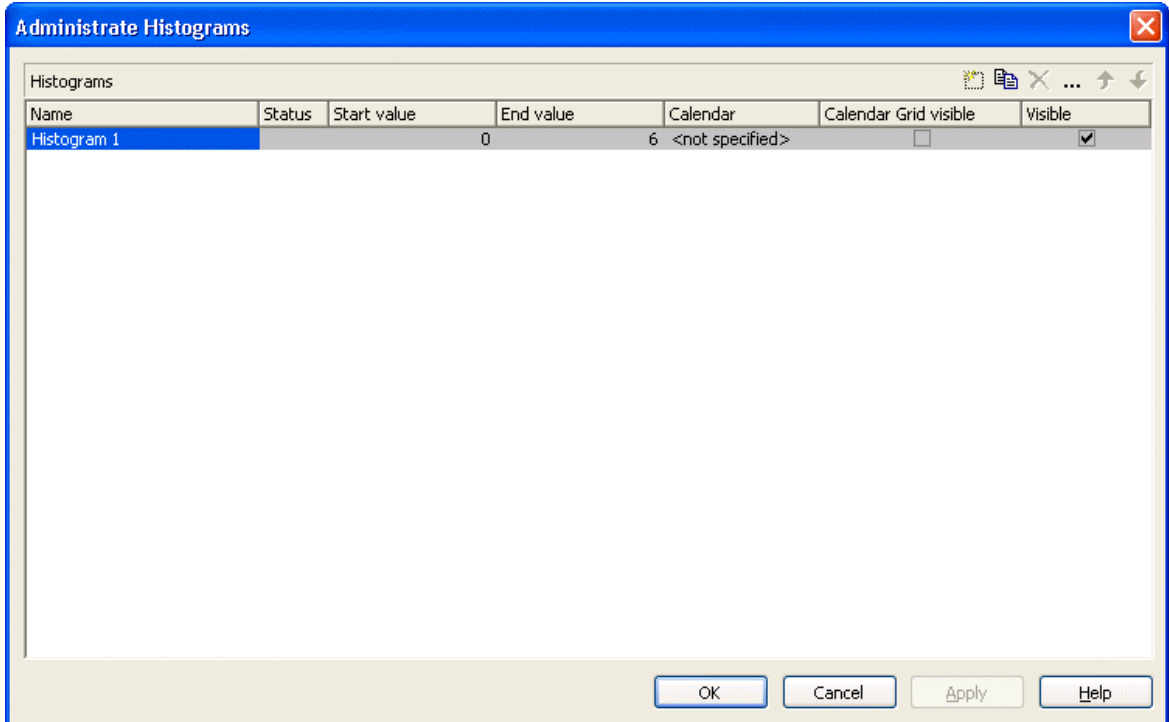
## 3.11 Histograms

Histograms are used to summarize activities to curves, with the activities fulfilling certain criteria.

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



To select the histogram(s) to be displayed and for editing histograms, please click on the button **Administer histograms**.

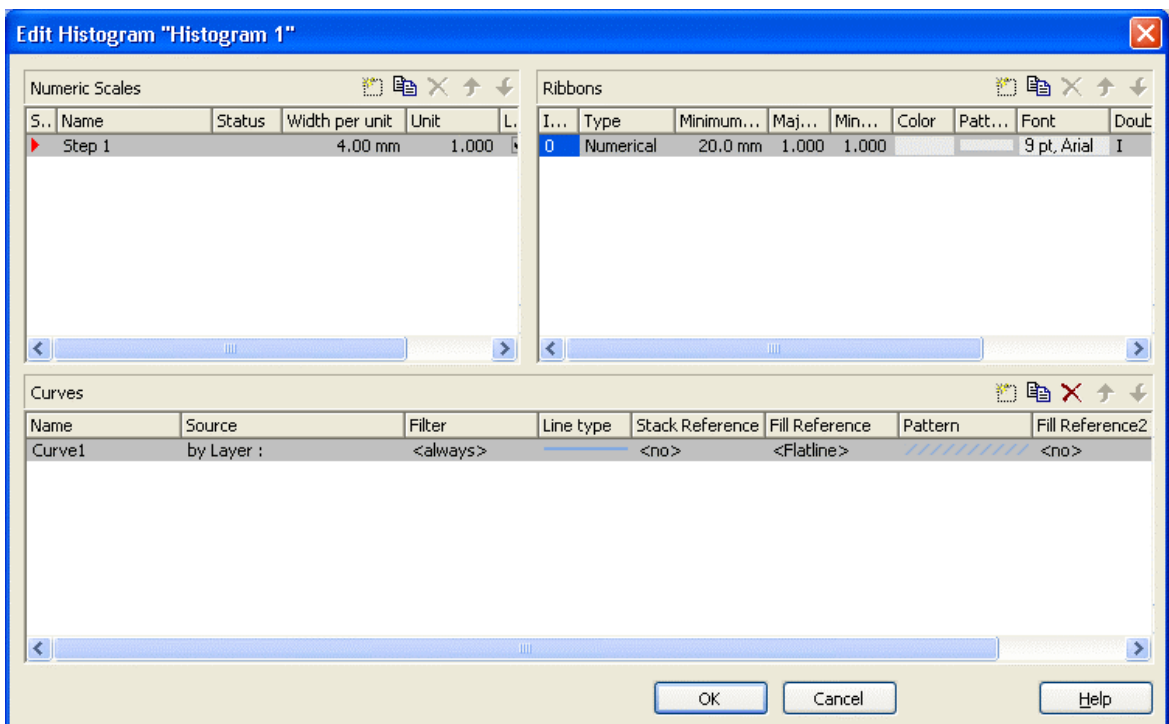


The dialog allows to select one or more histograms to be displayed.

A histogram has a numeric scale (y axis) and curves. Its x axis is scaled by the Gantt chart time scale.

For each histogram you can define the start and the end values of the numeric scale separately.

To edit a histogram, please mark it and click on the **Edit** button (...).

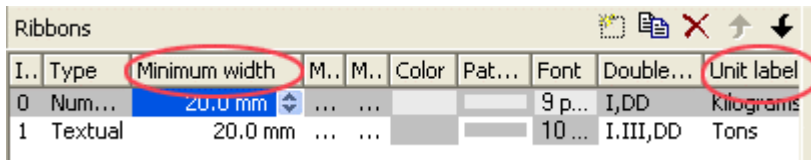


## 94 Important Concepts: Histograms

### ► Numeric scales

In the above dialog you can define different numeric scales and select the 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 you can define its features.

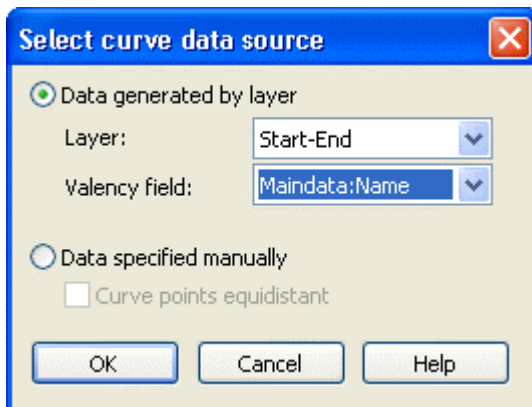
In the **Ribbons** area you can assign one or more ribbons to the numeric scale being edited. To each ribbon you can set a **Type**, a **Minimum width**, a number to define after how many units a **Major** or a **Minor ticks** should occur, you can assign a background **Color**, **Font** features, a **Double format** and 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 remains invisible.



I..	Type	Minimum width	M..	M..	Color	Pat...	Font	Double...	Unit label
0	Num...	20.0 mm	...	...	...	...	9 p...	I,DD	Kilograms
1	Textual	20.0 mm	...	...	...	...	10...	I.III,DD	Tons

### ► Histogram curves

A histogram may contain several capacity curves, for each of which you can individually define a number of parameters. A **Name** and a **Line type** are the most simple ones. For a curve to be generated, a **Source** needs to be specified to supply the data, i.e. the values of the points. For this, please click on the **Source** field and then on the **Edit** button (...). The below dialog will appear:



Select curve data source

Data generated by layer

Layer: Start-End

Valency field: Maindata:Name

Data specified manually

Curve points equidistant

OK Cancel Help

You can choose between two alternatives:

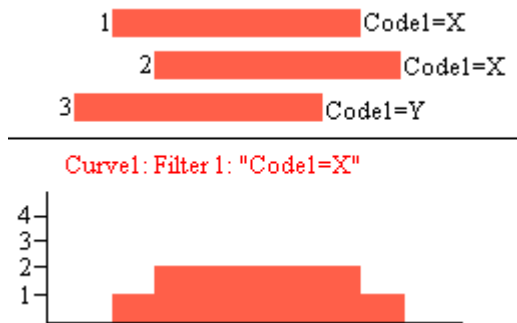
#### ► 1. Data generated by layers

The curves are generated from data of the activities. When summing up the activities to a curve, the start and end dates of each of the selected layers (e.g. named "Start-End") are used.

When generating data from layers, you can once more filter out certain layers that shall add to the curve. For this, in the upstream **Edit Histogram** dialog you can choose a **Filter** for each curve for the selection of activities.

**Example:**

Only those activities that fulfil the conditions of Filter1 contribute to Curve1. Filter1 contains the condition "Code1 = X", i.e. only the activities 1 and 2 for which "Code1 = X" applies contribute to Curve 1.



For curves generated by layers you can select a data field of the activity that provides the fraction on the scale, by which the curve is to rise on the numeric scale when an activity is added (e.g. by 5 units).

► **2. Data defined by the API**

This option allows to set curve values by the API. The latter offers the VcCurve method **SetValues** by which the points can be freely defined.

For curves generated by the API you can set in the **Select curve data source** dialog whether the curve points are to be created with a regular spacing (**Curve points equidistant**), where the curve points cannot be modified interactively, or in arbitrary positions that allow for interactive modifications.

*Curve points equidistant:* Please specify the start value (**startDate**) and the y values of the histogram curve. The curve points are calculated from the start value and the values set to **Time Unit** and **Smallest time interval** (on the property page **General**).

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

Curves generated in this way cannot be edited interactively.

*Curve points not equidistant:* Specify pairs of x and y values:

Set Values X1, Y1

Set Values X2, Y2

Set Values X3, Y3...



The fields **Time Unit** and **Smallest time interval** do not apply. A curve generated this way can be edited interactively.

### ► **Reference curve**

A typical way to use curves defined by the **SetValues** method of the API is the capacity curve. It mostly serves as a reference curve, that forms areas with other curves, to which colors and patterns can be assigned.

The **Fill Reference** field allows you to specify the curve to limit the opposite end of the area (starting at the curve being edited). If you select <Flatline>, the area will reach down to the x axis, possibly hiding other curves on its way (which depends on the drawing priorities of the curves). In applications often the capacity curve is assigned here.

The curve line and fill pattern of the area you can set in the **Line type** and **Pattern** fields.

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 each 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/background colors for the fill pattern below a curve.

In addition to the first reference curve, you can specify a second one.

For this, please select a curve in the **Fill Reference2** field. The area between the curves is displayed only if the y values of the curve being edited are higher than the y values of the second reference curve, i.e. if the area expands below the curve being edited.

In the corresponding **Pattern** field you can specify the pattern and the color of the area.

Examples of handling histograms you can find in "Tutorial: Displaying Histograms" and in "Tutorial: Displaying Capacity Bottlenecks".



### ► **Stacking Curves**

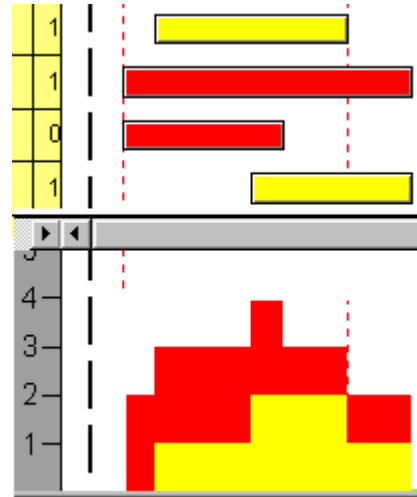
Stacking curves is useful, for instance, if a histogram displays different curves that visualize the workload of single resources but in addition shall indicate the total workload.

In the example below, there are red and yellow activities indicating that they occupy different resources. Filters of corresponding conditions collect them to form a curve each. Being stacked the curves indicate the total workload of the system.

To display the red urve on top of the yellow one, please select the yellow curve (Curve 1) to be the **Stack Reference** of the red curve (Curve 2).

(If you select <No> in the **Stack Reference** for all curves, the curves will overlap visually and may hide each other).

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



*Curve2 is stacked on Curve1.*

---

## **3.12 How to Use a Calendar**

## 3.13 Layers

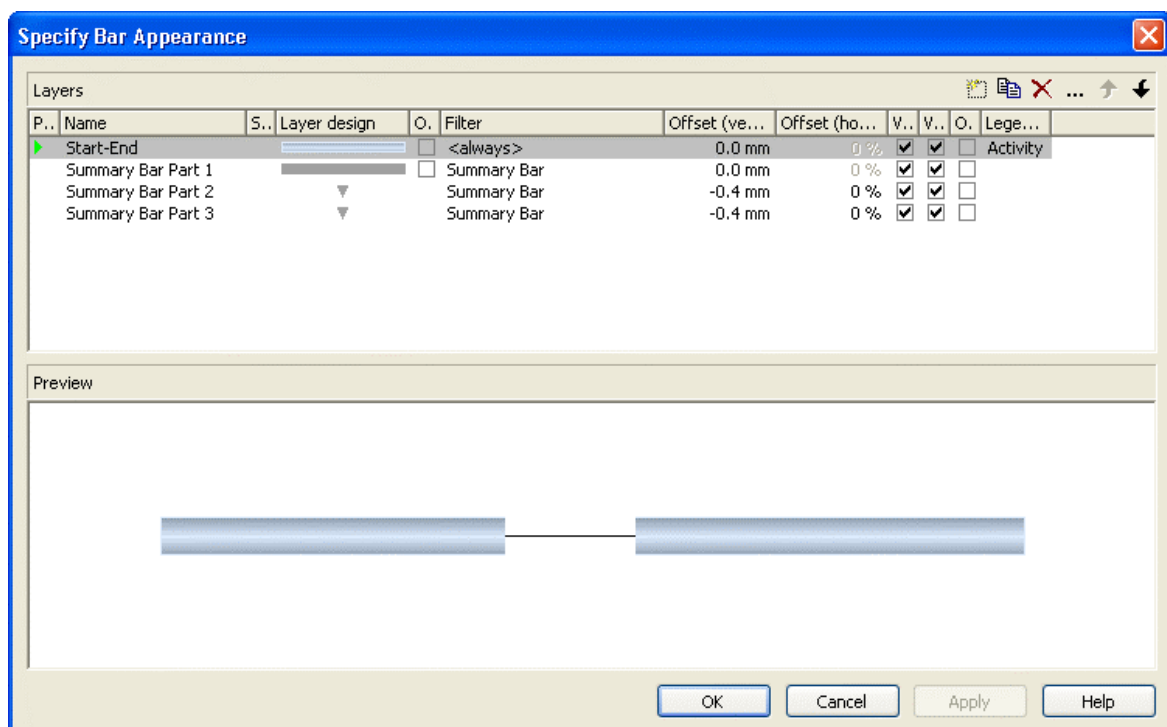
A layer is the graphical representation of a single date (symbol or bitmap layers) or of a pair of dates (rectangle, wedge-shaped or line layers).



Activities (or nodes) are graphically displayed by one or more layers. If an activity comprises several layers, the layers are graphically "stratified", starting with the layer of lowest priority and finishing with the layer of highest priority.


For each layer a filter is used. By the filter, only those layers are collected that comply with the criteria defined by the filter.

Layers can have different patterns, background colors, pattern colors and annotations. In addition, they can be of varying heights and may have a vertical and horizontal offsets. These options enable the layers of a node to differ from each other and to remain visible when displayed.

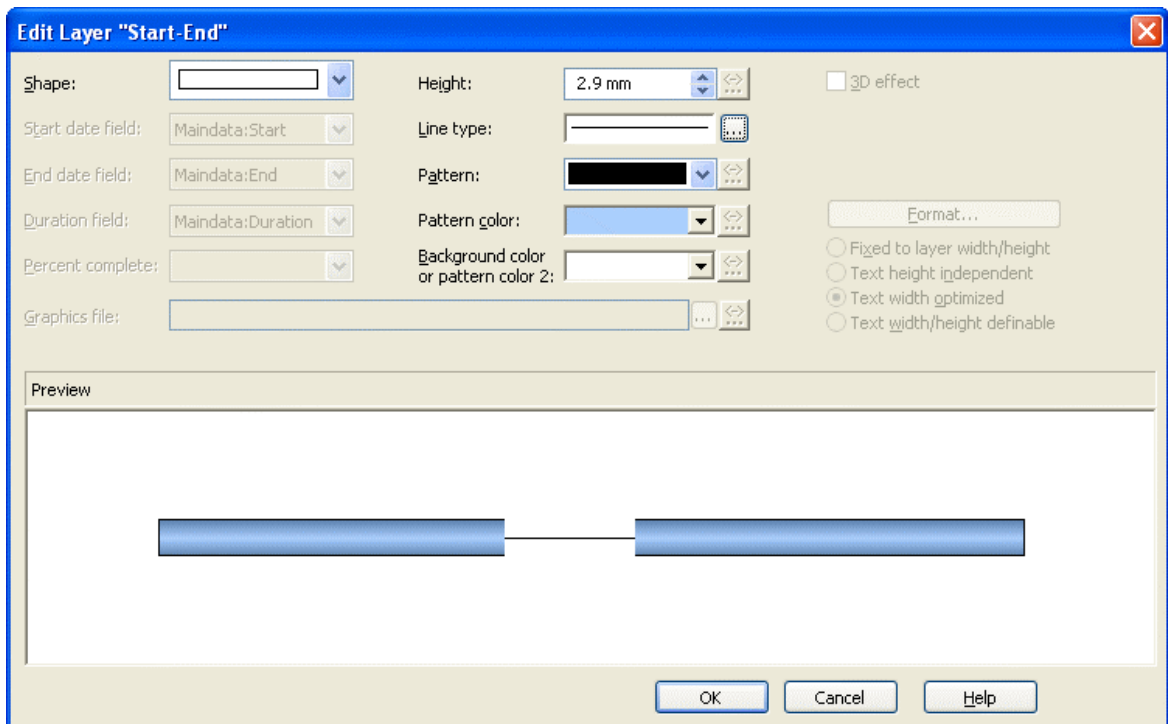
In the **Specify Bar Appearance** dialog box, you can define the layers of a node and specify their drawing priority.



By clicking on the buttons right-hand at the top of the layer you can add () , copy () , delete () or edit layers () .

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

## 100 Important Concepts: Layers



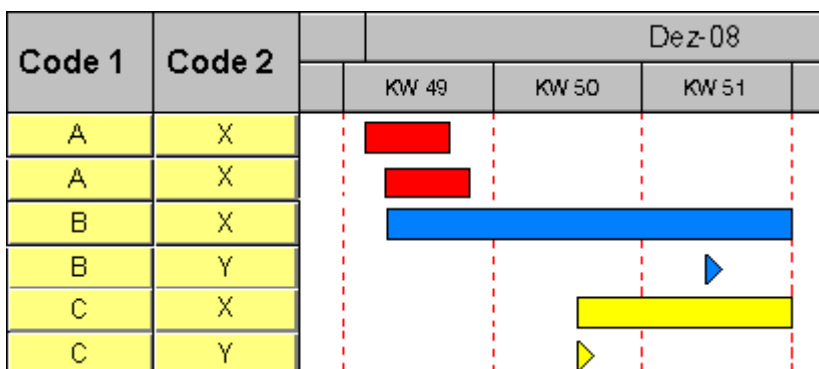
### ► Filter for a layer

By using filters, you can assign a layer to selected nodes, the selection of which depends on their data.

To define the conditions of a filter, in the **Specify Bar Appearance** dialog please click in the **Filter** field. Of the two buttons appearing, please click on the **...** button to open the **Administrate Filters** dialog box. You can reach the **Edit Filter** dialog by pressing **...** in the top right corner of the window.

(Also see "Important Concepts: Filters".)

In the example below, for a value of "X" in the field "Code2" a rectangle layer is defined and for a value of "Y" a symbol layer is defined. The colors are mapped using the field "Code1".



► **Layer shapes**

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

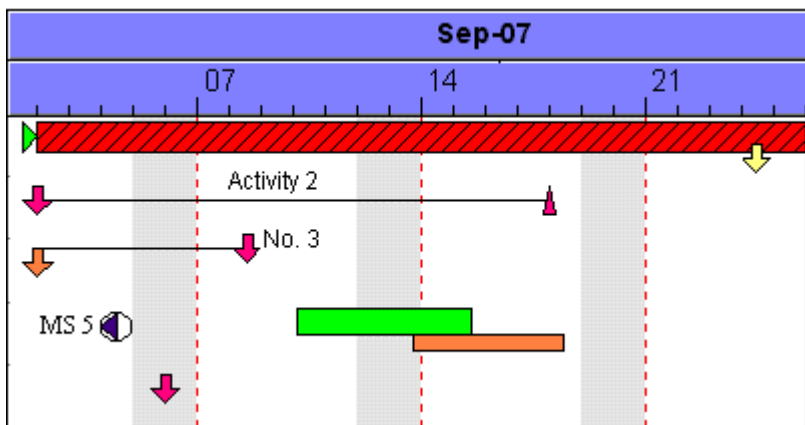
Select the layer shape from the **Shape** select box in the **Edit Layer** dialog box.

Symbol layers represent specific dates in time. There are some symbol layers that were predefined, but you can also define your own symbol layers. For example, you can use your company logo as a bitmap layer. You can specify the desired bitmap files in the **Graphics file** field.

Pairs of dates are visualized by rectangle, wedge-shaped or line layers. Wedge-shaped layers are useful for visualising increasing and decreasing activities, e. g. during the project start or end.

Of an invisible symbol layer, only its annotation is visible, and these layers will not be displayed in the legend.

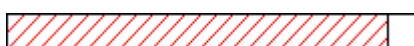
Combining layer forms, patterns, colors and filters, a large variety of different layers can be defined. The below picture shows some examples:



► **Degree of completion**

VARCHARTXGantt allows to display the degree of completion of an activity. For this, please

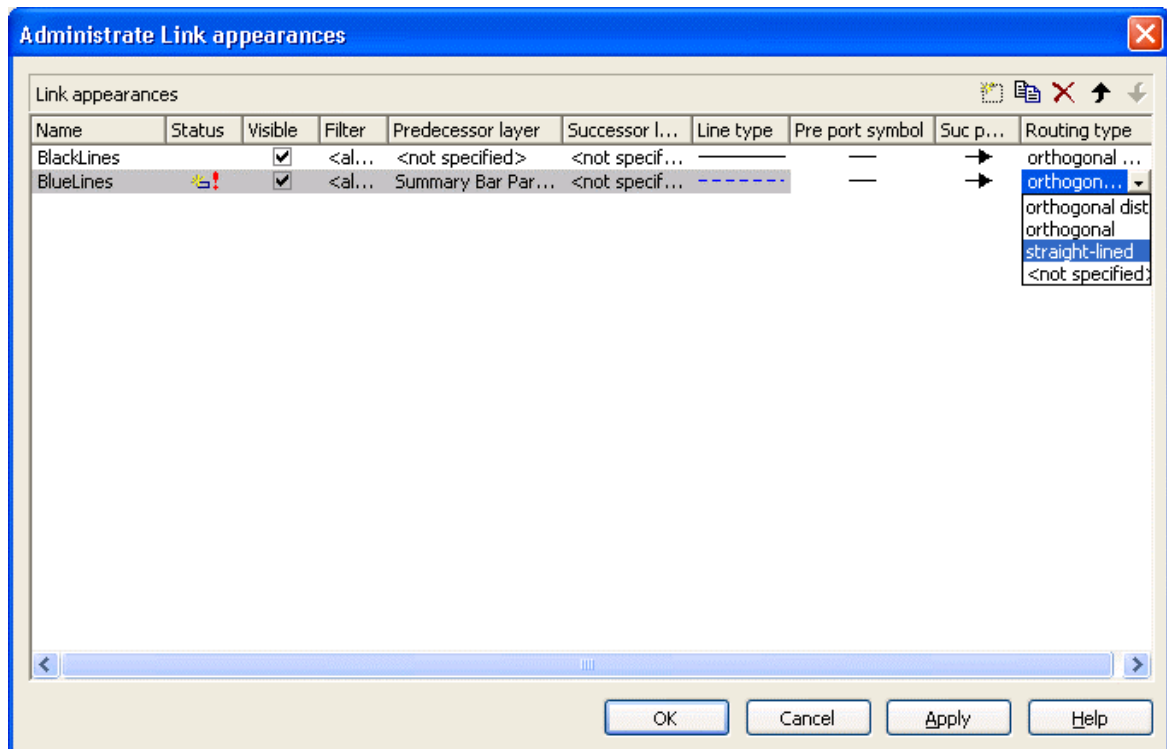
create a layer named "Completed" and edit it by using the **Edit Layer** dialog box. For wedge-shaped and rectangle layers you can select the data field that contains the percentage degree of completion of the selected layer. So, for the layer "Completed" please select the data field "% completed". Now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



*Degree of completion: 90 %*

## 3.14 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.



### ► Further Specifications for the Link Appearances

For further information about link appearances please see chapter 4.28 "The Administrate Link Appearances Dialog Box".

---

## 3.15 Links

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

### ► Generating Links

Links can be generated via the API by the method **VcGantt.InsertLinkRecord**. You can delete links by the methods **VcGantt.DeleteLinkRecord** and **VcLink.DeleteLink**.

### ► Events

You can react to the following events:

**VcLinksLeftClicking**

**VcLinksMarked**

**VcLinksMarking**

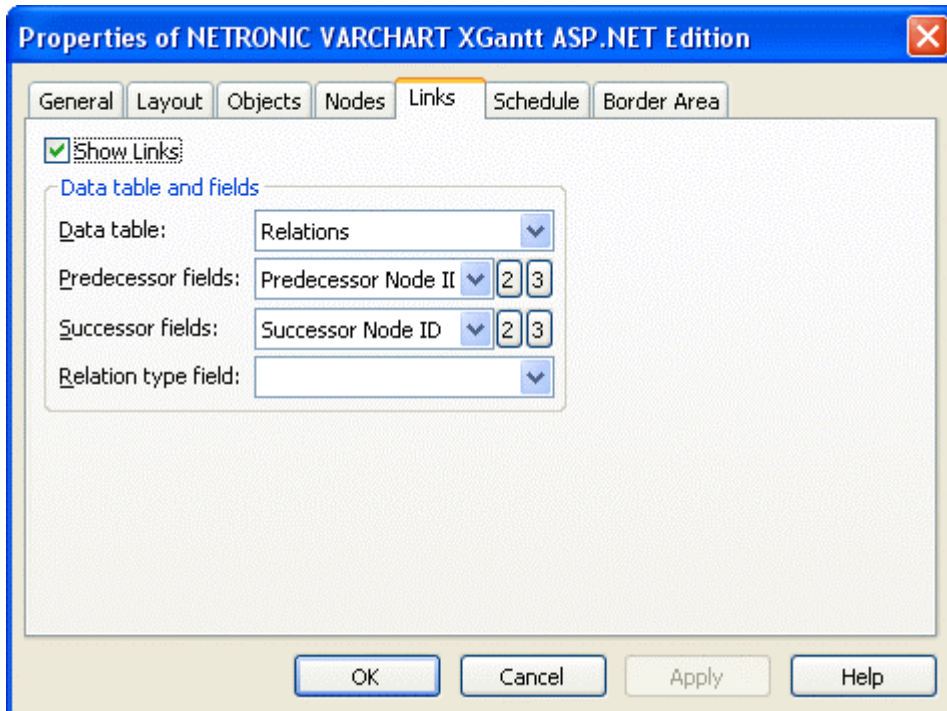
### ► Specifying Links

On the **Links** property page you can choose whether the links are to be displayed, and, if desired, set more options.

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.



## 104 Important Concepts: Links



You can specify 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

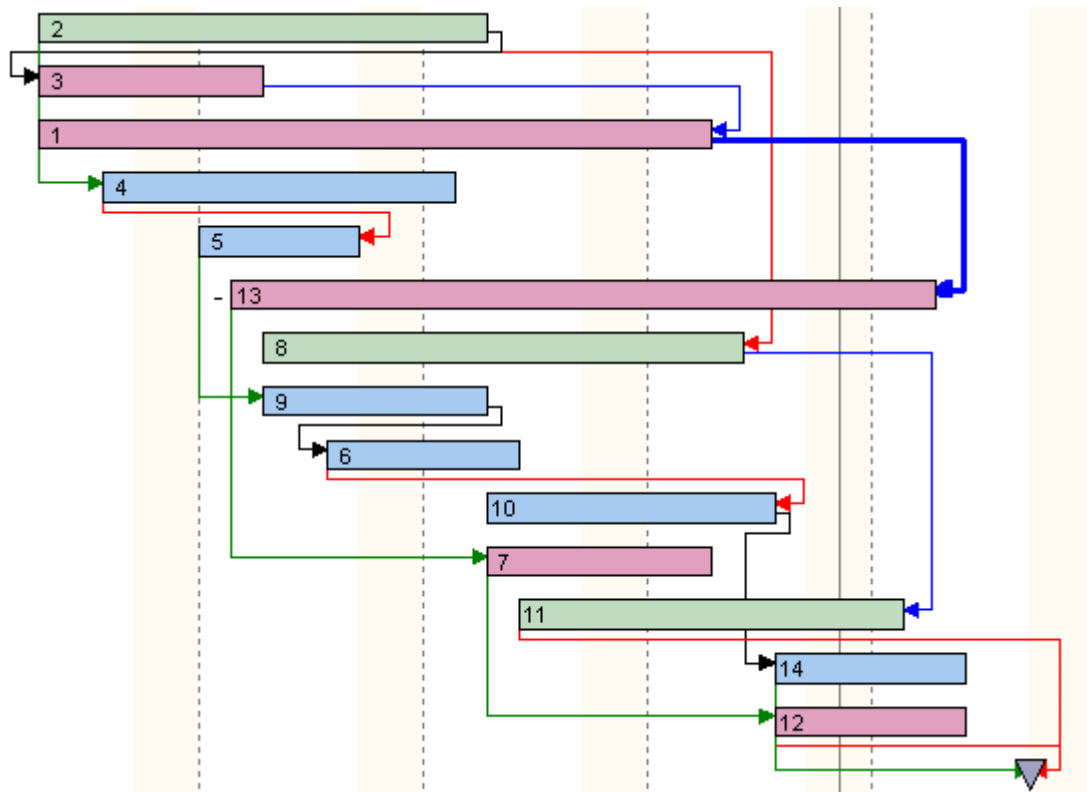
### ► Types of Links

In the combo box **Relation type field** you can select a data field that the link type is to be loaded from.

#### **Link types:**

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

This data field enables the link type to be visualized by the appropriate line routing.



*Example visualizing different link types*

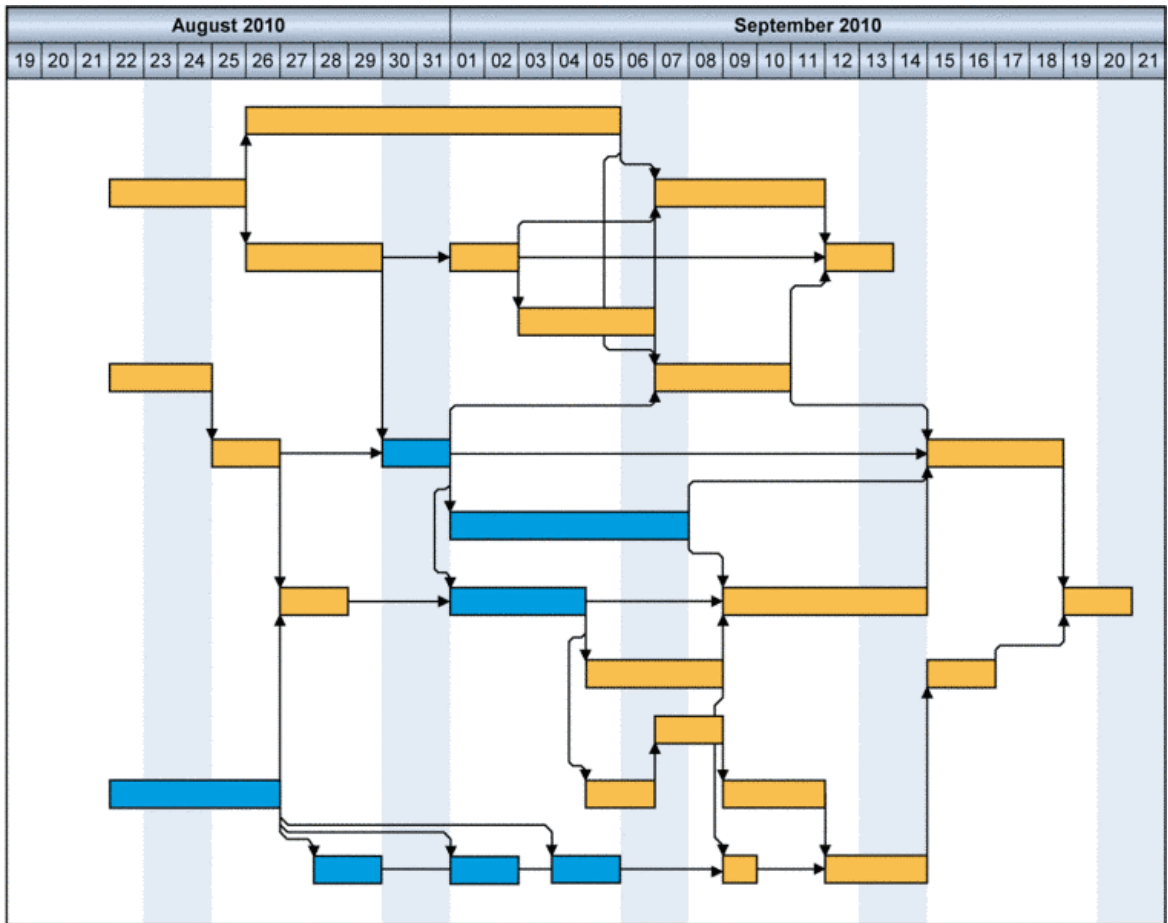
### ► Automated Layout

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.

## 106 Important Concepts: Links



## 3.16 Localization of Text Output

The **VcTextEntrySupplying** event allows to replace all items in the names of the months and days that appear during runtime in order to, for example, translate them into a different language.

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

### Example Code VB.NET

```
VcGanttASP1.TextEntrySupplyingEventEnabled = True
```

### Example Code C#

```
vcGanttASP1.TextEntrySupplyingEventEnabled = true;
```

Then capture the **VcTextEntrySupplying** event and specify the text you want to have appear.

### Example Code VB.NET

```
Private Sub VcGantt1_VcTextEntrySupplying(ByVal sender As Object, ByVal
e As NETRONIC.XGantt.VcTextEntrySupplyingEventArgs) Handles
VcGantt1.VcTextEntrySupplying

    Select Case e.ControlIndex
        Case VcTextEntryIndex.vcTXERibCW
            e.Text = "CW"
        Case VcTextEntryIndex.vcTXERibDay0
            e.Text = "Mo"
        Case VcTextEntryIndex.vcTXERibMon8
            e.Text = "September"
        Case VcTextEntryIndex.vcTXERibQuar3
            e.Text = "Quarter 3"
    End Select
End Sub
```

### Example Code C#

```
private void VcGantt1_VcTextEntrySupplying(object sender,
NETRONIC.XGantt.VcTextEntrySupplyingEventArgs e)
{
    switch(e.ControlIndex)
    {
        case VcTextEntryIndex.vcTXERibCW:
            e.Text = "CW";
            break;
        case VcTextEntryIndex.vcTXERibDay0:
            e.Text = "Mo";
            break;
        case VcTextEntryIndex.vcTXERibMon8:
            e.Text = "September";
            break;
    }
}
```

## 108 Important Concepts: Localization of Text Output

```
    case VcTextEntryIndex.vcTXERibQuar3:  
        e.Text = "Quarter 3";  
        break;  
    }  
}
```

## 3.17 Maps

Maps are used to set certain properties in dependence of 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 - logically - consist of at least two columns. One contains the keys, the other one contains the values. One key is assigned to one value. If there are more than two columns, more than one value is assigned to one key.


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

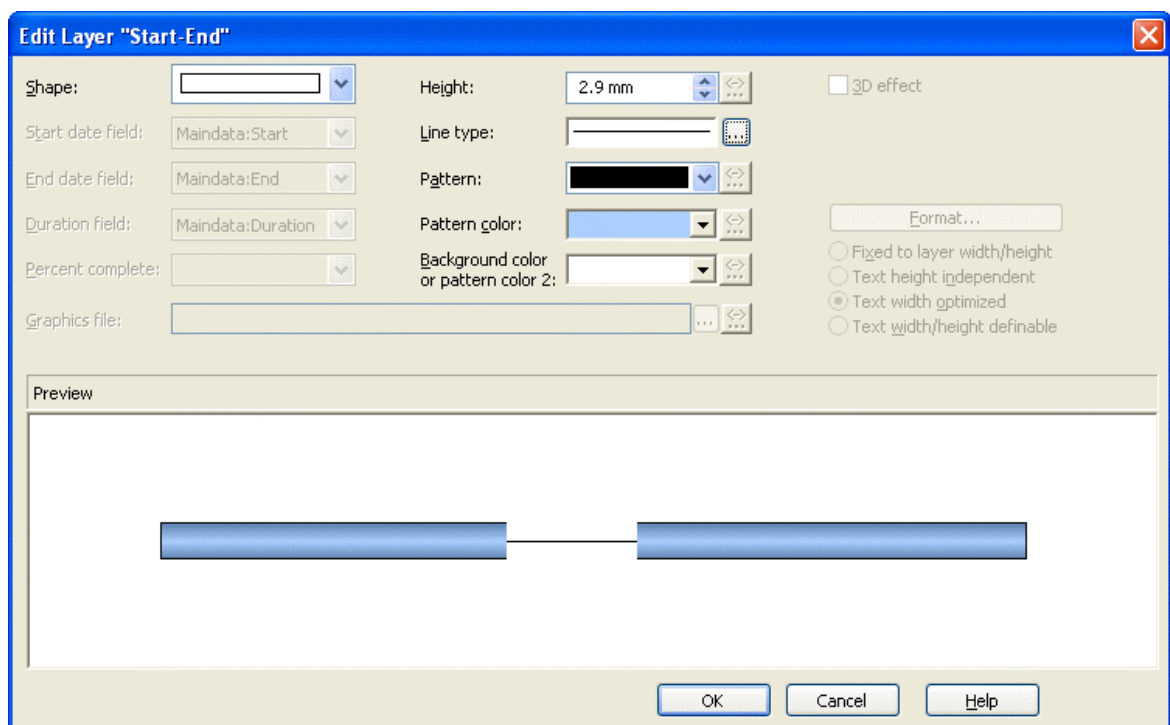
The principle is, that a field value of an object is compared to the key of the map. If they are equal, the map value(s) will apply.

By using filters, you can also specify ranges of values as keys.

### ► Example: Background color of layers

In the below example, the background color of a layer will be assigned in dependence on the node data via a map.

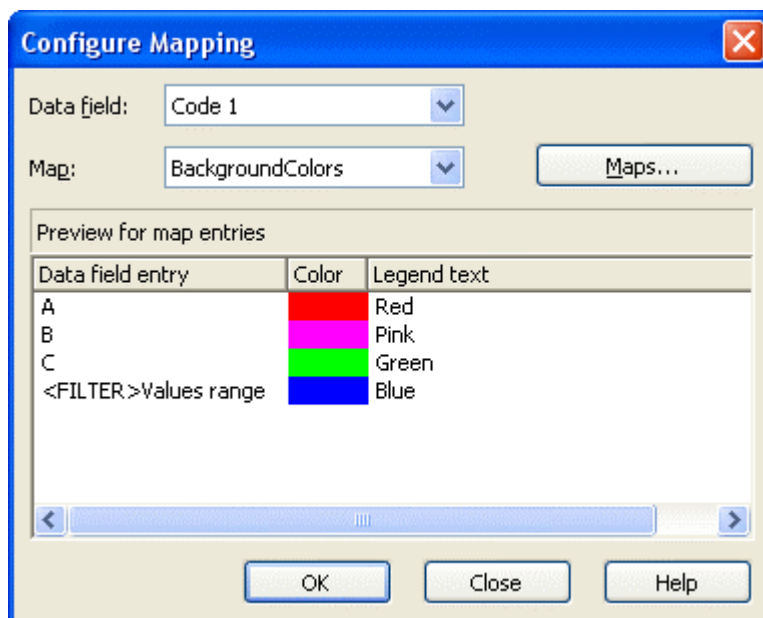
In the **Edit Layer** dialog box, please click on the  button near the **Background color** field .



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

### ► **Configuring 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, you can create a filter for this range of values which you select 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.



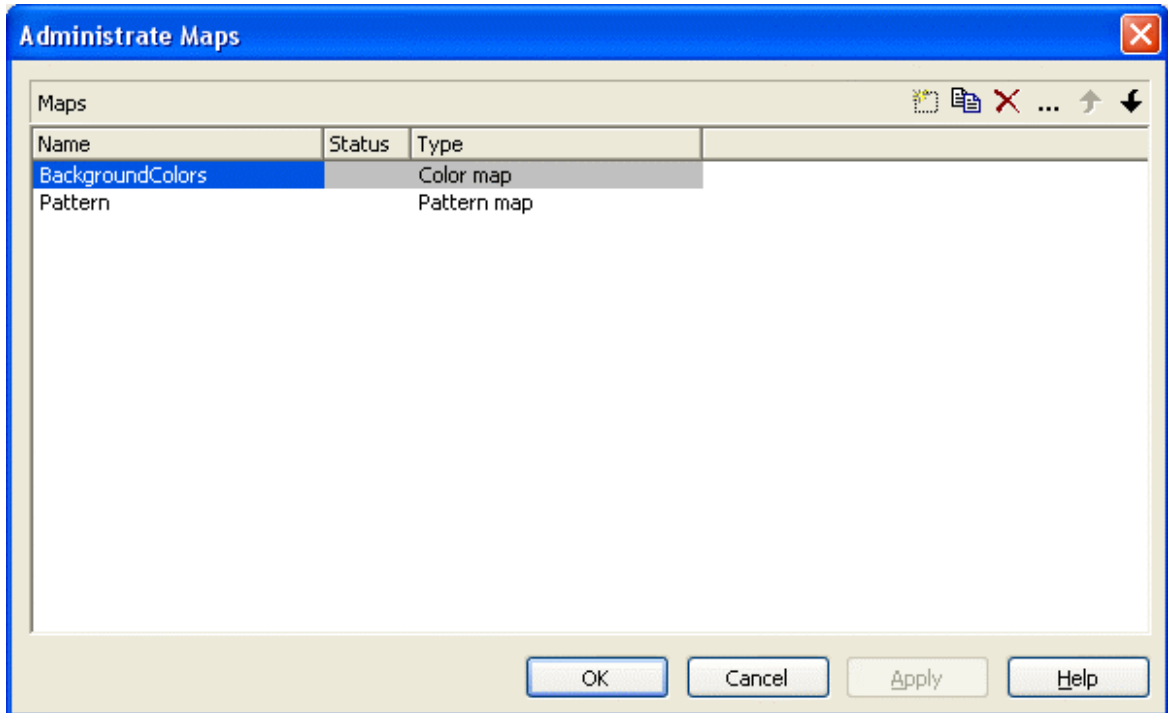
To configure a mapping, To configure a mapping, please select a node **Data field** at the top of the dialog, the values of which shall be compared to the key values of the map. From the field below, select an appropriate map **Map**. (Only those maps are selectable which match the attribute selected 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, its 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.

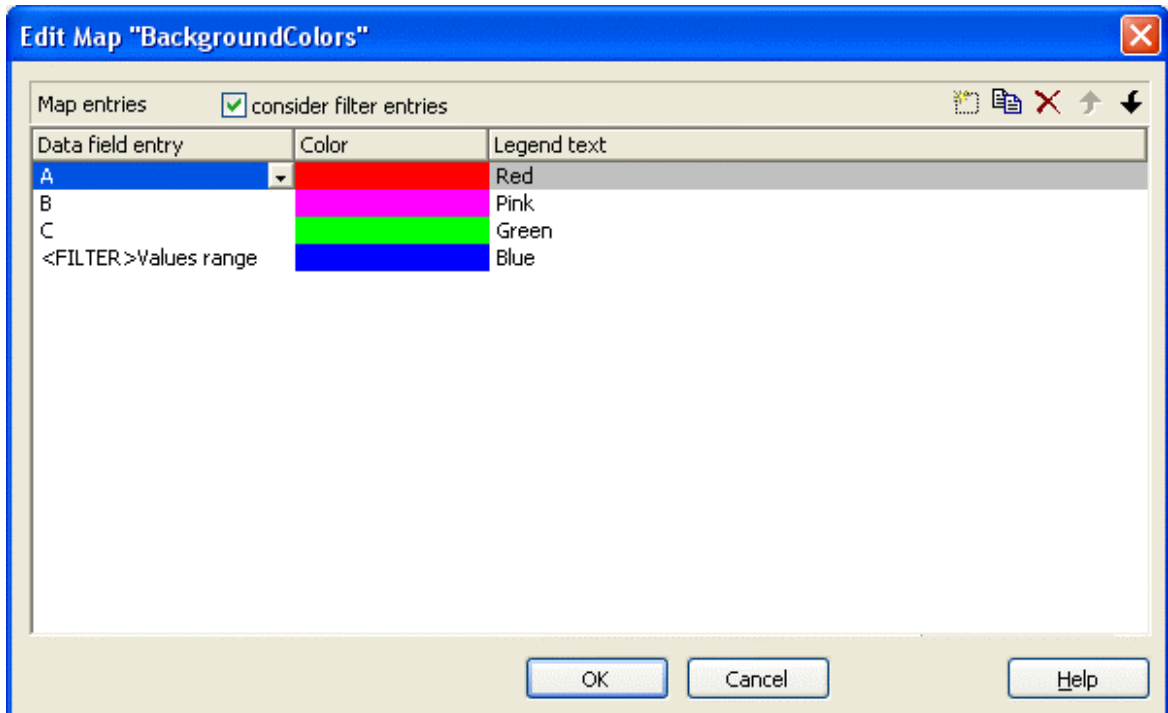


### ► Editing Maps

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



## 112 Important Concepts: Maps



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.

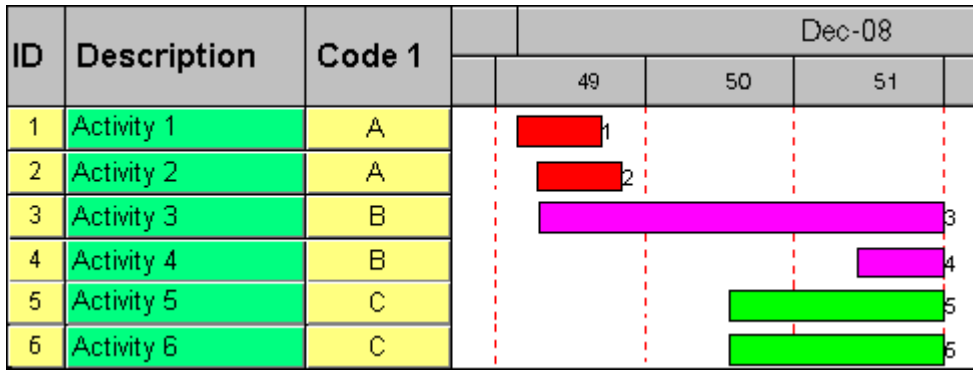
By the buttons right-hand at the top you can create, copy or delete keys (map entries) or modify their position in the table.

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 a range of values.

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 the activities of the field value = "A" are displayed in red, the activities of the field value = "B" are displayed in pink, etc. The default background color is gray. The latter is used for activities that have no data field value or that have filed value which 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 adjust the map during runtime using VcMap methods, which lets the user modify your default settings via a dialog designed by yourself.

---

## 3.18 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 multiState 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 field. 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 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 VcGantt property **InplaceEditingAllowed**).
- 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).

### ► **Instructions for Programming**

The graphics files are best placed in a directory of their own below the directory of the web application. The names of the graphics files in the graphics map can then be set without specifying a path. Beside, by the VcGantt property **FilePath** VARCHART needs to be told where to find the graphic files.

Sample code in C#:

```
VcGantt1.FilePath = this.Page.MapPath("GraphicsFileFolder");
```

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.

## 3.19 Node (Activity)

A node (activity) represents a record of the Maindata table. Nodes can be created via the programming interface by the method **InsertNodeRecord**. They can be deleted by **DeleteNodeRecord**.

### ► Further Settings to Nodes

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

- The data fields that the data of start, finish, and duration of interactively created nodes are to be stored to.
- 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 a calendar is required for an individual node, you can set a data field to store the name of the calendar.
- Whether a user is enabled to move several marked nodes at a time.
- Whether a marked node is enabled to be moved as a whole, that is, with all its layers.

### ► Events

You can react to the below events:

- VcNodeLeftClicking
- VcNodesMarked
- VcNodesMarking

## 3.20 Resource Scheduler

The ResourceScheduler2 is a substantial enhancement of ResourceScheduler1 (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

## 118 Important Concepts: Resource Scheduler

<b>Task Table</b>	
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

<b>Operations Table</b>	
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 preparation time
OperationResultSelectedTimingResourceIDFieldIndex	Determined ID of the timing resource

<b>Operations Table</b>	
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
OperationWorkInProgressFieldIndex	Degree of completion of the operation

<b>Resourcen Table</b>	
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

<b>Assignment Table</b>	
AssignmentDataTableName	Name of the assignment table
AssignmentIsResultFieldIndex	Was the data record generated by the scheduling procedure?
AssignmentIsVisibleFieldIndex	Should the assignment be visible in



## 120 Important Concepts: Resource Scheduler

Assignment Table	
	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 and Methods	
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**.

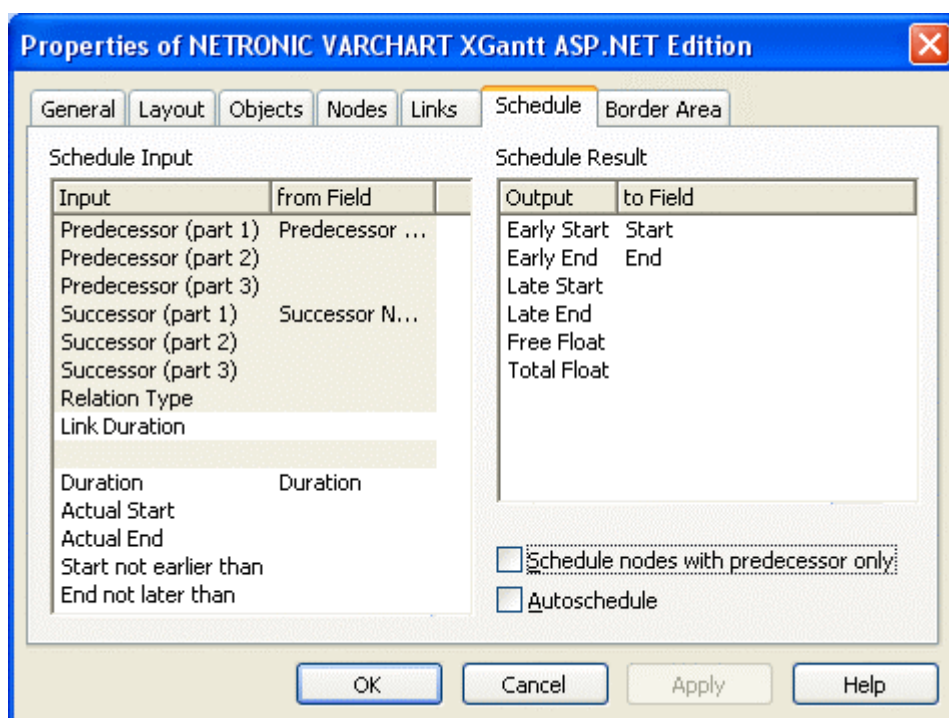
## 3.21 Schedule

You can perform simple date calculations by using the VARCHART XGantt scheduler. The start and end dates of the project 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 want to use for the input (**Schedule Input**) and 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 is the durations of the activities, their logical dependencies and the project start. Those informations are used to calculate the early/late start and end dates plus 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 that were made on the **Links** property page.



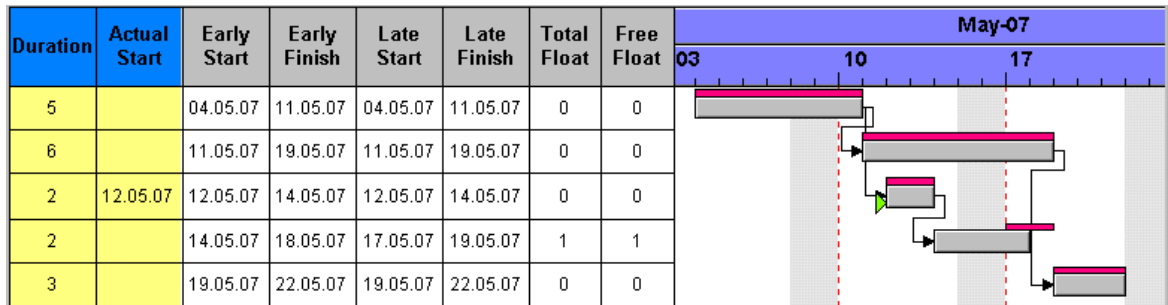
The results are stored to data fields of the interface. Available results are: **Early Start**, **Early Finish**, **Late Start**, **Late Finish**, **Total Float** and **Free Float**. To each of the results you can assign a field from the list of fields specified in the data definition. The below examples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

## 122 Important Concepts: Schedule

### Example Code VB.NET

```
VcGantt1.ScheduleProject ("04.05.2007", "11.05.2007")
```

The settings illustrated above would give the following graphical display:

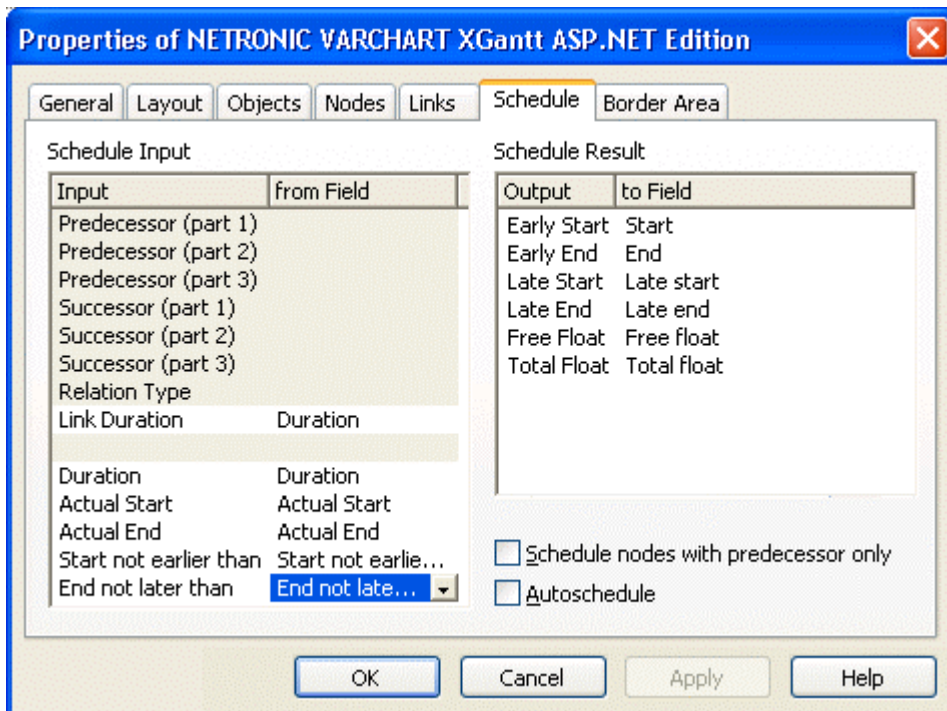


In this example, the early and late dates are both shown as layers.

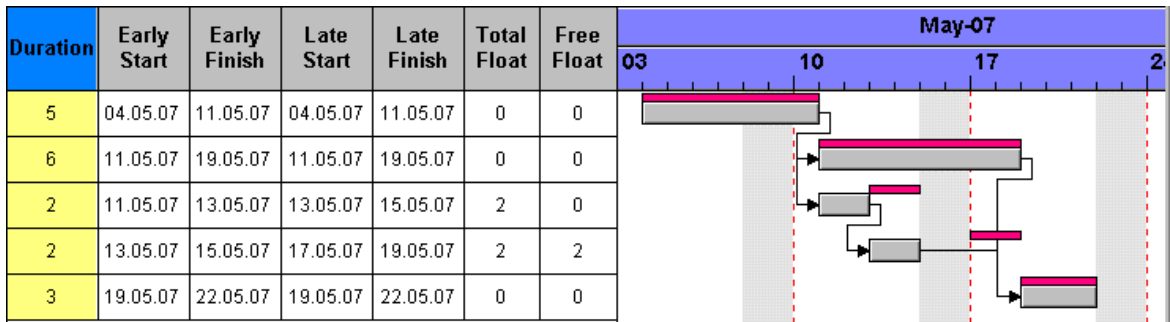
There are further possibilities to manipulate VARCHART XGantt scheduler's date calculations.

1. You can specify 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** expressions by defining a field from the data definition for each respective value in the left-hand table on the **Schedule** property page.

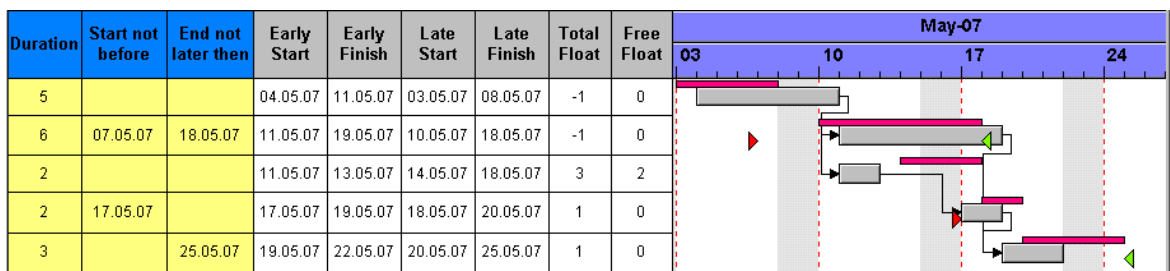
The below diagram shows the settings that were made for the example following:



By setting the actual start of an activity, the early and late dates are also fixed. In the following example, the actual start date set is marked by a green triangle.



Using the expressions **Start not before** and **End not later than** may or may not have an effect. In the following example, the date limits are marked by red and green triangles. Some do not have any effect on the date calculation, although the end date restriction of the second activity means that a negative float has been calculated for the first two.

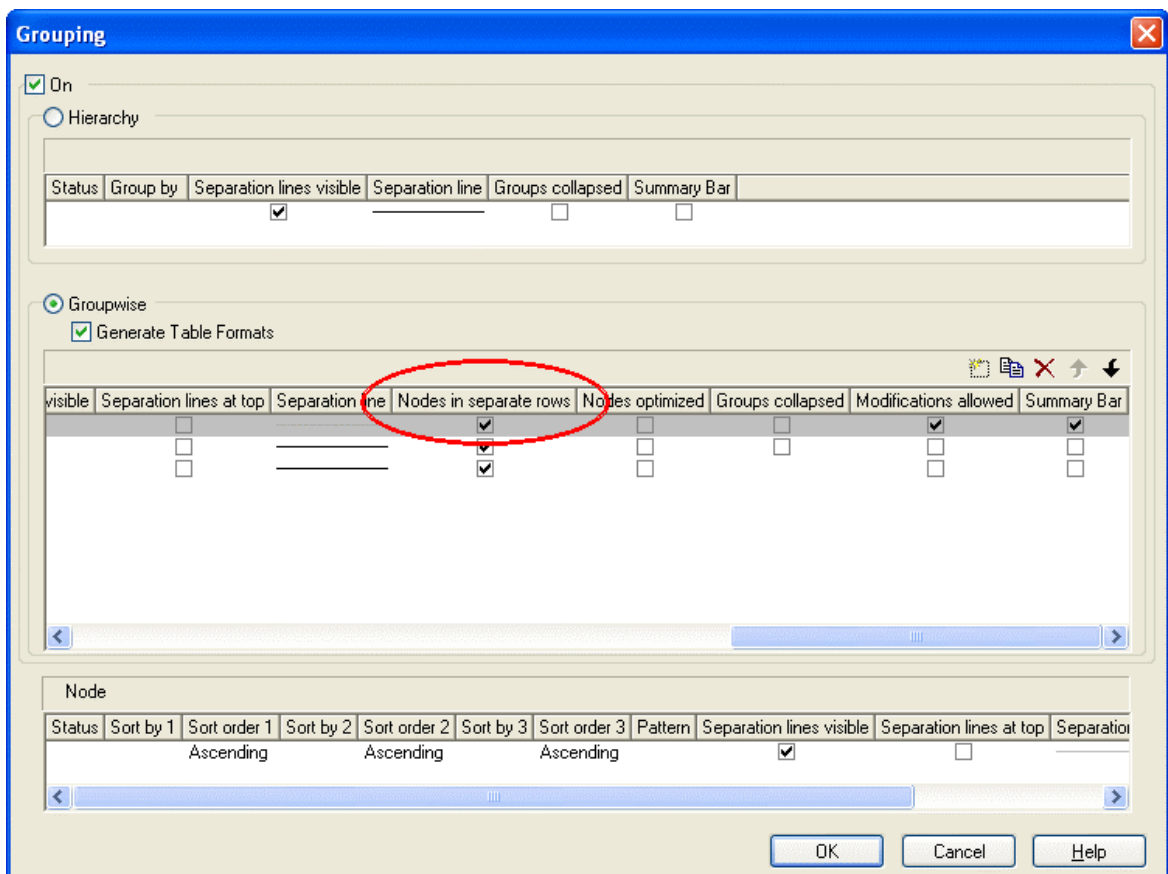


## 3.22 Sorting

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 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 **VcGroupLevel-Layout.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

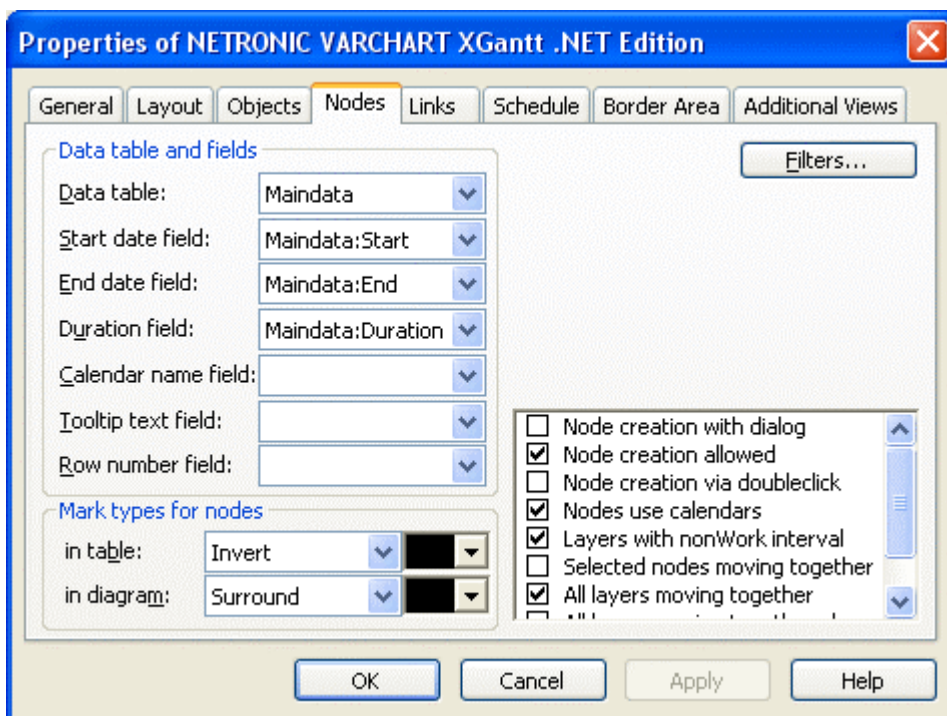
addition, you can select an ascending or a descending sorting order for each of the data fields.

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.

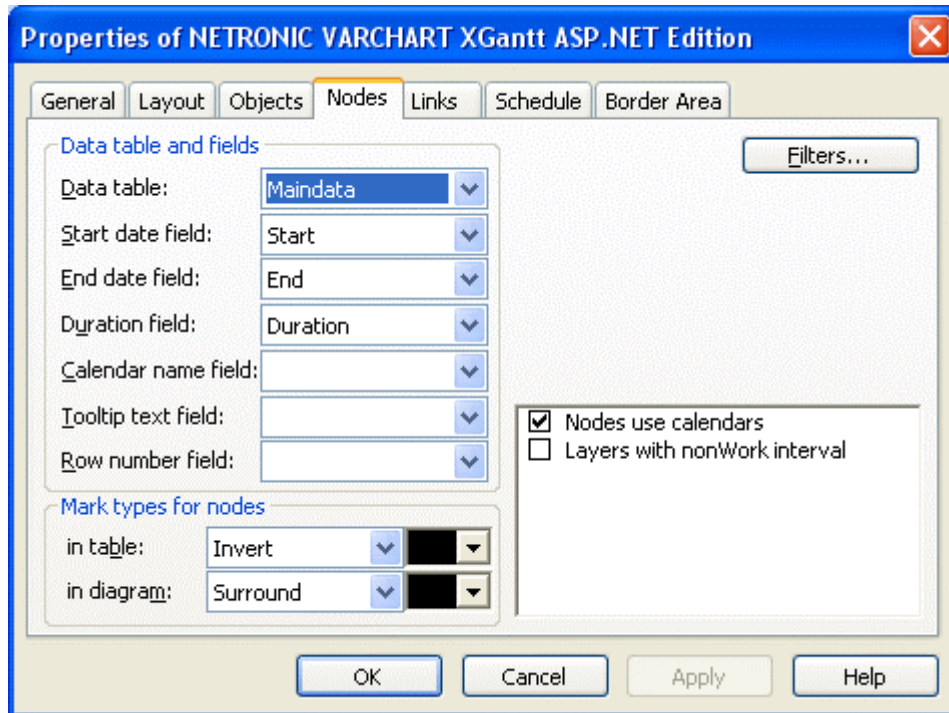
Further sorting options can be set on the **Nodes** property page:



- 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.
- Further sorting options can be set on the **Nodes** property page:



## 126 Important Concepts: Sorting

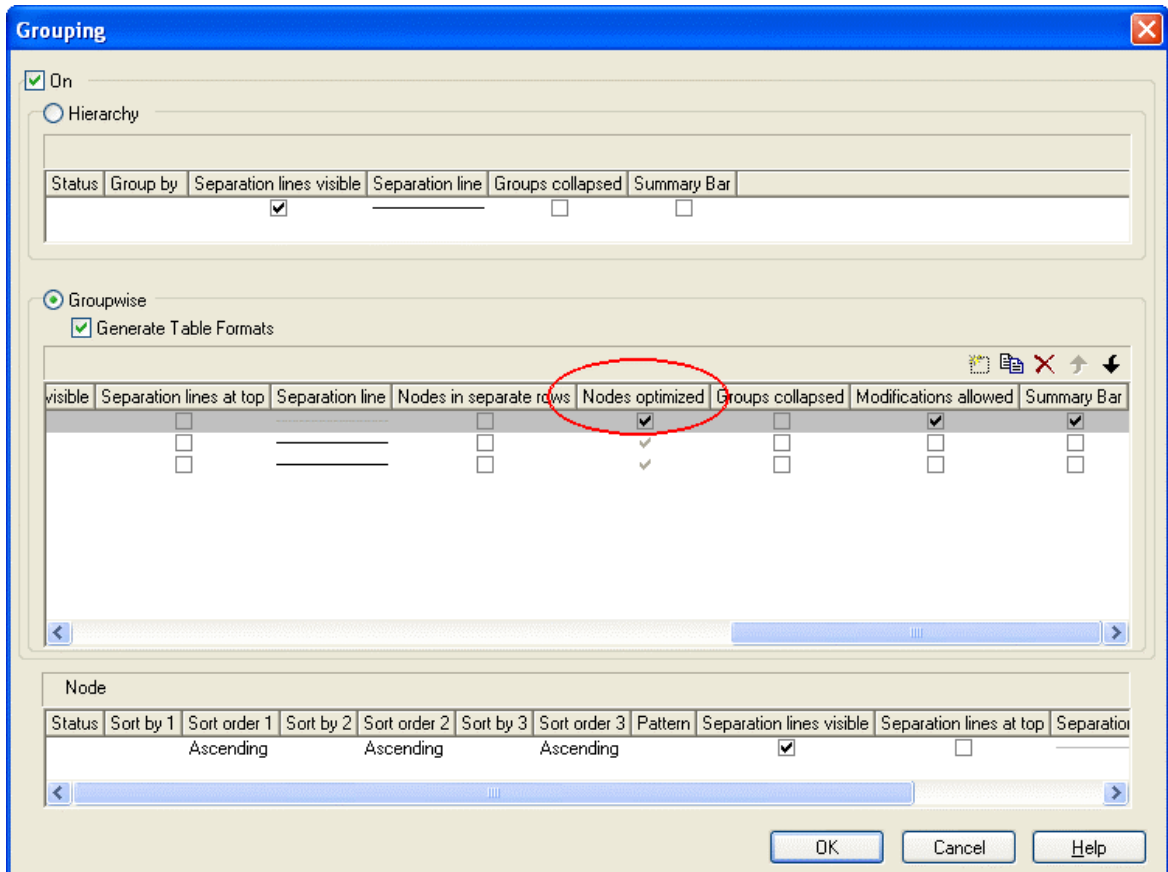


- 
- You can determine a data field to store the row number of each activity. The row number field will not be updated until you have saved the data via the **Save As** method.

### 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:

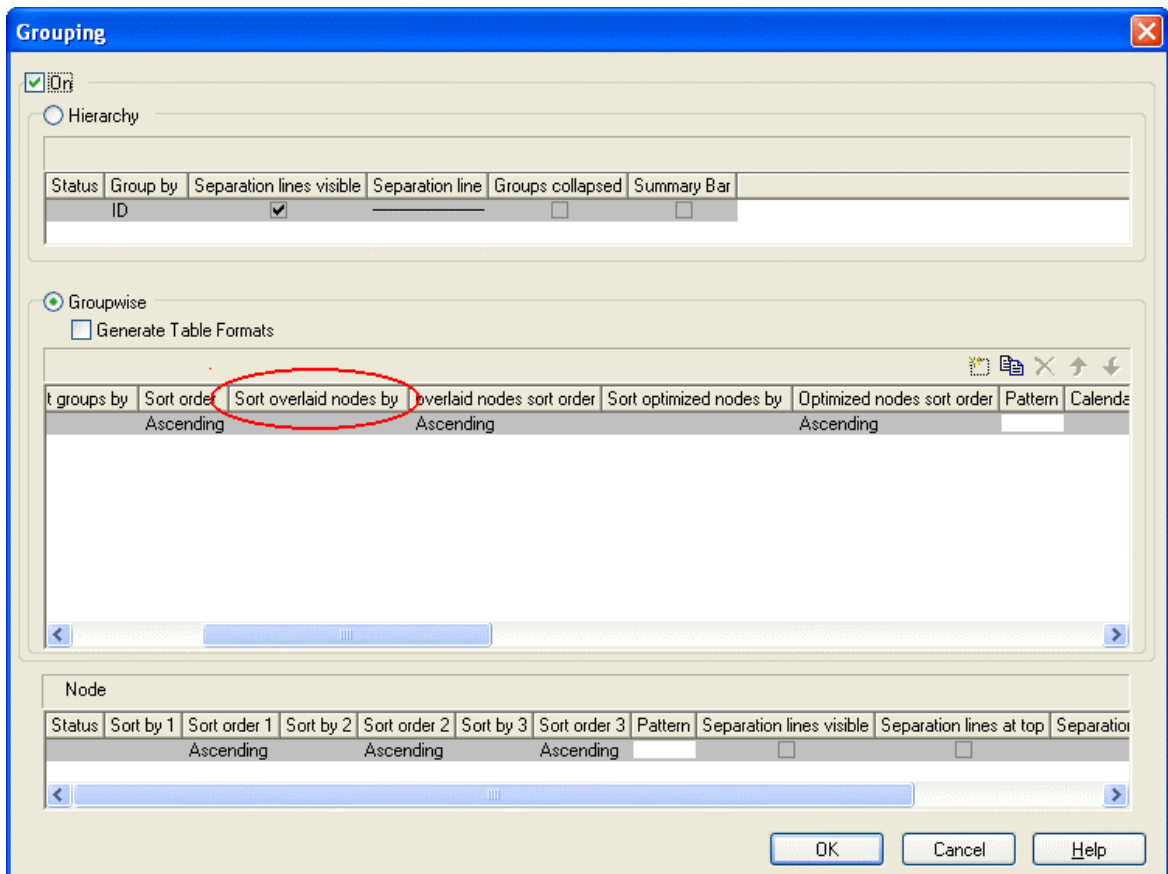


You can deactivate this check box which will entail the nodes of a row being displayed as overlapping. You can alternatively set this feature by the API property **VcGroup.NodesArrangedOptimized**.

The drawing priority of the nodes you can set by the field **Sort overlapping nodes by**:



## 128 Important Concepts: Sorting

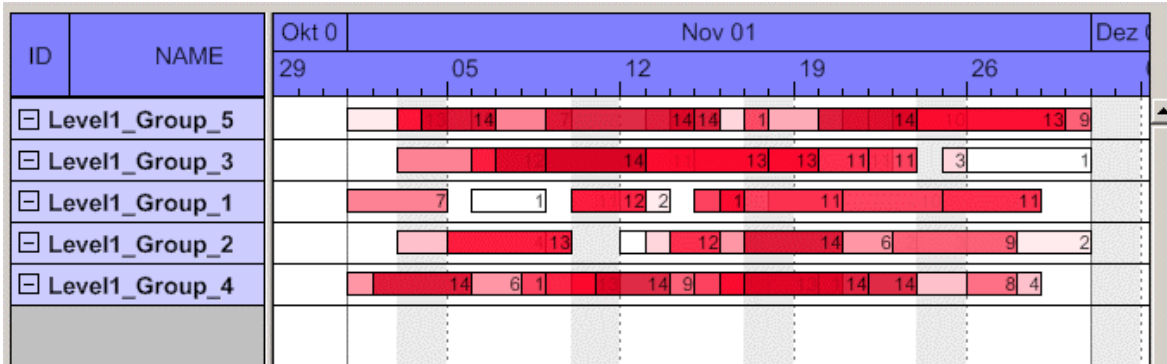


In analogy to overlapping nodes, you can sort optimized nodes by the field **Sort optimized nodes by**.

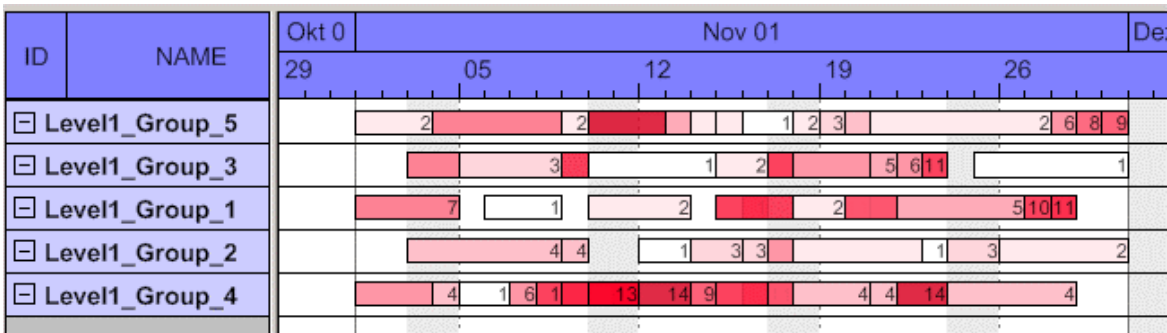
If you do not set a sorting priority, the nodes by default will be displayed in the order of their date and duration, the latest and shortest ones being drawn on top of the earlier and longer ones. The drawing priority can also be set by the API properties **VcLevelLayout.OverlaidNodesSortDataFieldIndex** 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 **OverlaidNodesSortOrder** and **OptimizedNodesSortOrder**, respectively.

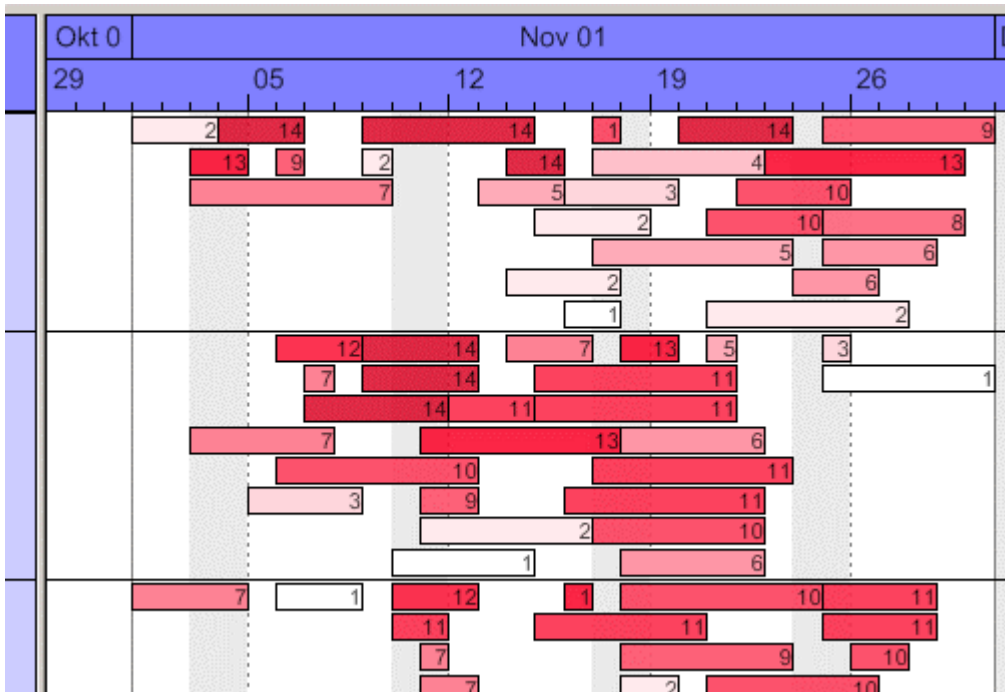
Below, some results of the settings are shown:



Overlay node arrangement showing an ascending drawing priority of dark nodes (dark nodes drawn on top of light nodes)

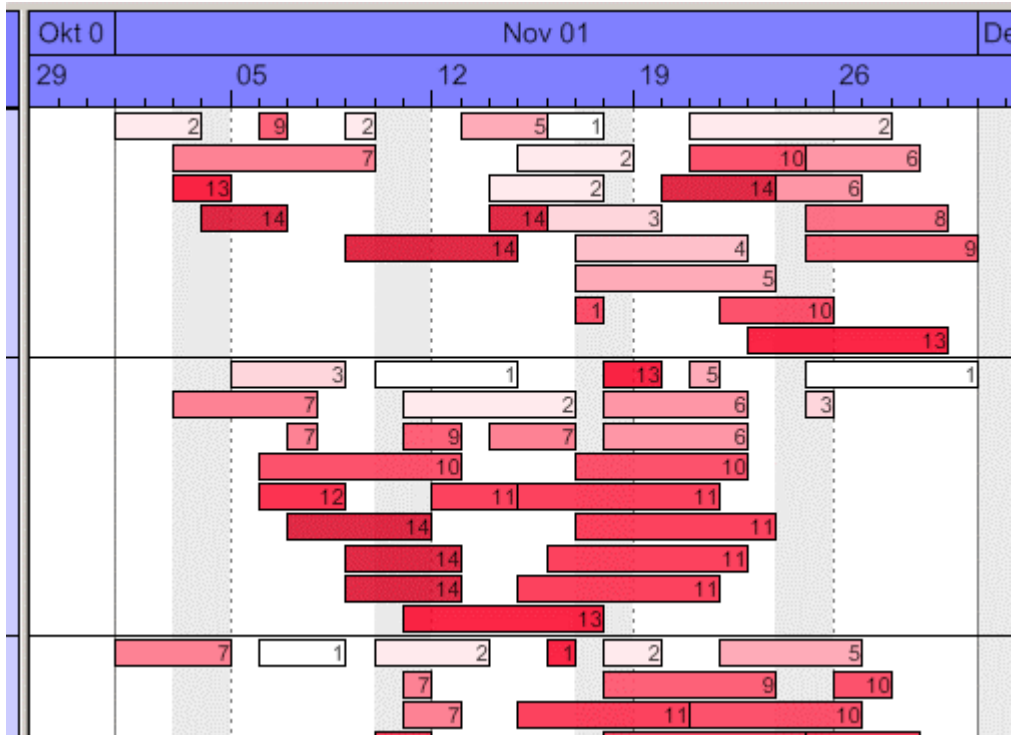


Overlay node arrangement showing a descending drawing priority of dark nodes (light nodes drawn on top of dark nodes)



Optimized node arrangement showing an ascending drawing priority of dark nodes (dark nodes drawn in the upper section of the row)

## 130 Important Concepts: Sorting



*Optimized node arrangement showing an descending drawing priority of dark nodes (light nodes drawn in the upper section of the row)*

## 3.23 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 that 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	<input type="checkbox"/> SW Development	02.09.98
1.2	<input checked="" type="checkbox"/> Design&Concept	02.09.98
1.3	<input type="checkbox"/> Coding	09.09.98
1.3.1	Phase A (DB)	09.09.98
1.3.2	Phase B (GUI)	15.09.98
1.4	<input checked="" type="checkbox"/> Testing	17.09.98
1.5	Sales & Marketing	05.09.98
1.6	Delivery	24.09.98
1.7	Final Party	

## 132 Important Concepts: Table

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

## 3.24 Time Scale

Above the diagram area, the time scale is displayed. You can display one or more annotated ribbons of the time scale below the diagram area, too (see **Edit Time Scale Section** dialog box, **Ribbons, Position**). The appropriate timescale for the time period displayed can be selected.

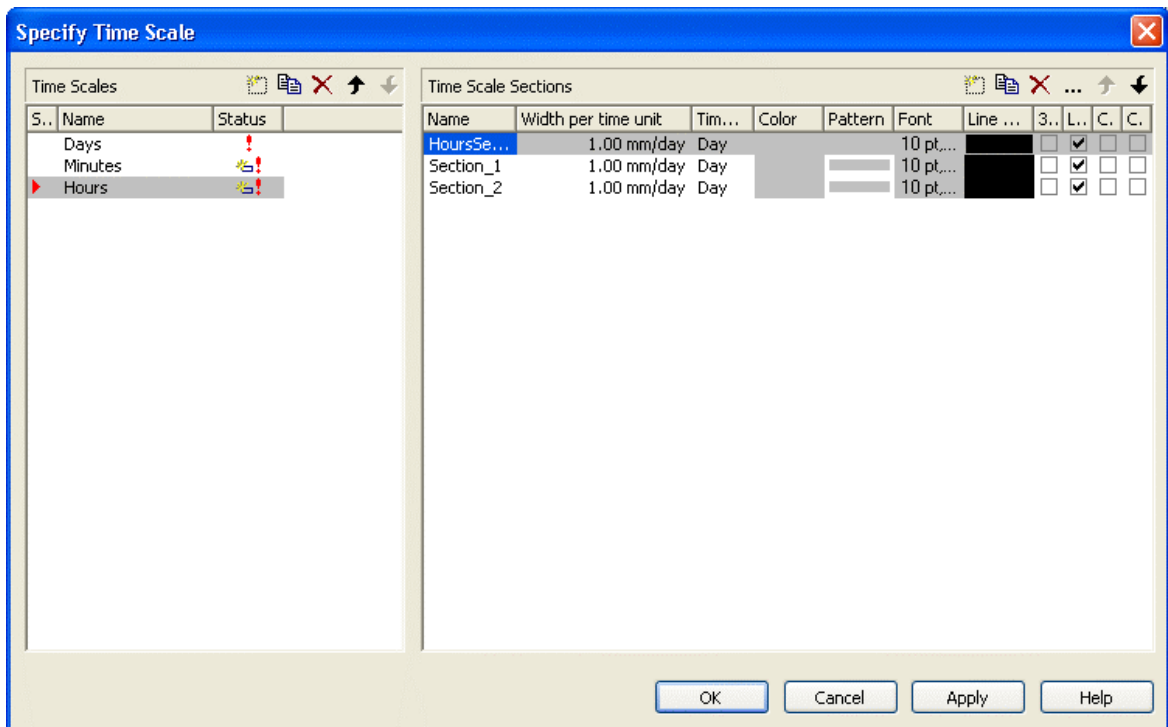
You can divide the time scale into sections, specifying the number of sections to be displayed, their ranges and scales. Project phases that you want to plan in particular detail can be displayed in a more "magnified" form than the other phases: Perhaps you wish to present your project plans for the immediate future in more detail than your plans for the distant future or past, enabling you to concentrate on the project phases that are currently of most interest to you and shift the focus as your project progresses. Or you can start with a general project overview and continue your planning in increasing detail.

Nov-07					Dec-07		
CW 45	CW 46	CW 47	CW 48		CW 49	CW 50	CW 51

There is a whole range of options for designing the timescale, sections and 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 transparent, you can define grids, e.g. a day or week grid, for each section.

You can select the timescale you want to use for your diagram (**Selected**) from the range of preset timescales offered in the **Specify Time Scale** dialog. The time scales differ from one another in the width of the time unit and the ribbons.

## 134 Important Concepts: Time Scale



It is possible to change the selection during runtime.

### ► Specifying start and end dates of the time scale

The default start and end dates of the time scale are specified on the **General** property page (**Project Start** and **Project End**). At runtime, fit this value to the current data via the **TimeScaleStart** property or the **OptimizeTimeScaleStartEnd** method. The date format is "DD.MM.YYYY;hh:mm:ss".

**Note:** The end date is not included. If you specify **TimeScaleEnd** = "31.12.02" for example, the last day displayed will be the 30.12.02.

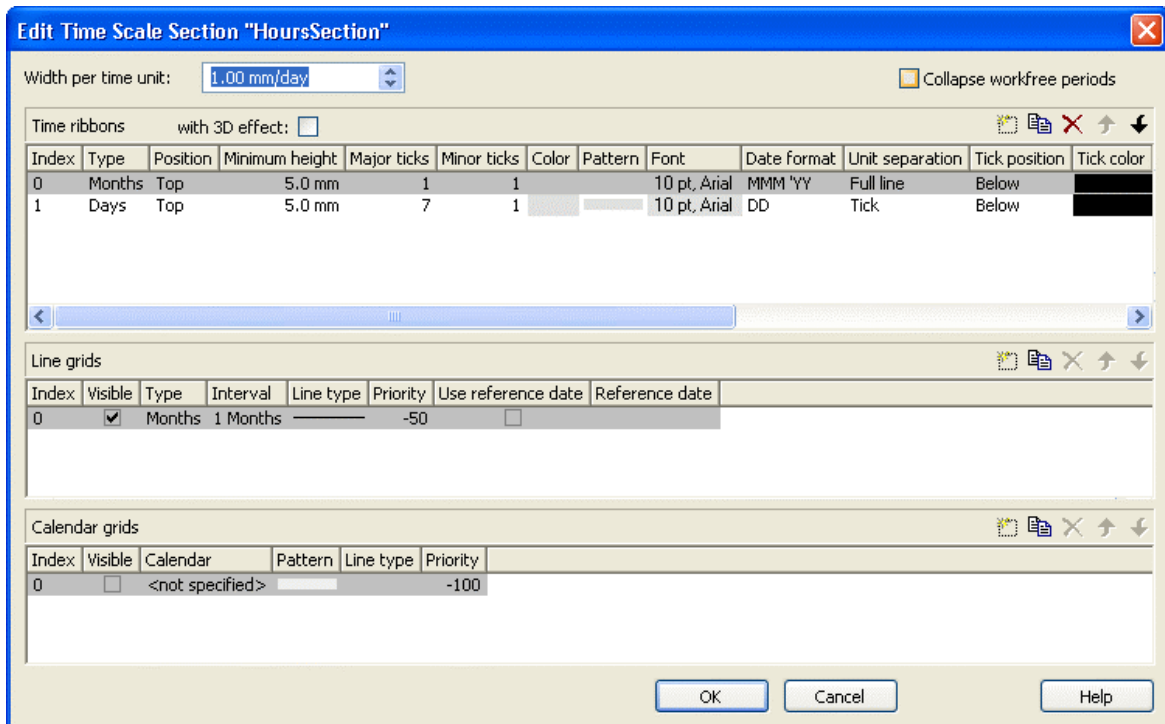
### ► Sections

You can split the time scale into sections to highlight certain planning phases and specify different ribbons for each section. In the **Specify Time Scale** dialog you can set the **Time unit** and the **Width per time unit** individually for each section. Also, for each section you can define a separate color, font, pattern 3D effect, line grid and calendar grid, and specify whether workfree periods are suppressed.

When you select a line grid, variable vertical grid lines are displayed in the appropriate section.

When you use a calendar, a predefined calendar grid can be displayed in the appropriate section where workfree periods are marked by colored vertical areas.

From the **Specify Time Scale** dialog you can reach the **Edit Time Scale Section** dialog box where you can edit each of the ribbons and grids of each section.



### ► Width per time unit

The unit is the smallest unit the time scale is divided to. Possible unit widths are: second, minute, hour and day. You can specify the unit in the **Specify Time Scale**.

You can specify the **Width per time unit** in millimetres per unit width in steps of 100th of a millimetre per unit width. The minimum width you can assign to the time unit is 0.01 mm.

### ► Ribbons

Ribbons serve the purpose of annotating the timescale. Each section may be assigned several ribbons (e.g. one with a monthly and a second with a daily scale). For each ribbon you can specify the **Position**, i. e. whether it is to be displayed or not and whether it is to be displayed at the top or at the bottom of the diagram. Furthermore, you can specify for each ribbon the following: the type, minimal height, major and minor ticks, color, font, date format, unit separation, alignment, serial annotation, reference date, calendar.

Which date formats are available for a particular ribbon depends on the type of ribbon selected.

To compose the date you can use the following tokens:



## 136 Important Concepts: Time Scale

- 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
- 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: **TWWW - TM - TQ - YYYY**, 2. ribbon: **TD**

CW37 - September - Quarter 3 - 2007							
Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday	Monday

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 VB.NET

```
Private Sub VcGantt1_VcTextEntrySupplying(ByVal sender As Object, ByVal
e As NETRONIC.XGantt.VcTextEntrySupplyingEventArgs) Handles
VcGantt1.VcTextEntrySupplying

    Select Case e.ControlIndex
        Case VcTextEntryIndex.vcTXERibDay0
            e.Text = "Lundi"
        ...
        Case VcTextEntryIndex.vcTXERibCW
```

## 138 Important Concepts: Time Scale

```
        e.Text = "Semaine"  
    Case VcTextEntryIndex.vcTXERibMon8  
        e.Text = "Septembre"  
    Case VcTextEntryIndex.vcTXERibQuar3  
        e.Text = "3. Trimestre"  
    End Select  
End Sub
```

### Example Code C#

```
private void VcGantt1_VcTextEntrySupplying(object sender,  
NETRONIC.XGantt.VcTextEntrySupplyingEventArgs e)  
{  
    switch(e.ControlIndex)  
    {  
        case VcTextEntryIndex.vcTXERibDay0:  
            e.Text = "Lundi";  
            break;  
        ...  
        case VcTextEntryIndex.vcTXERibCW:  
            e.Text = "Semaine";  
            break;  
        case VcTextEntryIndex.vcTXERibMon8:  
            e.Text = "Septembre";  
            break;  
        case VcTextEntryIndex.vcTXERibQuar3:  
            e.Text = "3. Trimestre";  
            break;  
    }  
}
```

Semaine 37 - Septembre - 3. Trimestre - 2007							
Lundi	Mardi	Mercredi	Jeudi	Vendredi	Samedi	Dimanche	Lundi

---

## 3.25 Tooltips during Runtime

You can use tooltips to provide information on the object that was touched by the mouse. By the event **VcToolTipTextSupplying** you can edit the texts of all the tooltips that appear at run time, for example in order to translate them into different languages or suppress them.

To activate the event, activate the check box **VcToolTipTextSupplying events** on the **General** property page.

Or set the property **ToolTipTextSupplyingEventEnabled** to **True**.

---

## 3.26 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 VARCHAR 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 **VcDummyObject** 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.msp](http://www.microsoft.com/globaldev/getwr/steps/wrg_unicode.msp)). 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".

---

## **3.27 Viewer Metafile (\*.vmf)**

---

## 3.28 Writing PDF files

Writing PDF files is only possible if an appropriate PDF printing driver is available on your server. 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 for ASP.NET, a PDF printing driver has to fulfill the below requirements concerning controlling and print quality:

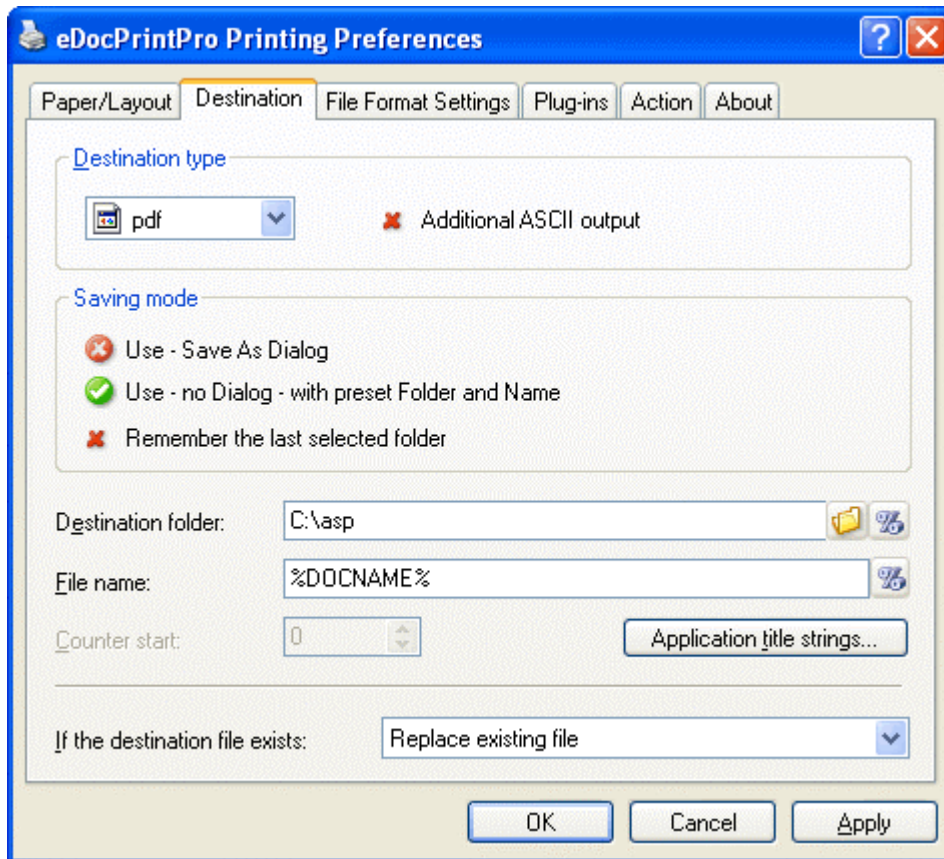
- As a basic principle, the driver needs to offer the option to switch off all runtime dialogs and message boxes, in particular dialogs for setting file names and paths.
- If file names and paths shall not be set until runtime and if this is only possible by modifying entries of the Windows registry, the permissions of the user account that enable the ASP.Net application to run, have to be set accordingly.
- As in terms of permissions ASP.NET applications run with their own user account, all settings of the account have to be adjustable, even if it has no desktop.
- 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](http://www.adobe.com)] and the free driver **eDocPrintPro** [[www.pdfprinter.at](http://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. In that dialog please 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. Also, files already existing should be replaced without a dialog. The required settings in **eDocPrintPro** look as follows:



- As ASP.Net applications usually run with an own user account, this account has to be granted writing permission for the target directory.
- In addition, the user account needs the print permission.
- 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"
```

The **eDocPrintPro** settings of the current user are stored to the registry by the below path:



## 144 Important Concepts: Writing PDF files

HKEY\_CURRENT\_USER\Software\MAYComputer\eDocPrintPro\eDocPrintPro

As of **eDocPrintPro** version 3.13 a user may define default settings that are valid for all user accounts including the ASP.NET user account. After the above settings were made for the current user it will suffice to copy them to the below path of the registry:

HKEY\_LOCAL\_MACHINE\SOFTWARE\MayComputer\eDocPrintPro\Def  
\eDocPrintPro

For further information concerning configuration and usage of **eDocPrintPro** please contact the producer.

---



---

## 4 Property Pages and Dialog Boxes


---

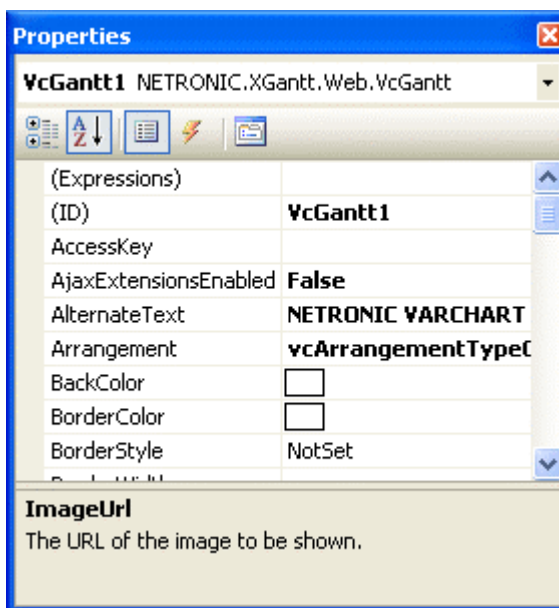
### 4.1 General Information

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.

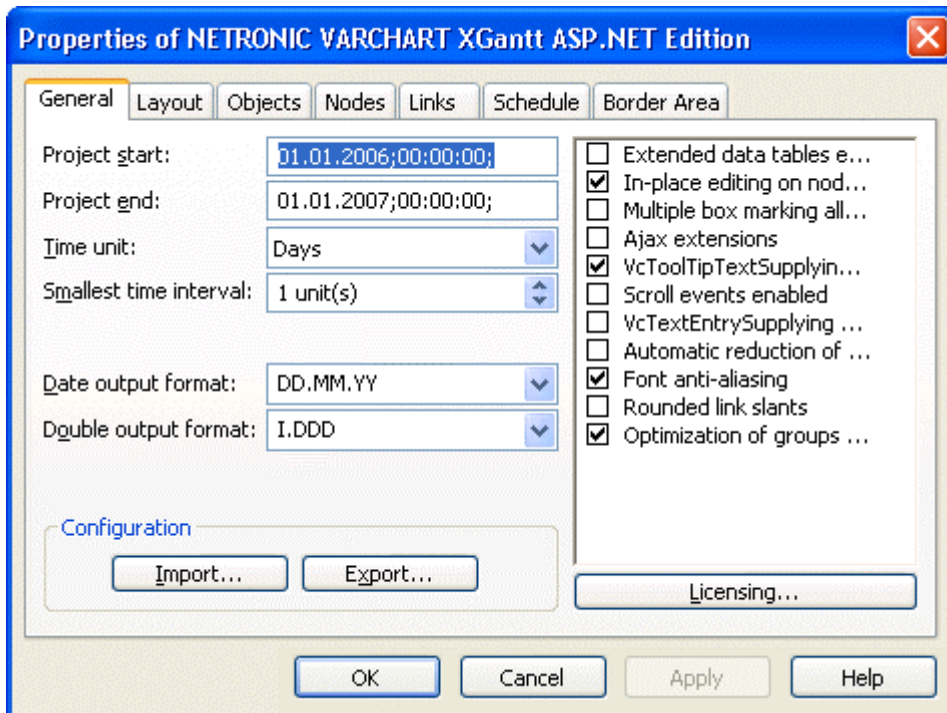
or

- 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 the general settings of VARCHART XGantt.

### Project start

Specify the default start date of the time scale. At run time, this value can be adapted to the current data by the property **TimeScaleStart** or by the method **OptimizeTimeScaleStartEnd**. The date format is "DD.MM.YYYY;hh:mm:ss;". This feature can also be set by the property **VcScheduler.ScheduledProjectStartDate**

### Project end

Specify the default end date of the time scale. At run time, this value can be adapted to the current data by using the property **TimeScaleEnd** or the method **OptimizeTimeScaleStartEnd**. The date format is "DD.MM.YYYY;hh:mm:ss;". This feature can also be set by the property **VcScheduler.ScheduledProjectEndDate**

**Note:** The actual end date is not included. If you set **TimeScaleEnd** = "31.12.09" for example, the last day displayed will be December, 30st 2009.

## Time unit

Select the time unit for your diagram. The value entered here will be used to calculate the duration (see Chapter "Important Concepts: Layer") and for the interactive modification and moving of the nodes in the diagram.

**Example:** If you select the time unit "Days" here, the nodes can only be moved in as many day steps as specified in the field **Smallest time interval**.

This feature can also be set by the property **VcGanttASP.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 in half-hour steps. This way a bar or layer will "snap" at a full hour and at half an hour.

This feature can also be set by the property **VcGantt.TimeUnitsPerStep**.

## Date output format

From the combo box, select a format for your date output, or define a format.

The format will also apply to the dialogs at runtime.

This feature can also be set by the property **VcGanttASP.DateOutput-Format**.

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 **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

## 148 The "General" Property Page

WW:	two-digit figure for the number of the calendar week: 01-53
TW:	text for "calendar week" (adjustable by using the event <b>VcTextEntrySupplying</b> )
Q:	one-digit figure for the quarter: 1-4
TQ:	name of quarter (adjustable by using the event <b>VcTextEntrySupplying</b> )
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 <b>VcTextEntrySupplying</b> )
TH:	"am" or "pm" (adjustable by using the event <b>VcTextEntrySupplying</b> )
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** do not need '\' as a prefix.

### Double output format

From the select box, please choose 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 digits) and **\$ I,III.DD** or **I.III,DD €** (two-digit currency).

This feature can also be set by the property **VcGanttASP.DoubleOutputFormat** .

## Configuration

You can store the settings of the property pages to a configuration outside your project at any time, and load them when required. This is very useful if you want to use previous settings again or you need the settings for different projects.

A configuration consists of two files of the same name that have different extensions, an ini- and an IFD file, which both are indispensable.

You can specify either a local file including the path or a URL.

An 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. If you specify a URL for configuration already at design time, the INI and IFD files will be downloaded, but they will be added to the project as a resource and be used at run time rather than loading the files directly.

### How to save your current configuration:

Click on the **Export** button and enter a name for the INI file. An IFD file of the same name will be created automatically.

### How to load a saved configuration:

Click on the **Import** button and select the file needed.

## Extended data tables enabled

If you tick this box you can create and use up to 99 data tables, instead of merely the two default tables **Main data** and **Relations**. This option can also be set by the property **VcGanttASP.ExtendedDataTablesEnabled**.

## 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 **VcGanttASP.InPlaceEditingOnNodesInTableEnabled**.

If to certain data fields in-place editing shall not be permitted, please don't select the option **editable** in the data definition.

## Multiple box marking allowed

By ticking this box, the user can select several boxes at the same time by clicking on them without having to keep the CTRL-key pressed. This option is disabled by default.

This feature can also be set by the property **VcGanttASP.MultipleBox-MarkingAllowed**.

## Ajax extensions

Tick this option if Ajax extensions are to be switched on.

This feature can also be set by the property **VcGanttASP.AjaxExtensions-Enabled**.

## VcToolTipTextSupplying events

Tick this option if the event **VcToolTipTextSupplying** is to be activated. It also can be set by the **ToolTipTextSupplyingEventEnabled** property. The event **VcToolTipTextSupplying** lets you set the text strings to be displayed as tooltip texts with the objects.

## Scroll events enabled

By ticking this box, you may 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.

## VcTextEntrySupplying events

By ticking this box you can trigger the **VcTextEntrySupplying** event. This event lets you modify the texts of the names of days and months that occur during run time, for example for translation into different languages.

This feature can also be set by the property **VcGanttASP.TextEntry-SupplyingEventEnabled**.

## Automatic 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 **VcGanttASP.RowHeightReductionEnabled**.

## Font anti-aliasing

This option allows to set anti-aliasing to font characterst. Some fonts, especially non-Latin ones may lose clarity, so this option should be switched off in those cases. This feature can also be set by the property **VcGanttASP.FontAntiAliasingEnabled**.

## 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 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.

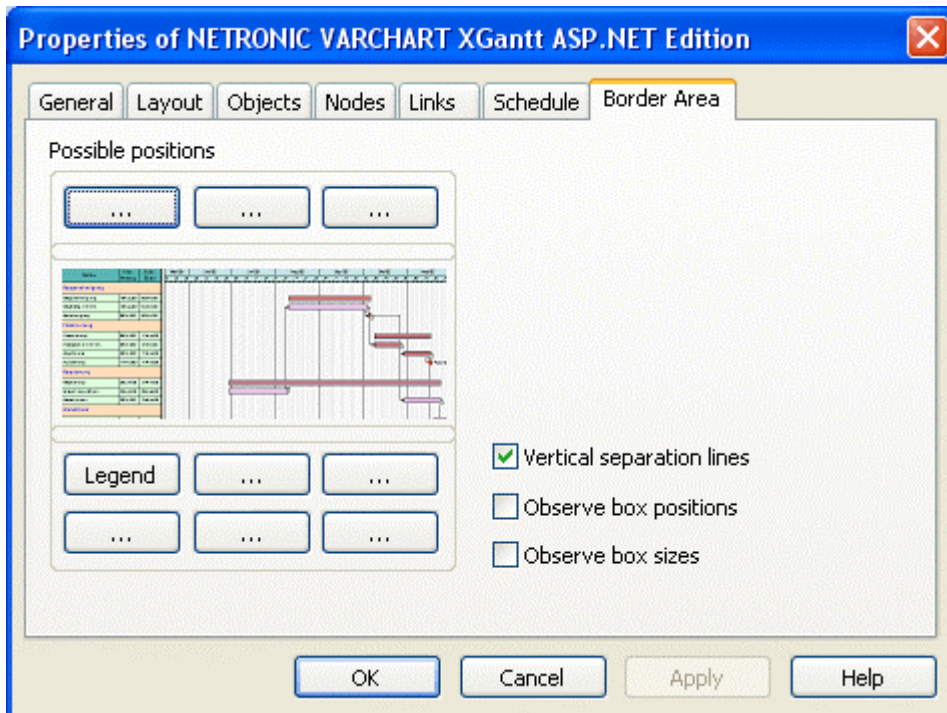
This feature can also be set by the **>b!VcGanttASP.GroupOptimizationOnInteractionsEnabled** property

## Licensing

Press this button to get to the **Licensing** dialog box. For further information see chapter **Licensing**.



## 4.3 The "Border Area" Property Page



### Possible positions

There are three areas above and six areas below the diagram which you can use 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 or below the diagram to get to 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.

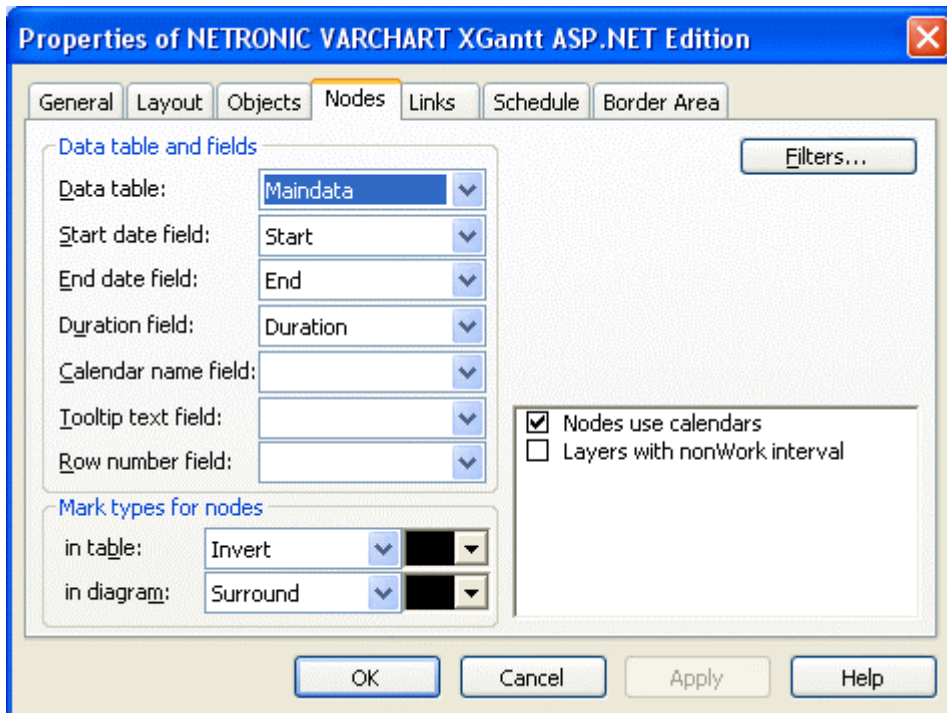
### Observe box position

Activate this check box, if the box positions are to be observed as exactly as possible. Alternatively, 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 observed as exactly as possible. The chart may be enlarged and/or the texts in the boxes may be clipped.

## 4.4 The "Nodes" Property Page



### Data table

Select the data table which shall be used for the representation of the nodes.

This feature can also be set by the property **VcGanttASP.NodesDataTableName**.

### Start date field

Please select the data field to store the start date of an interactively created node. Only date fields are offered in the combo box.

This feature can also be set by the property **VcGanttASP.NodeStartDateDataFieldIndex**.

### End date field

Please select the data field to store the finish of an interactively created node. Only date fields are offered in the combo box.

This feature can also be set by the property **VcGanttASP.NodeEndDateDataFieldIndex**.

## Duration field

Please select a data field to store the duration of an interactively created layer. Only numeric data fields are available.

This feature can also be set by the property **VcGanttASP.NodeDuration-DataFieldIndex**.

## 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 **Nodes use calendar** needs to be activated. Besides, the calendars need to have been created before loading the nodes.

This feature can also be set by the property **VcGanttASP.NodeCalendar-NameDataFieldIndex**.

## Row number field

Please select a data field which stores the row number of the node. This is only possible as long as no data has been loaded. 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 **VcGanttASP.NodeRow-NumberDataFieldIndex**.

## Tooltip text field

The data field specified here is only important for the VMF export. If you show a VMF file by the WebViewer software and there right-click on a node, the contents of the selected data field will be shown as a tooltip. No further settings are required.

To show tooltips in your application, activate the check box **VcToolTipText-Supplying events** on the **General** property page or set the VcGantt property **ToolTipTextSupplyingEventEnabled** = True and specify the data fields to be displayed in the **VcToolTipTextSupplying** event.

## Mark type for nodes in table

Use the left field to specify whether node marks are used in the table and, if desired, select the type of node marking from the list:

- No Mark

- Surround inside
- Invert
- Pickmarks inside

The field to the right lets you select a color for the marking type.

## **Mark type for nodes in diagram**

Use the left field to specify whether node marks are used in the diagram and, if desired, select the type of node marking from the list:

- No Mark
- Surround
- Surround inside
- Invert
- Pickmarks
- Pickmarks inside

The field to the right lets you select a color for the marking type.

## **Filters**

This button lets you open the **Administrate Filters** dialog box. The filter settings that pre-select the nodes can only be entered at runtime by the property **ActiveNodeFilter** of the object **VcGantt**.

## **Nodes use calendars**

Tick this box to assign calendars to the nodes. Assigning calendars to nodes has the following effects: The starts and ends of the activities are not positioned on workfree days. The workfree periods are considered when calculating the duration of the activities. Currently, the default is a five-day calendar ("BaseCalendar").

This feature can also be set by the property **VcGanttASP.NodesUse-Calendars**.

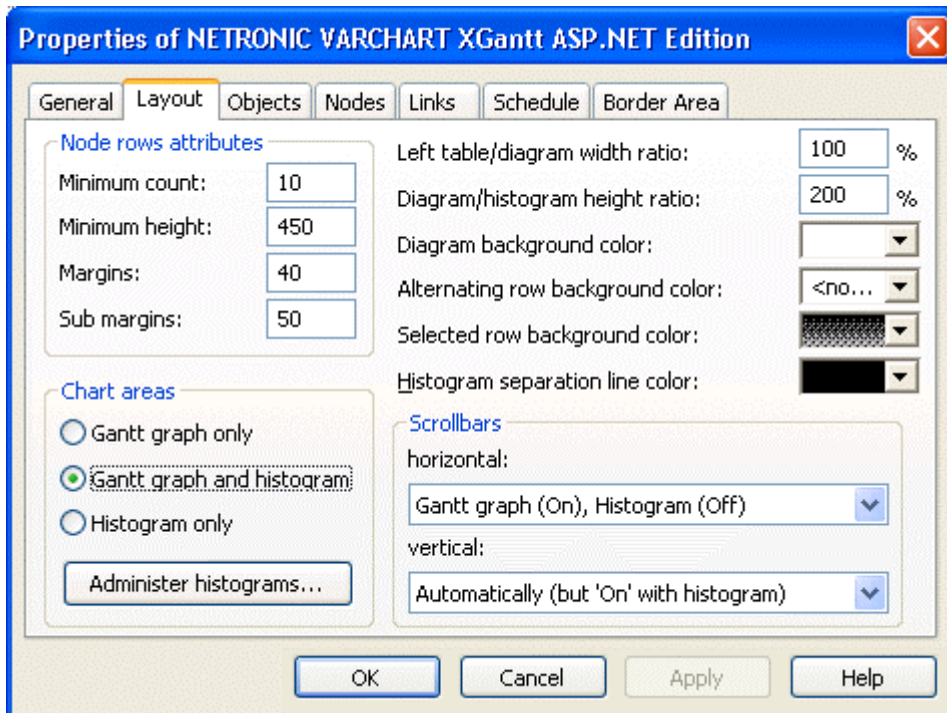
If no individual calendar has been assigned per node, the calendar which was defined as active in the CalendarCollection is used.

## **Layers with nonWork interval**

Please activate this check box to have workfree intervals highlighted. In workfree intervals, bar shaped layers will be displayed as a solid line.

This feature can also be set by the property **VcGanttASP.LayersWithNon-WorkInterval**

## 4.5 The "Layout" Property Page



On this property page you can establish and modify the layout of the chart.

### Minimum count

Specify how many node rows are to be displayed in the diagram area at the program start.

This feature can also be set by the property **VcGanttASP.NumberOfInitial-RowCount**.

### Minimum height

Specify the minimum height of the node rows in 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 **VcGanttASP.-MinimumRowHeight**.

## Margins

Specify the minimum vertical spacing between the node and the upper or lower node row border in 1/100 mm.

This feature can also be set by the property **VcGanttASP.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 **VcGanttASP.SubRowMargins**.

## Chart areas

Specify what the diagram is supposed to display:

- the Gantt diagram only
- the Gantt diagram and the histogram
- the 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 total diagram (table area plus diagram area) at the start of the program. In order to display the table completely on the start, enter the value "-1".

This feature can also be set by the property **VcGanttASP.DiagramWidth-Ratio**.

## 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 histogram completely on the start, set the value "-1".

This feature can also be set by the property **VcGanttASP.Diagram-HistogramHeightRatio**.



## Diagram 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 **VcGanttASP.Diagram-BackgroundColor**.

## Alternating row background color

This field lets you set a second background color to the diagram, which alternates linewise with the **Diagram background color**.

This feature can also be set by the property **VcGanttASP.Diagram-AlternatingRowBackgroundColor**.

## Selected row background color

This field lets you set a background color to the selected row of the diagram.

This feature can also be set by the property **VcGanttASP.SelectedRow-BackgroundColor**.

## Histogram separation line color

By this option you can set the color to the separation lines between histograms.

This feature can also be set by the property **VcGanttASP.Histogram-SeparationLineColor**

## Scrollbars

By these options you can set the horizontal and vertical scroll bars. For the horizontal scroll bar, you can choose between the below options:

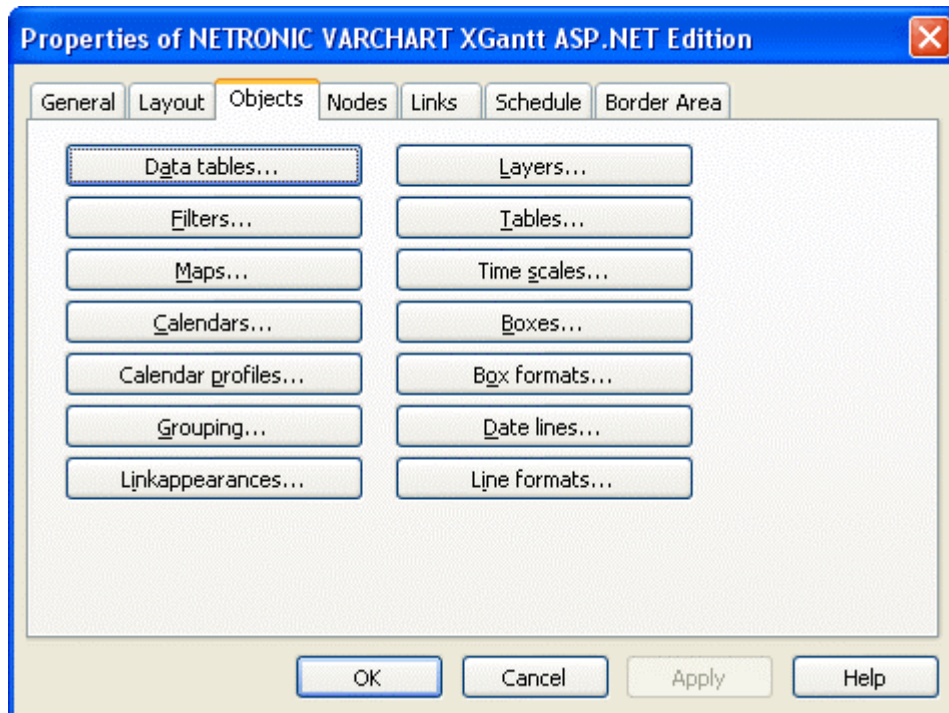
1. **Gantt graph (on), Histogram (off)** - the horizontal scroll bar is located between the Gantt graph and the histogram
2. **Gantt graph (off), Histogram (on)** - the horizontal scroll bar is located below the histogram
3. **None** - there is no horizontal scroll bar.

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.

---

## 4.6 The "Objects" Property Page



### Data tables

Opens the dialog **Administrate Data Tables**.

### Filters

Opens the **Administrate Filters** dialog box.

### Maps

Opens the dialog **Administrate Maps**.

### Calendars

Opens the dialog **Specify Calendars**.

### Calendar profiles

Opens the dialog **Administrate Calendar Profiles**.

## **Grouping**

Opens the dialog **Grouping**.

## **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 **Administrative Boxes**.

## **Box formats**

Opens the dialog **Administrative Box Formats**.

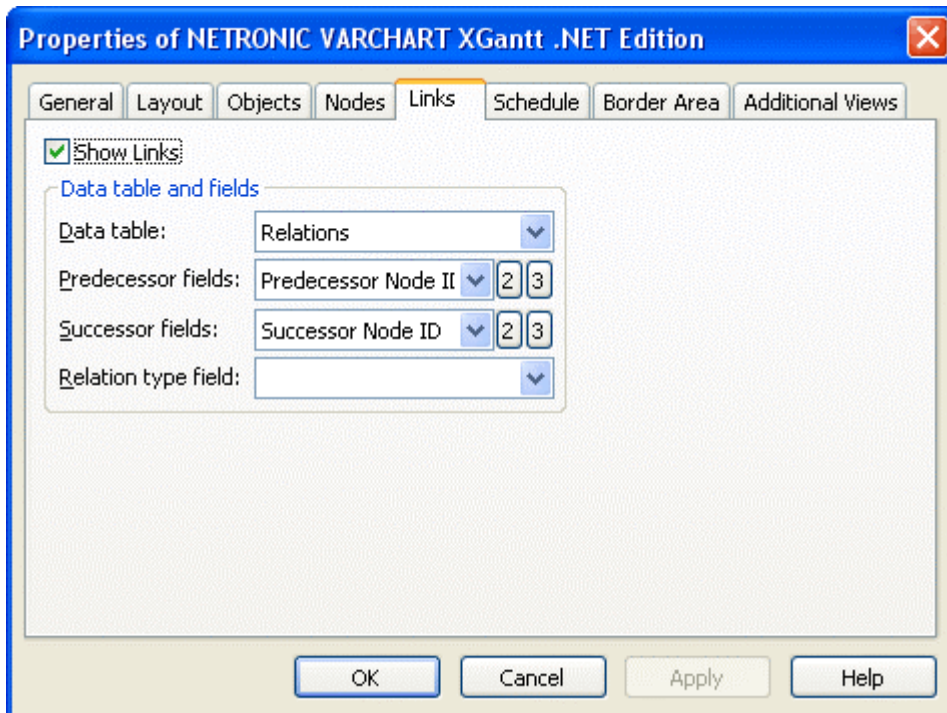
## **Date lines**

Opens the **Specify Date Lines** dialog box.

## **Line formats**

This button lets you open the dialog **Administrative Line Formats**.

## 4.7 The "Links" Property Page



This property page lets you display links between nodes and establish and modify the appearance of the links.

### Data table

Select a data table which contains the fields for the relations. This feature can also be set by the property **VcGanttASP.LinksDataTableName**.

### Predecessor field

This field lets you set the data field or fields from the data table selected above to which the identification of the predecessor node of the link is/are stored.

This feature can also be set by the property **VcGanttASP.LinksPredecessor-DataFieldIndex**.

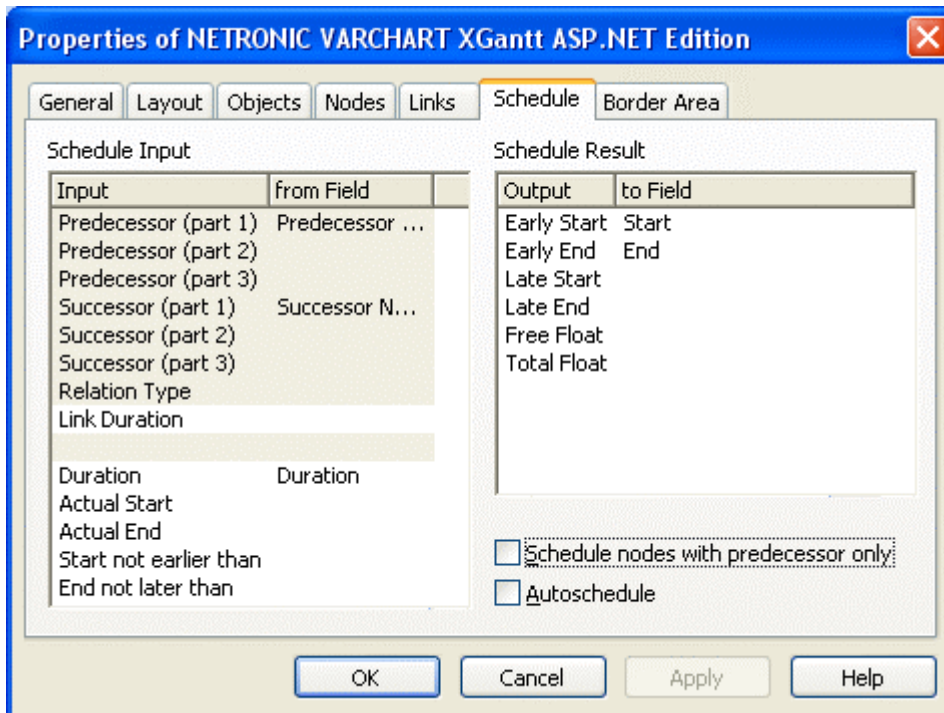
### Successor field

This field lets you set the data field or fields from the data table selected above to which the identification of the successor node of the link is/are stored. This feature can also be set by the property **VcGanttASP.Links-SuccessorDataFieldIndex**.

## **Relation type field**

Select the data field that contains the relation type. This feature can also be set by the property **VcGanttASP.LinkTypeDataFieldIndex**.

## 4.8 The "Schedule" Property Page



This property page lets you adapt the date calculation settings of VARCHART XNet to your interface by specifying the data fields that you want to use for the input (**Schedule Input**) and output (**Schedule Result**) of the scheduler (also see "Important Concepts: Scheduling").

### Schedule Input

Please select for each entry of the column, from which field its contents is to be loaded. The scheduler uses the data fields of the data tables of nodes and links previously set. The calculations of the scheduler are based on the project start, their logic dependencies and the project start given. The fields **Predecessor** and **Successor** cannot be edited by the **Schedule Input** table. They merely display the settings of the **Links** property page.

### Schedule Result

Specify for each result to which field it is to be stored. The scheduler stores only to data fields of 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.

## **Schedule nodes with predecessor only**

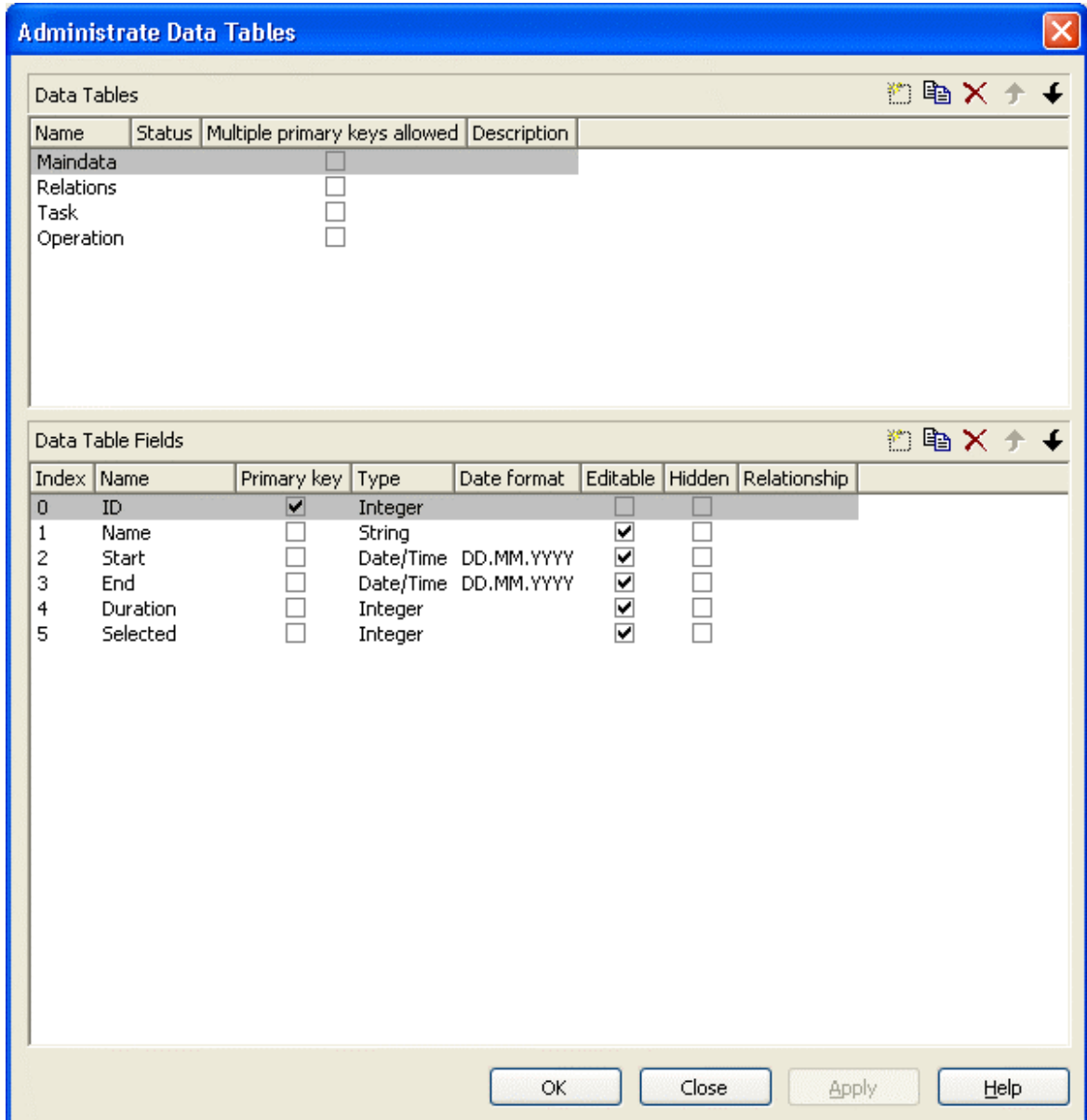
If you activate this check box, only those nodes will be scheduled that have a predecessor node, otherwise all nodes will be scheduled. A project start set will be disregarded when scheduling in the first case.

## **Autoschedule**

If this option is activated, the duration of the depending dates will be recalculated automatically each time a link is created or deleted or if an activity is modified.





## 4.9 The "Administrate Data Tables" Dialog Box








You can reach this dialog via the property page **Objects**. Here you can 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)** tables. 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

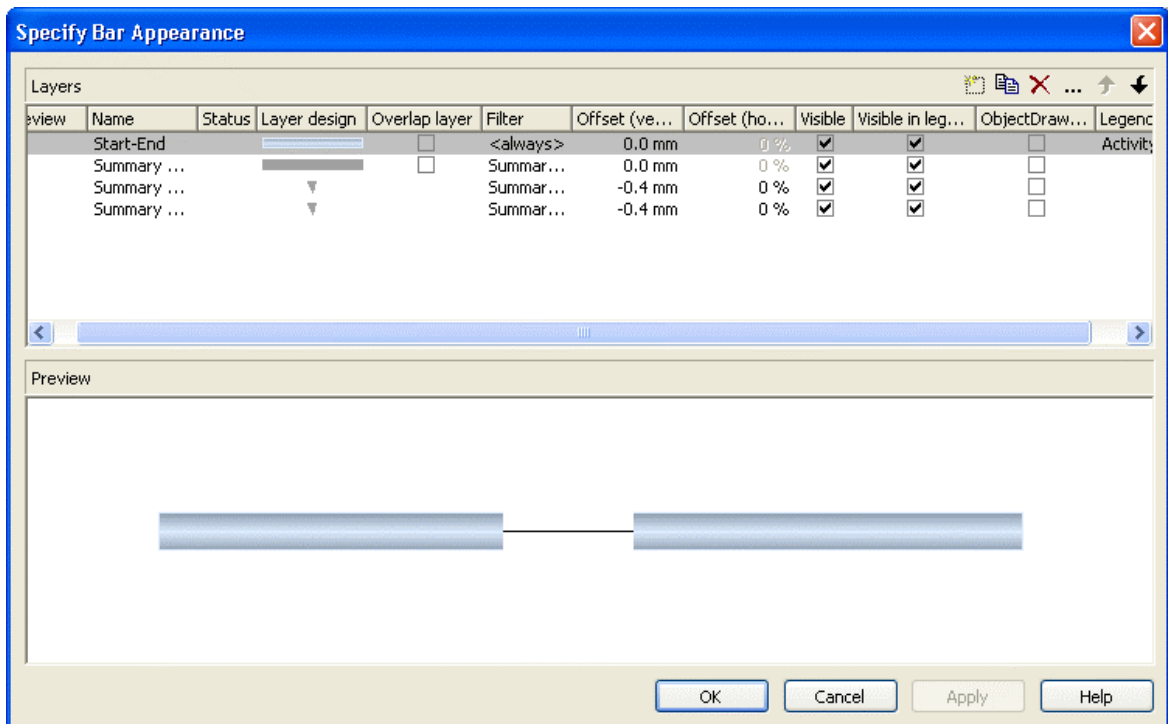
Here you can create and edit data table fields for the selected data table.

- **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:
  - Alphanumeric
  - Integer
  - Date/Time
- **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.

## 4.10 The "Specify Bar Appearance" Dialog Box



Activities are represented by bars. The bar appearance is composed of one or several layers that are assigned to the activities dynamically via filters. Layers represent single dates (symbol layers) or pairs of dates (rectangle layers or line layers) in graphical shape. The dates are taken from the data fields that you can specify in 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 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 the layers to overlap. The layer at the top of the **Layer** table is drawn first; the layer at the bottom of the **Layer** table is drawn last and may overlap the layers 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

Define one layer per line in the table.

### Preview



The layers marked by a small arrowhead in the **Preview** column are displayed in the preview window.

The green arrowhead marks the layer on which the cursor is currently positioned. The layer is temporarily displayed in the preview window, that is, as long as the cursor is on the layer. By clicking on the arrowhead you can make it turn red, and vice versa. A red arrowhead indicates that a layer is displayed permanently in the preview window.

### Name

Lists the names of all layers that were defined. The names can be edited.

### Status

In this column all layers added () and/or modified () after the dialog box was opened are marked by a symbol.

### Layer design


This list contains appearances of layers. To modify the appearance of a layer, click on the **Edit layer** button above the list or double-click on the **Layer design** entry to get to the **Edit Layer** dialog box where you can define the graphical attributes and the annotation of the layer.


### Overlap layer

In the mode **All nodes in one row** and not **optimized**, you can display overlapping layers by an overlap layer. There is only one appearance to all overlap layers. No filter can be used for it.

### Filter

The filter associated to a layer controls the activities that are displayed by the layer. To assign a filter to a layer, mark the **Filter** field. Two buttons will appear:

 Open the select box that lists the available filters and select one.

 Alternatively, click on the **Edit** button in the **Filter** field to get to the **Administrative Filters** dialog box where you can edit, copy, define or delete a filter.

Examples of filters: "Standard", "Critical", "Milestone". The chosen filter stipulates the condition that an activity must fulfil in order to apply the layer. For example, if you choose the "Critical" filter for the "Early" layer, the "Early" layer will only be displayed in critical activities.

## Offset (vertical)

The vertical offset from the central horizontal line of a bar is to be specified in millimetres. Positive values cause the layer to be shifted upwards, negative values will shift the layer downwards.

If you mark 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, respectively.



By the second button you can get to the **Configure Mapping** dialog box. Here you can set vertical offsets in dependence of data.



If a vertical offset was mapped, there will be a bold display of the arrow on the button.

## Offset (horizontal)

*(only for symbol layers)* When you mark the **Offset (horizontal)** field of a symbol layer, two buttons appear with an arrow pointing upwards and downwards respectively. You can use the arrows to increase or decrease the horizontal offset against the layer date (-50 to +50 %).

## Visible

Uncheck 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 **VcObjectDrawing** and **VcObjectDrawn** for nodes which are displayed by this layer.

## Legend text

Define a legend text for the layer.


## Add layer

 A new layer is created.

## Copy layer

 A copy of the selected layer under a new name is created.

## Delete layer


 Deletes the selected layer.


## Edit layer

 You will reach the **Edit Layer** dialog box.

## Promote/Demote layer

If a node comprises more than one layer, the layers are stacked on top of one another. The top layer in the table is drawn first. The lower the position of a layer in the table, the more layers it overlaps, i.e. the order of the layers in the table is the order in which they will be drawn in the diagram.

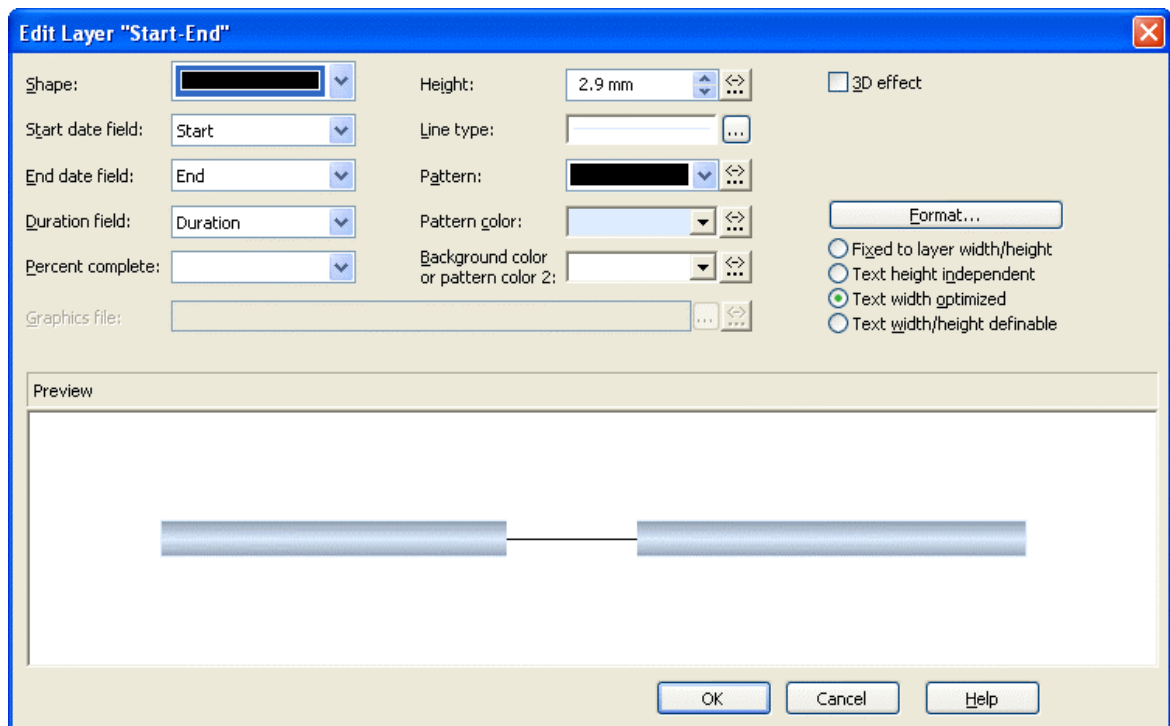
 The selected layer will be moved up one position in the table and one position towards the background in the diagram. The layer at the top of the table is overlapped by all other layers.

 The selected layer will be moved down one position in the table and one position towards the front in the diagram. The layer at the bottom of the table overlaps 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.

## 4.11 The "Edit Layer" Dialog Box



This dialog box you can get to via the **Specify Bar Appearance** dialog box. The name of the layer edited is displayed in the headline.

### 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.



## Start date field

Specify the start date of the selected layer, e.g. Early Start, Late Start, Scheduled Start.

## 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.

## 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 VARCHAR property **FilePath**. If it is not found searching will continue in the current directory of the application and in the installation directory of the VARCHAR 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.

## 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.


## Line type


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.

## Pattern

This field indicates the default layer pattern.

 By the **arrow** button you can open the list of patterns and select a pattern for the layer.


 By the second button you can get to the **Configure Mapping** dialog box. It allows to assign patterns to layers in dependence on data.


 If patterns were mapped, the arrow on the button will appear solid.

**Please note:** If the background color of a field of a node format which was assigned to the node appearance was not set to **transparent**, the selected pattern will not show through!

## Pattern color

This field lets you select the default color for the layer pattern.

 By the **arrow** button you can open the color picker to select a color. Also transparent colors are available.


 By the **edit** button you you can get to the **Configure Mapping** dialog box. It allows to assign colors to patterns in dependence on data.


 If colors were mapped, the arrow on the button will appear solid.

**Please note:**If the background color of a field of a node format which was assigned to the node appearance was not set to **transparent**, the selected pattern will not show through!

## Background color or pattern color 2

This field lets you set the default background color or the second pattern color to the layer.

 By the **arrow** button you can open the color picker to select a color. Also transparent colors are available.

 By the **edit** button you can get to the **Configure Mapping** dialog box. It allows to assign colors to patterns in dependence on data.

 If colors were mapped, the arrow on the button will appear solid.

**Please note:**If the background color of a field of a node format which was assigned to the node appearance was not set to **transparent**, the selected pattern will not show through!

## 3D-Effect

Decide whether or not the layer should be given a 3-dimensional perspective.

## Format

Opens the **Edit Layer Format** dialog.

## 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

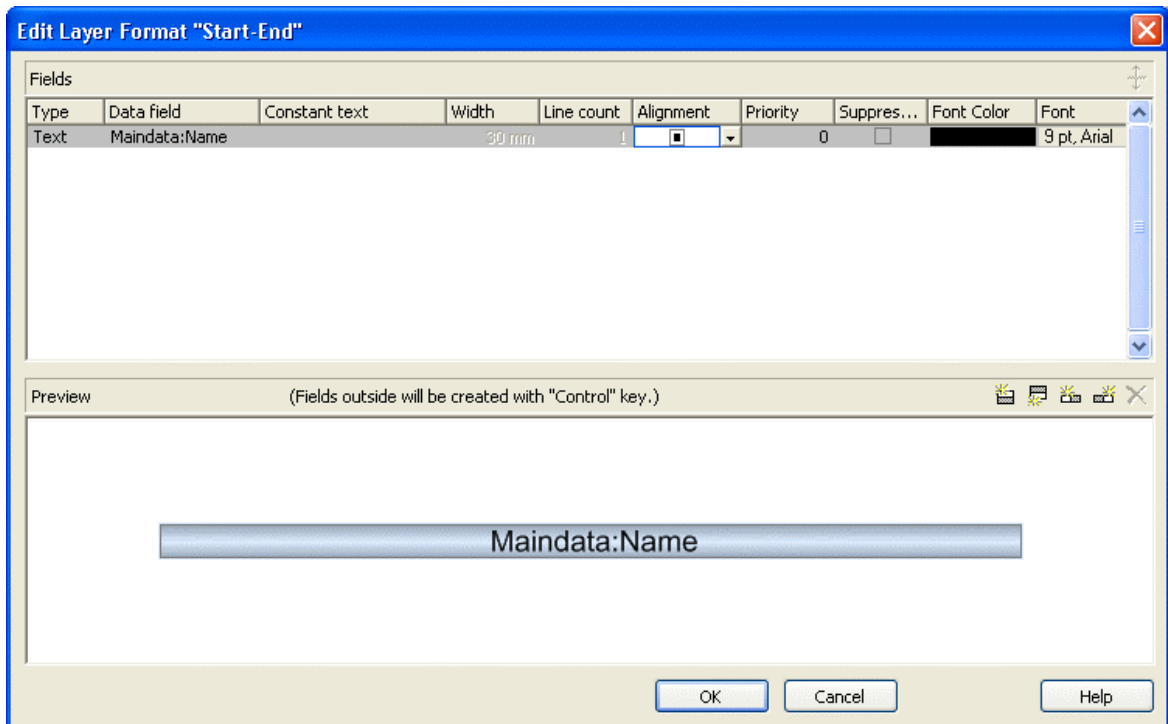
If you select this option, the annotation width and height will be independent of the layer width or height respectively. Then you can specify for each field the width and the number of lines individually in the **Edit Layer Format** dialog or by the properties **MinimumWidth** and **TextLineCount** in objects of the type **VcLayerFormatField**.

## 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.)

## 4.12 The "Edit Layer Format" Dialog Box



You can get to this dialog box via the **Format** button of the **Edit Layer** dialog box.

### Type

The field type (text) is displayed here.

### Data field

Select the data field the content of which is to be displayed in the current field. In addition to the data fields defined in the data definition table, you can select the option <Row number> to display the number of the row which contains the layer.

If the content of a data field does not fit into the current field, the excess characters will be clipped in the diagram.

### Constant text

*(only if no data field was 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

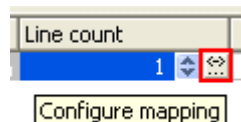
Specify the number of lines of text that can be displayed in the current field.

**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 calculated 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 allowed. If the total width of the layer is too small to show the contents of all layer fields, the priority of the layer field determines if its content is displayed. The content of the field of highest priority is displayed first, if possible, completely. The contents of fields of lower priorities are displayed subsequently. If a field content cannot be displayed completely, it will be suppressed or truncated (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 or truncated.

## 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 can get to the **Configure Mapping** dialog box. It allows to assign colors in dependence of data.

 If colors were mapped, the arrow on the button will appear solid.

## Font

Indicates the font style of the field. If you click on the field, two buttons will appear:

 The Windows **Font** dialog box will appear.

 By the second button you can get to the **Configure Mapping** dialog box. It allows to assign fonts in dependence of data.






 If fonts were mapped, the arrow on the button will appear solid.

## Apply selected property to all fields

 Applies the marked property to all fields.

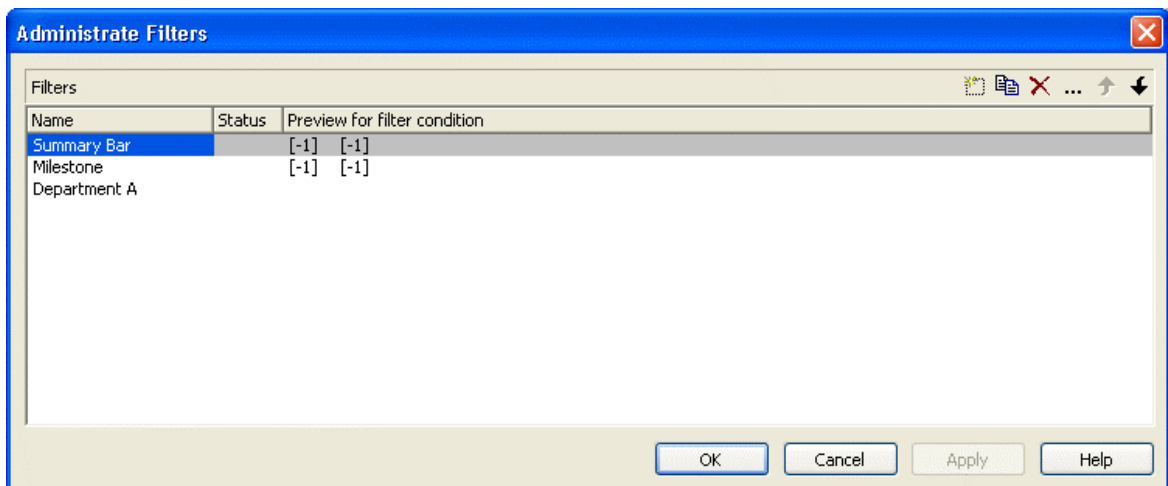
## Preview

The current fields are displayed in the preview window. If you click on a field, you can modify its attributes in the **Fields** table.

     By 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** key in addition. You also can use the **Del** key to delete fields.



## 4.13 The "Administrate Filters" Dialog Box





You can get to this dialog box

- via the **Objects** property page
- for layers: via the **Specify Bar Appearance** dialog box
- for table formats: via **Edit Table** dialog box
- for links: via the **Filter** button of the **Link** property page
- for histogram curves: via the **Filter** combo box of the **Edit Histogram** dialog
- for nodes: via 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 each filter that has been added () and/or modified () since the dialog box was opened is marked by a symbol.


### Data definition table

This column shows the data definition table (**Maindata** or **Relations**) for each filter and is only shown if the check box **Extended data tables enabled** on the property page **General** is not ticked.


## Preview for the filter condition

This column shows the criteria of each filter. The criteria cannot be edited here. To modify the filter criteria, click on the **Edit filter** button.


## Add filter

 A new filter will be created. You can modify its default name by double-clicking and editing it. New filters are created context-sensitively, i. e. the data definition table always will be specified 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.

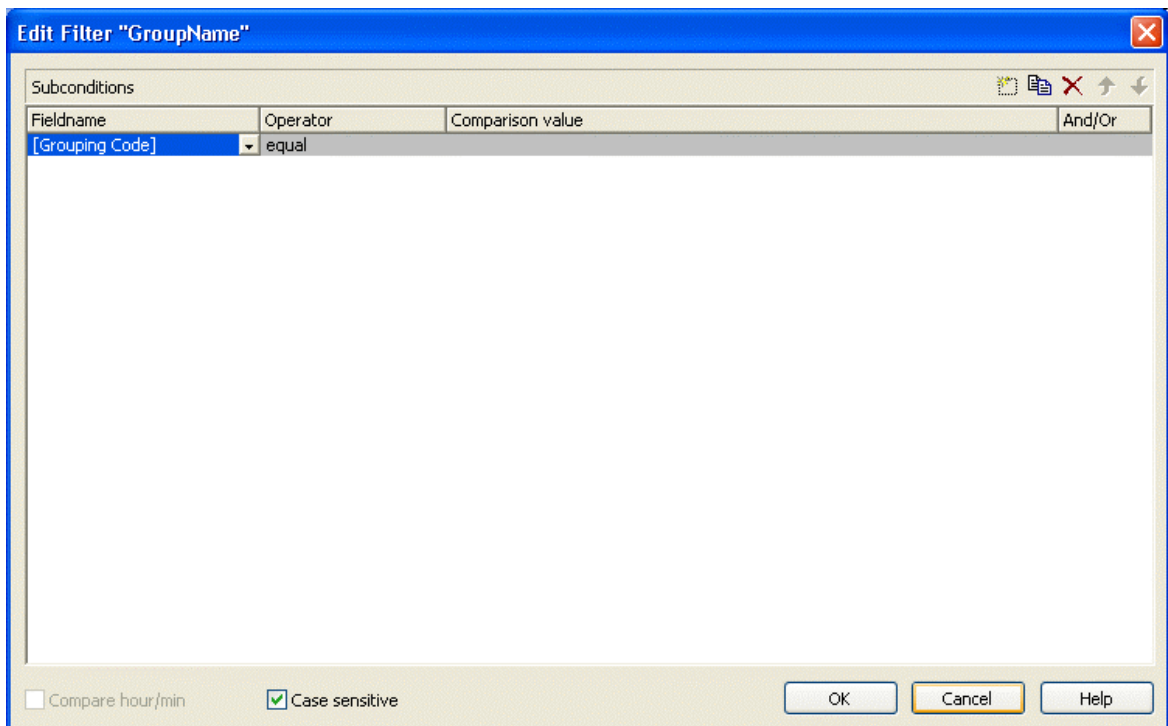
## Edit filter

 Press the **Edit filter** button to view or modify the criteria of a filter. The **Edit Filter** dialog box will appear where you can edit the criteria of the corresponding filter.

## Promote / demote filter

 By these buttons you can move the filter by one position up or down in the list.

## 4.14 The "Edit Filter" Dialog Box



You will get to this dialog box after clicking on the **Edit filter** button in the **Administrative Filters** dialog box. The head line of this dialog box indicates the name of the current filter.

### 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, they are evaluated one by one, starting by the top of the list.

You can click on the **Evaluate subcondition earlier/later** button to move a selected subcondition upward or downward by one position in the table to have it worked off earlier or later.

## Fieldname

This list contains all data fields available to be compared to the comparison value. It also contains some predefined settings:

- The <summary bar level> entry can be used for displaying summary bars in Gantt diagrams. For example, you can specify a filter containing the condition "<summary bar 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** has to be activated on the **Sorting** property page.
- Filters containing the <grouping level> setting can be used for example in the **Edit Table** dialog (for Gantt diagrams) as row filters for basic rows.
- <Gantt: collapsed>: to collapse groups
- <Gantt: nodes in separate rows>: to display all nodes in separate rows
- <Gantt: nodes overlaid>: to allow for overlapping nodes
- <Gantt: row>: to define filters for single rows
- <Gantt: summary node>: definition of summary bars
- <Node Read Only>: filters that select 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).

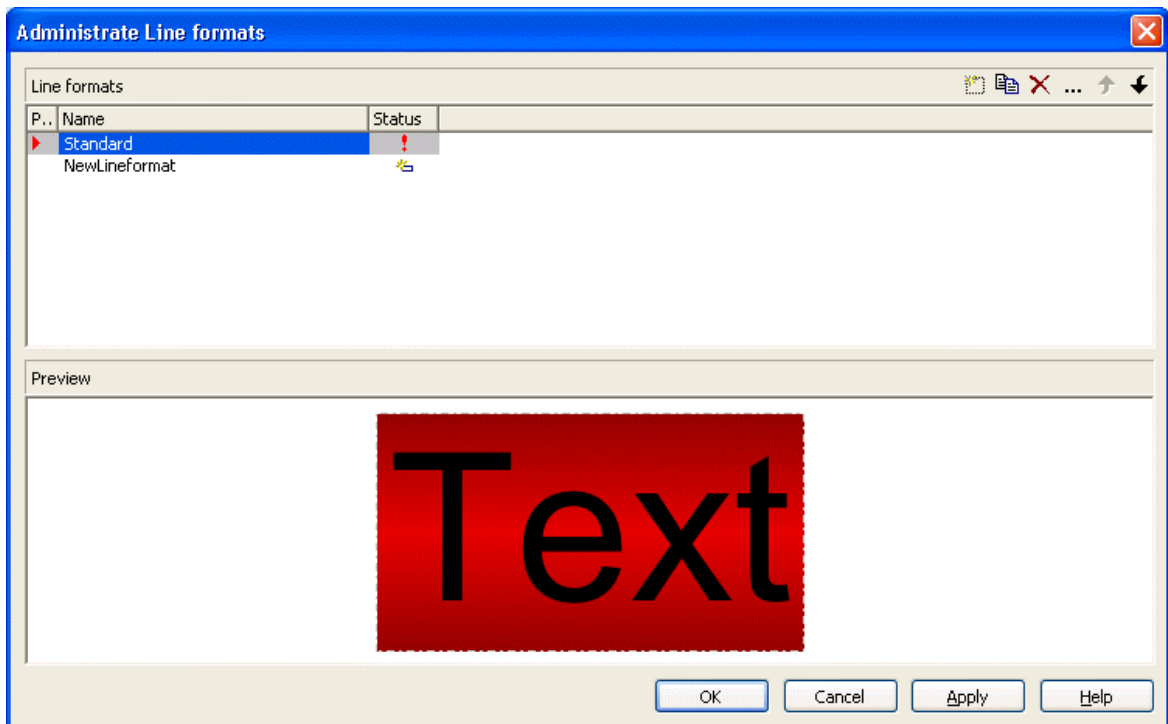
## **Compare hour/min**

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.

## 4.15 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 line format 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

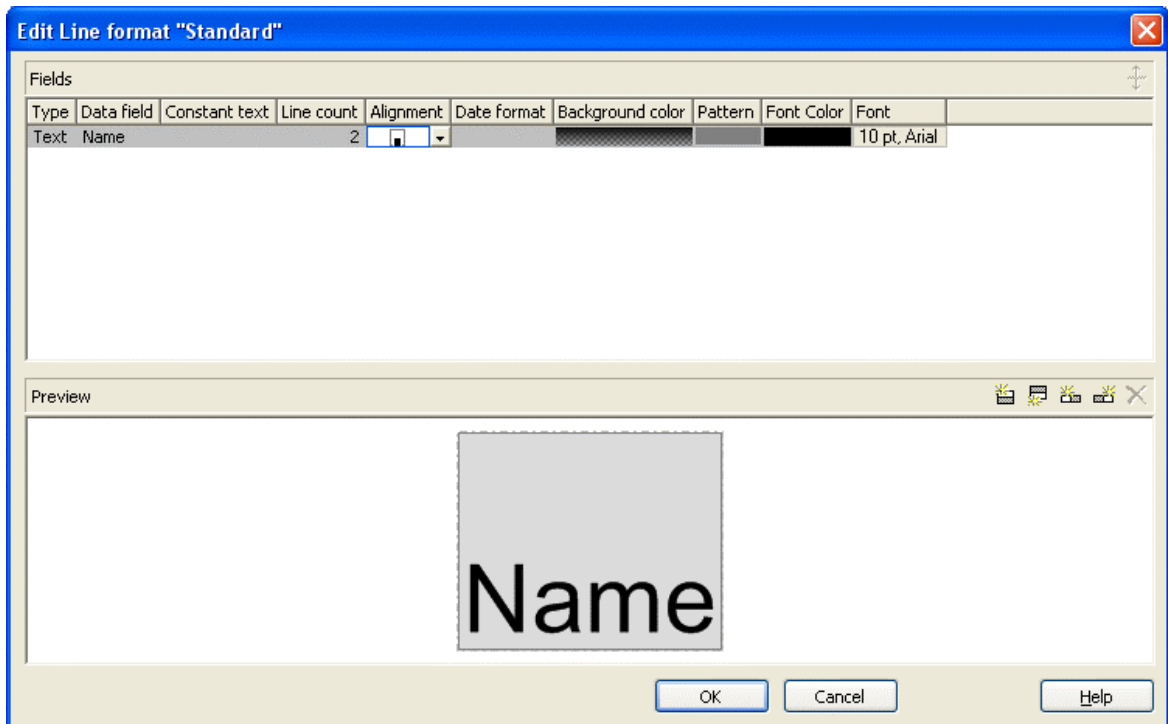
 Opens the dialog **Edit Line format** which lets you specify the attributes of the line format such as color, pattern etc.

## Promote / demote line format

  By these buttons you can move the line format by one position up or down in the list.



## 4.16 The "Edit Line format" 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.

### 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).

## Background color


Select the background color for the current field. You can define your own colors in addition to the ones suggested. Also, transparent colors are available.

If you click on the field, two buttons will appear:



by the arrow button you can open the color picker to select a background color.



by clicking on 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 (****).

## Pattern

Select the fill pattern for the current field.


## 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

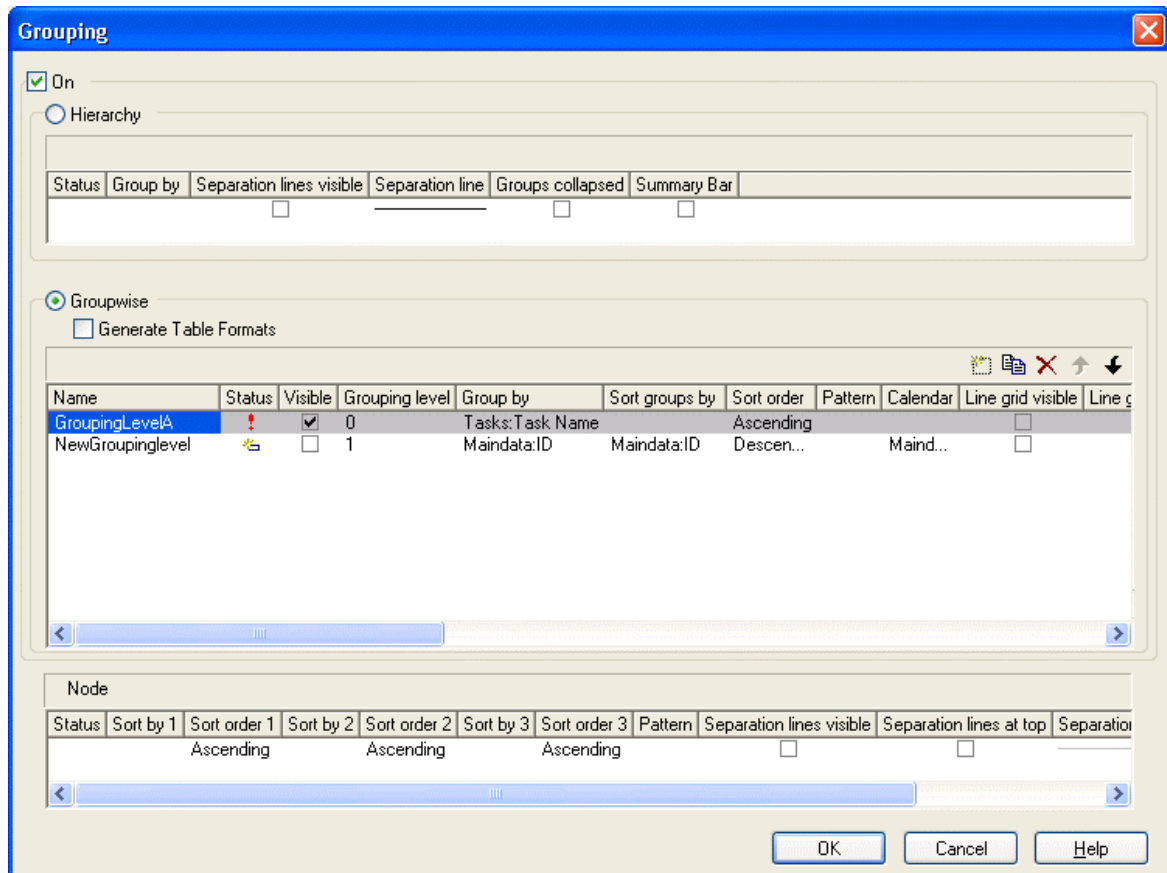
 Applies the marked property to all fields.

## 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. You also can use the **Del** button to delete fields.

## 4.17 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 are 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.

## Separation line

By clicking on  you can open the dialog **Line attributes** and specify the style of the separation lines.

## Groups collapsed







If you select this option, all group levels from the second one downward will be displayed collapsed on the start of the program. They can be expanded interactively after the start.

## Summary Bar

If you tick this box, summary bars will be displayed in all levels. If you want to display summary bars only for special levels, you have to define a layer with an appropriate filter condition (<Summary bar level> = ...).

### ► 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.

## 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.

## Groups sorted by

Select the data field by which the groups 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.

## 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.

## 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, a separation line will be drawn also between groups on different levels.

## Separation line

You can edit the appearance of the separating lines after clicking on the **Edit** button.

## Nodes in separate rows

If you choose this option, each node of a group will be displayed in a separate row.



If this option is **not** selected 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 optimized**

*Only selectable, if **Nodes in separate rows** has not been ticked.* Select this option to automatically optimize the node layout.

### **Groups collapsed**

*(Only available if **Nodes in separate rows** is activated)* 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 = ...).

### **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.

#### **► 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 activities 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.

## 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 also be drawn between groups of different levels.

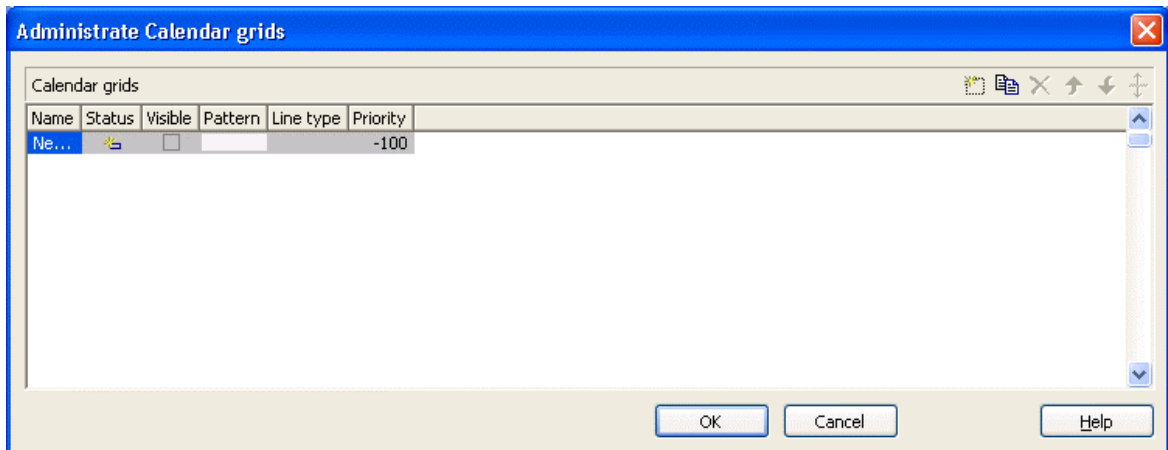
### **Separation line**


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.

## 4.18 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 following settings can be specified for the calendar grids:

### Name

Enter a name for the calendar grid.


### Index

Displays the serial number of a calendar grid (cannot be edited).


### Visible

Activate this check box for the calendar grids to be displayed.

### 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.

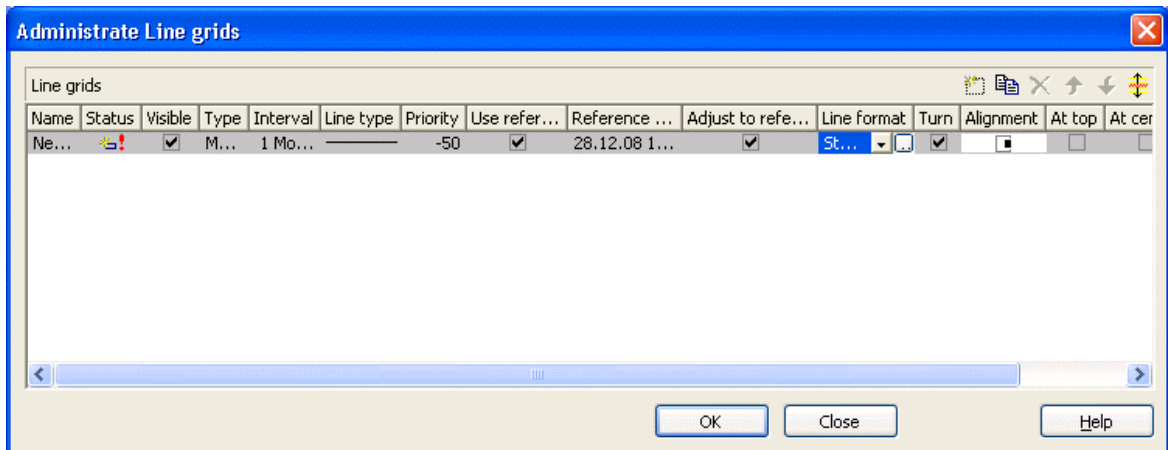
## **Line type**


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).

## 4.19 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 following settings can be specified for the line grids:

### Name

Enter a name for the line grid.

### Index

Displays the serial number of a line grid (cannot be edited).

### 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 chick 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**

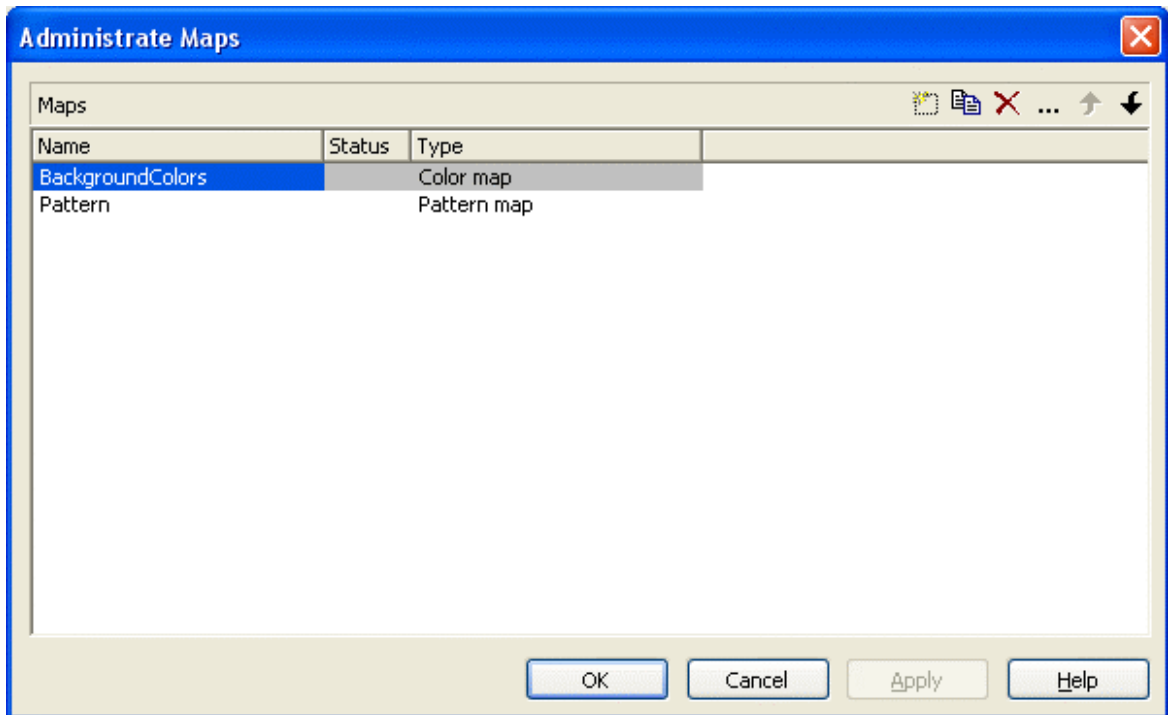
Tick this check box to position the annotations of the lines in the line grid at the center of the Gantt graph.

### **At bottom**

Tick this check box to position the annotations of the lines in the line grid at the bottom of the Gantt graph.



## 4.20 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.

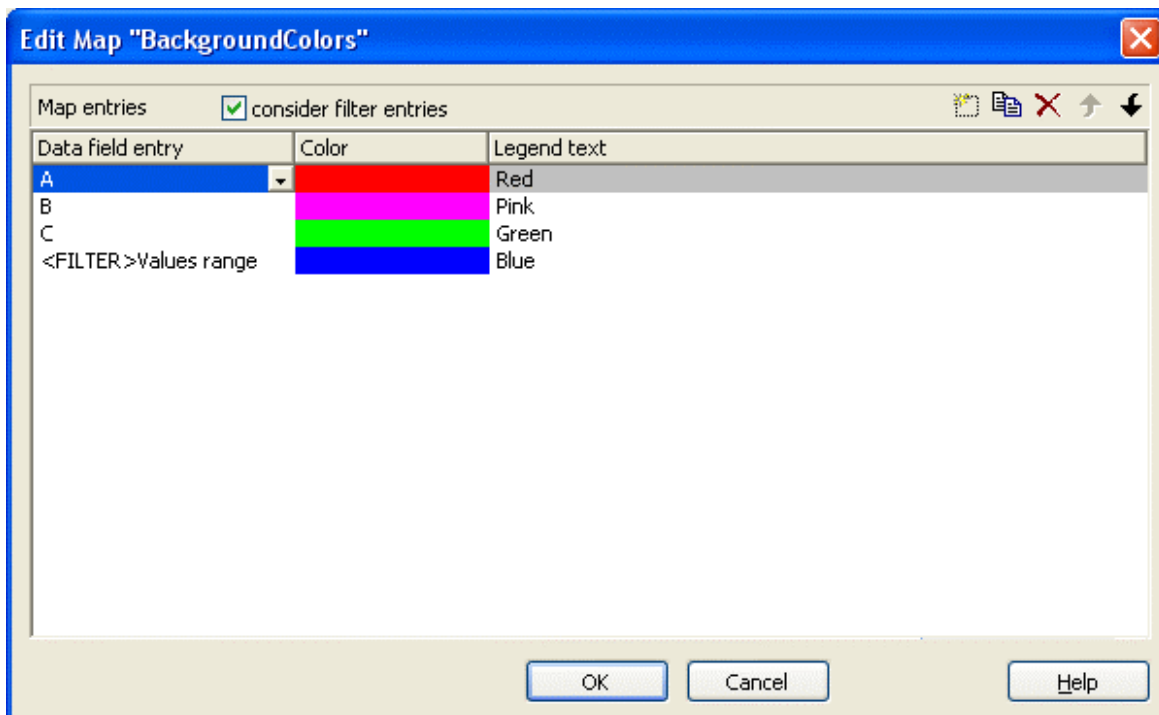
## Edit map

 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.

## 4.21 The "Edit Map" Dialog Box



You invoke this dialog box by clicking the **Edit map** button (⋮) of the **Administrative 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/Pattern

Assign colors or patterns to the data field entries. To do so, click on the corresponding field. A dialog box will open that lets you select a color or a pattern, respectively. The color dialog box also offers transparent colors.

## 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.

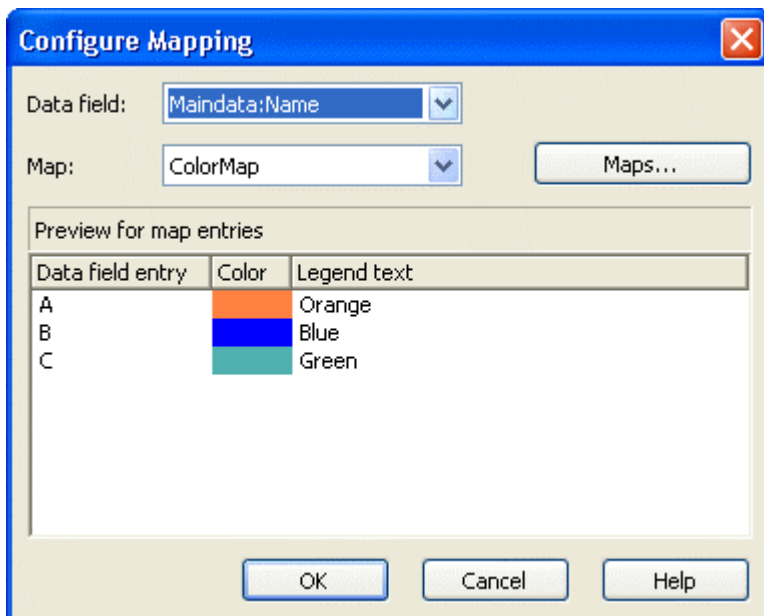
## Delete map entry


 The marked map entry in the list will be deleted. You can only delete map entries that are not currently used.

## pro mote / demote map entry

 The selected map entry can be moved by one position up or down in the list.

## 4.22 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 various dialogs, e.g. 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.

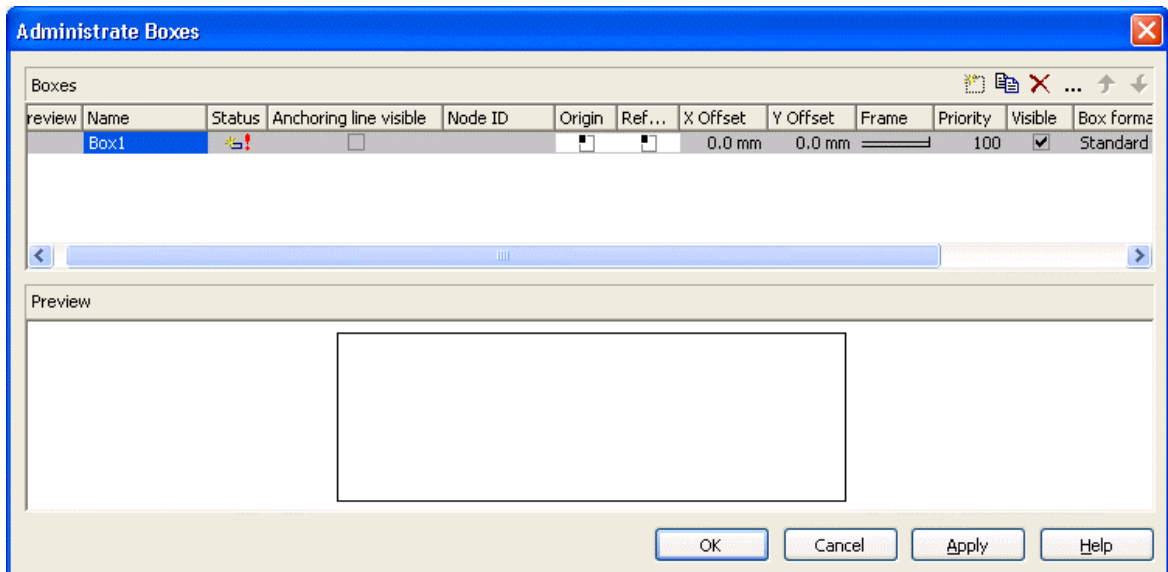
### Maps

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 attributes assigned to them.

## 4.23 The "Administer Boxes" Dialog Box



In the diagram area, boxes can be displayed, that you can administer by the above dialog. You can get to this dialog by the **Objects** property page.



### Preview

The box marked in the **Preview** column is displayed in the preview window.

### Name

Lists the names of all existing boxes. The names can be edited.

### Status

In the **Status** column all boxes added (  ) and / or modified (  ) after the dialog box was opened are marked by a symbol.

### 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.

## 214 The "Administer Boxes" Dialog Box

**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 is 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 is 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 between origin and reference point in x direction.

### Y Offset

Set the distance between origin and reference point in y direction.

### Frame

If you click on the **Frame** field, an **Edit** button will appear that lets you open the **Line Attributes** dialog box. In the dialog box you can specify the type, the thickness and the color of the box frame line.

### Priority

Set the drawing priority of the box in relation to other objects in the diagram (nodes, grids, etc.). The priority of nodes is 0. If the priority of boxes is higher than the one of nodes, the boxes may hide the nodes and may thus inhibit interactive access.

## 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 select box you can choose a box format.



By the **Edit** button you can get to 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



The Box selected will be copied.

## Delete box



The box marked in the list will be deleted.

## Edit box



The **Edit Box** dialog box will appear.

## Promote / demote box

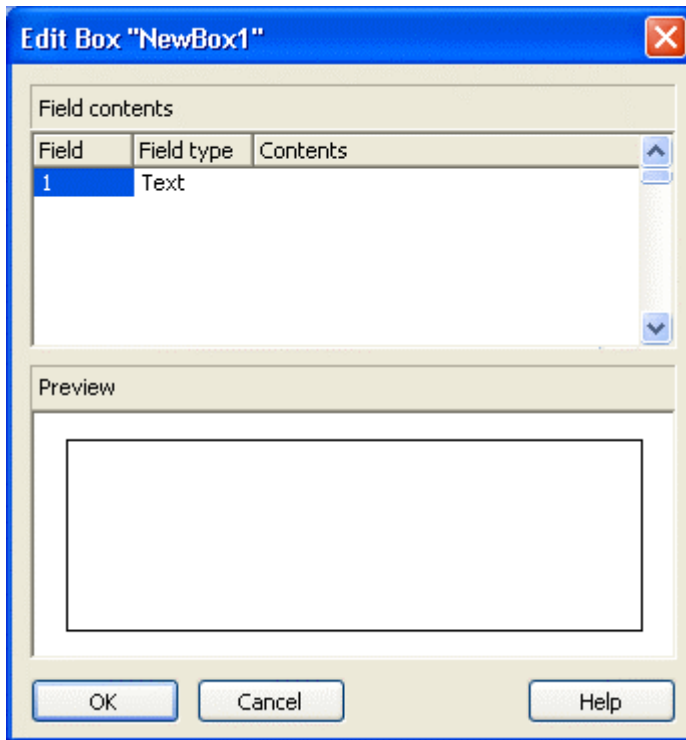


By these buttons you can move the box by one position up or down in the list.



---

## 4.24 The "Edit Box" Dialog Box



This dialog box will appear if you click the **Edit box** button in the **Administrate Boxes** dialog box .

This dialog box also appears at run time if you double-click 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).

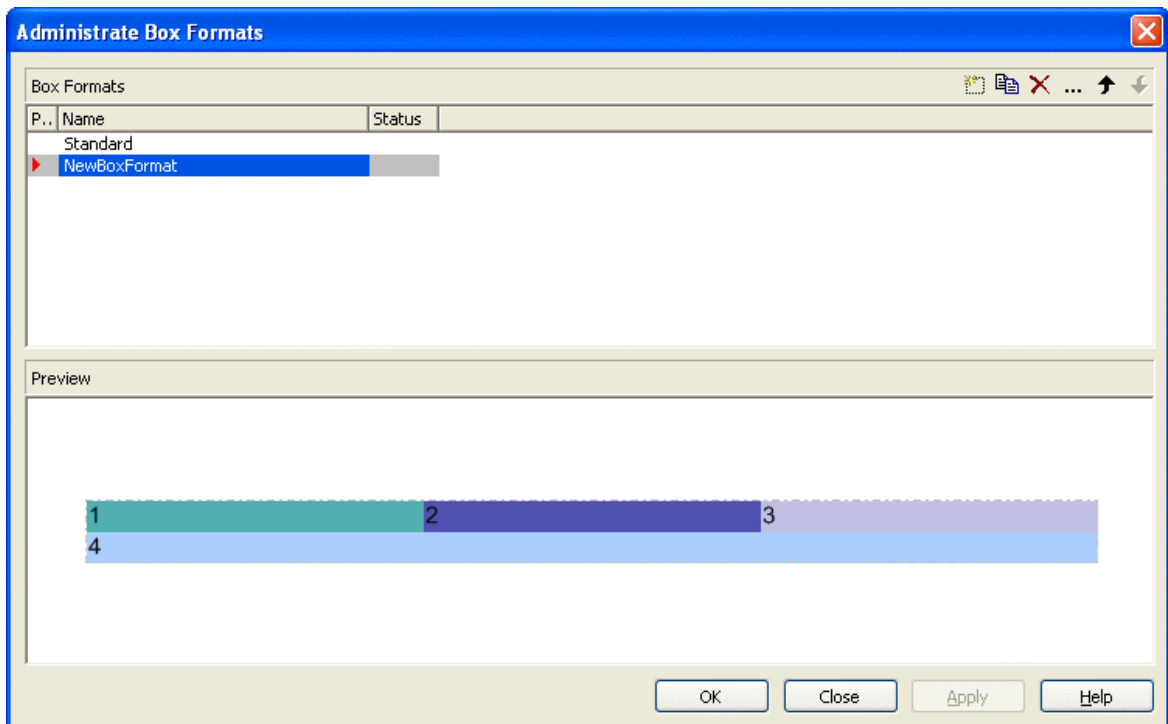
### Contents

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.

## 4.25 The "Administrate Box Formats" Dialog Box



This dialog you can get to by the **Objects** property page.



### Preview

The preview window shows the box format marked in the **Preview** column.


### Name

Lists the names of all existing formats. The names can be edited.

### Status

In the **Status** column the formats added (  ) or modified (  ) after the dialog box was opened are marked by a symbol.


### Add box format

 A new format will be created. You can change its default name by double-clicking and editing it.

## **Copy box format**

 The marked format will be copied.



## **Delete box format**

 The marked format in the list will be deleted. You can only delete formats that are not being used.

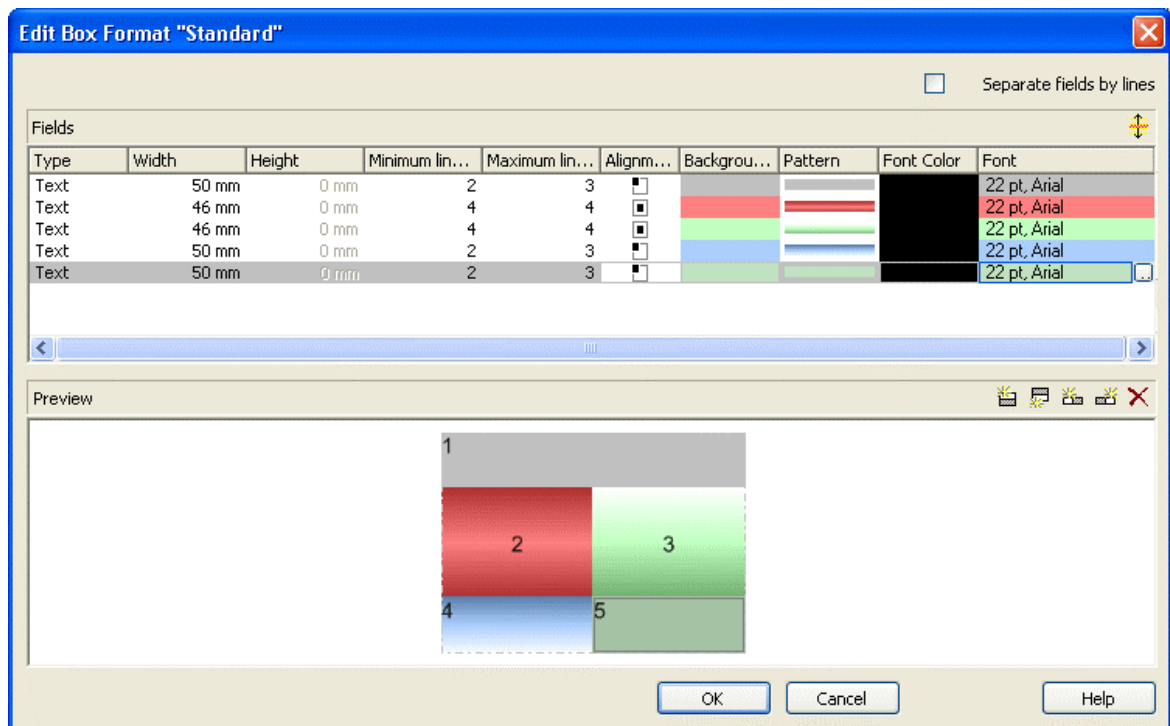
## **Edit box format**

 You will get to the **Edit Box Format** dialog box.

## **Promote / demote box format**

  By these buttons you can move the selected format by one position upward or downward in the list.

## 4.26 The "Edit Box Format" Dialog Box



This dialog box will appear if you click the **Edit box format** button in the **Administrative Box Formats** dialog box.

### Separate fields by lines

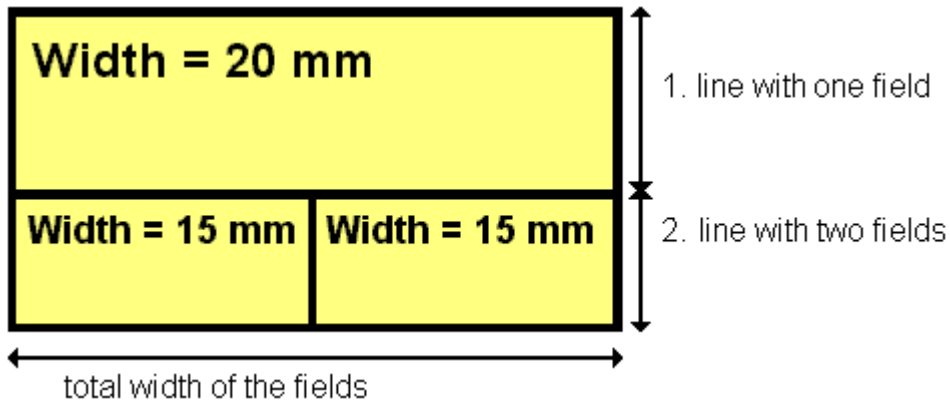
Activate this check box if the box fields are to be separated by lines.

### 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).

## Background color

Select the background color for the current field. You can define your own colors in addition to the ones suggested. Also, transparent colors are available.

## Pattern

Select the fill pattern for the current field.

## 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

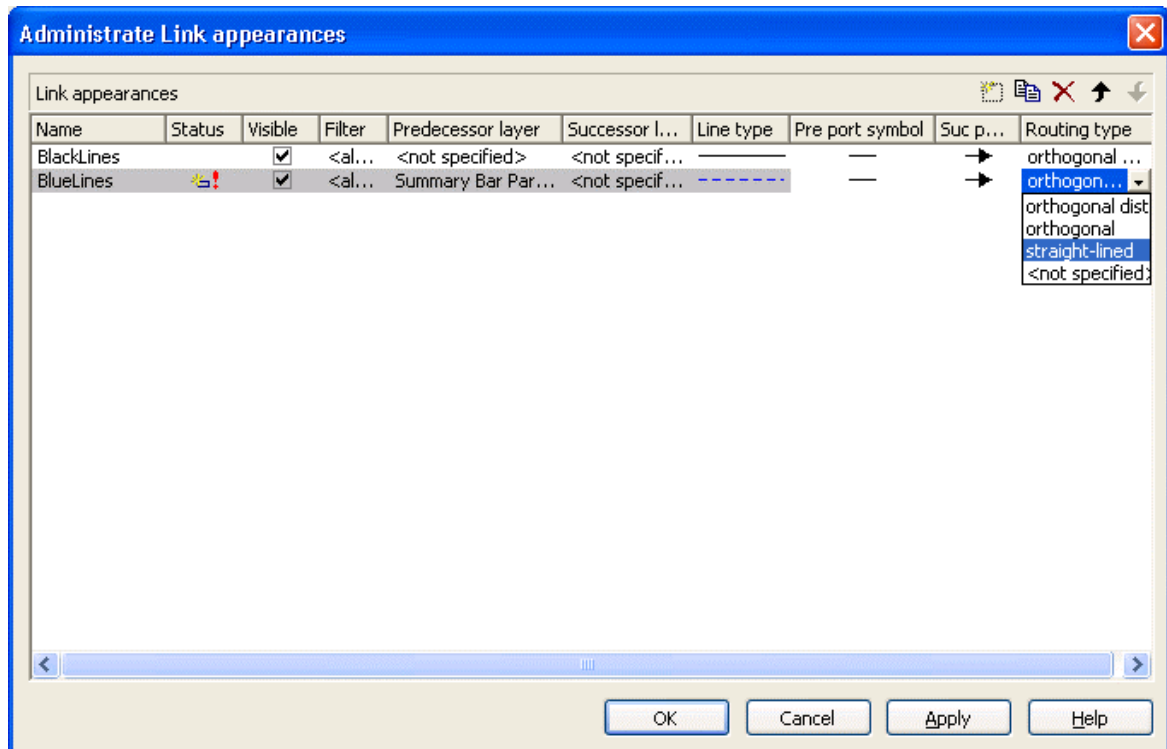
 Applies the marked property to all fields.

## 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.

## 4.27 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 appearances available. The names can be edited.

This feature can also be set by the property **LinkAppearanceName**.

### 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 **VcLinkAppearance.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 **VcLinkAppearance.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 **VcLinkAppearance.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 **Edit Line attributes** dialog box. There you can set type, thickness and color of the line.

This feature can also be set by the property **VcLinkAppearance.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 **VcLinkAppearance.-PredecessorPortSymbol**.

## Suc port symbol

Select a port symbol for a link that visually accentuates the junction of the link and the successor node.

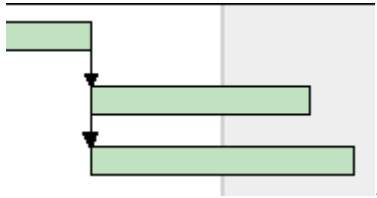


This feature can also be set by the property **VcLinkAppearance.Successor-PortSymbol**.

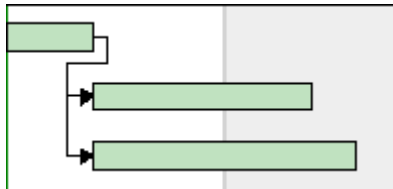
## Routing type

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.

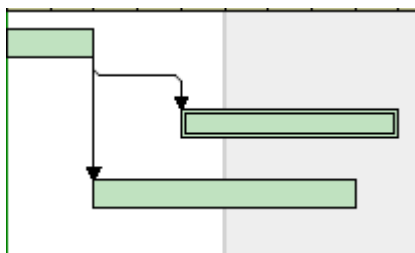
The routing type can also be set by the **VcLinkAppearance** property **RoutingType**.



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.

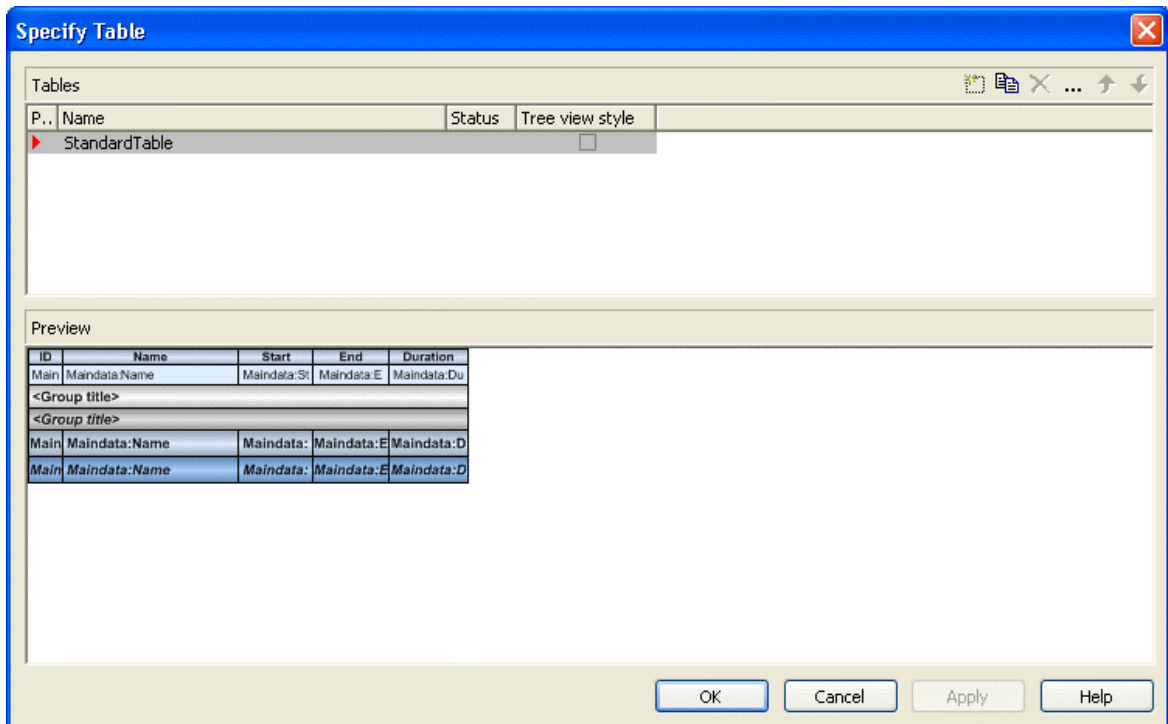
## **Delete link appearance**

 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 link appearance by one position up or down in the list.

## 4.28 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.

### 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.

ID	Name
-	
	Snyder
	Smith

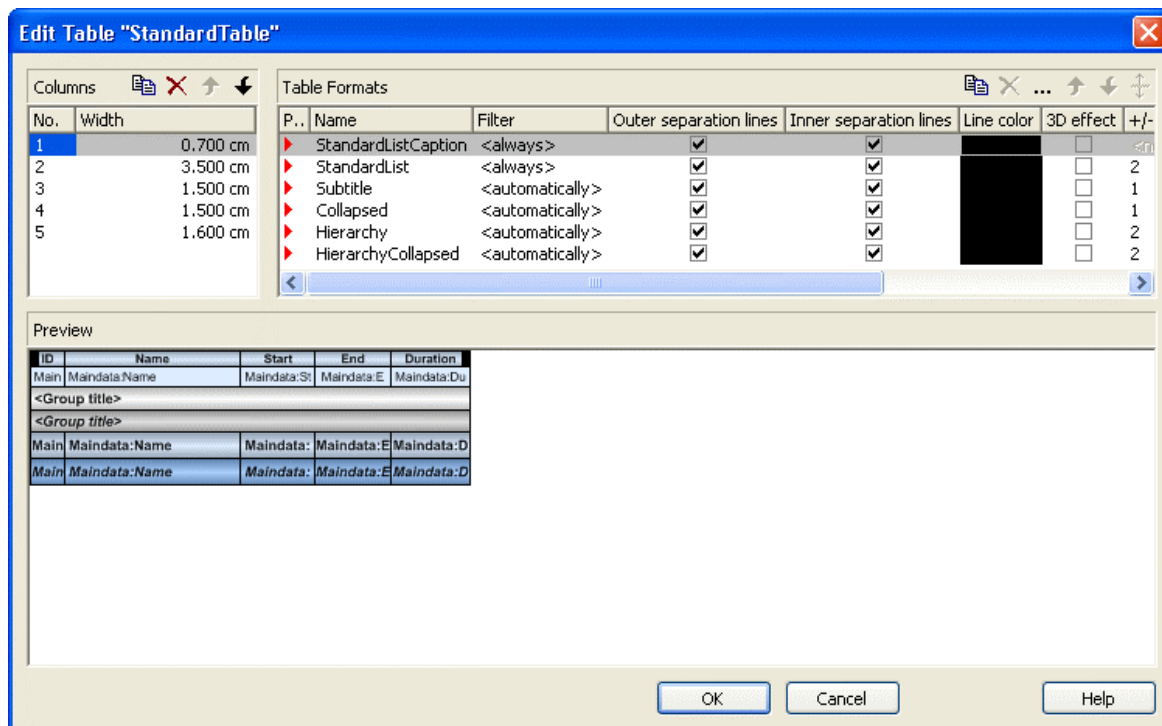
ID	Name
-	
1	Snyder
2	Smith

*Pictures above: a group with and without the tree view style set*

### **Add / copy / delete / edit / promote / 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.

## 4.29 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 addition, 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 subordinated 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.

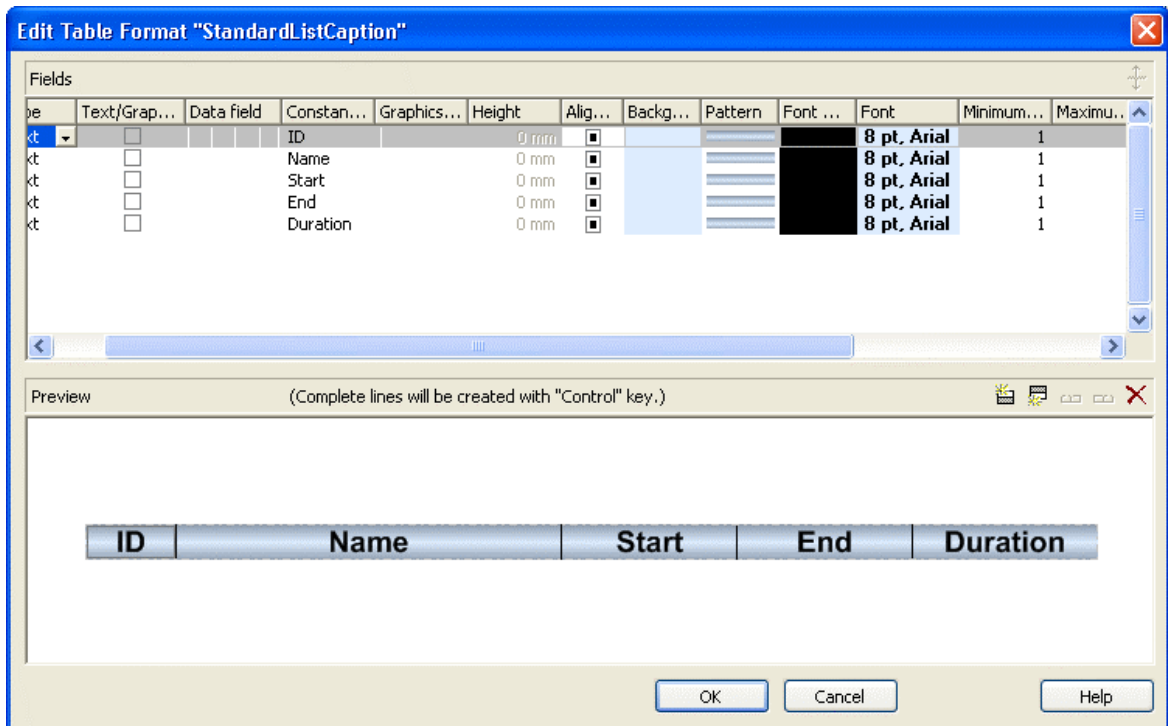
  ... By using these buttons at the top of the **Table Formats** list you can copy or delete table formats or open the **Edit Table Format** dialog.

**Note:** For the table format **StandardListCaption** (table caption) attributes cannot be assigned using 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.

## 4.30 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 an index that allows to access the table format field by API calls.

If you create a new table format field in the preview window, the index preliminarily will receive "?" instead of a number. The "?" will be replaced by a number when the dialog is left by **OK**, which can be verified when re-opening the dialog.

### 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.

### Text/graphics combined

If this check box is activated, in the table format field a text and a graphics can be combined as follows:

- **Type:** Text, **Text/graphics combined:** no: Only text will be displayed (as specified for **Data field** or for **Constant text**).
- **Type:** Graphics, **Text/graphics combined:** no: Only a graphics will be displayed (as specified for **Graphics file name**).
- **Type:** Text, **Text/graphics combined:** 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, **Text/graphics combined:** 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 the content of which is to be displayed in the current field. In addition 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 field, excess characters will be truncated when displayed.


## 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 the directory of the graphics file to be displayed in the selected table format field.

If you click on a **Graphics file name** field, two buttons will appear:


Click on the first button  to open the Windows dialog box **Choose Graphics File**. It lets you select the graphics file to be displayed in the selected table format field.

If a relative file name was chosen, at run time the file will at first be searched in the path set by the VcGantt property **FilePath**. If it is not found there, it will be searched in the current directory of the application and in the installation directory of the VARCHART XGantt control.



 Click this button to use a map for displaying graphics in table format fields depending on the node data. The **Configure Mapping** dialog box will open which lets you combine a map and a node data field, the map assigning graphics files in dependence of the data field entries.

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 a name of a graphics file. If in the data field or in the map no valid graphics file name can be found, the file name specified in the **Symbol file field** will be used.

After a node data field and a map have been combined, the arrow on the second button will turn to bold: .

 When you leave the **Symbol File Name** field, a symbol indicates that a map was assigned to a data field.

When the graphics is displayed, the color of the pixel in the top left corner will be replaced by the color of the diagram background, and so will all pixels of the same color. Therefore all pixels of the graphics that that show the same color as the top left corner pixel are 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).

### Background color

This field lets you set the default background color of the table format.


 By the arrow button you can open the color picker to select the background color. Also transparent colors are available.


 By the second button you can get to the **Configure Mapping** dialog box. It allows to assign colors to the table format in dependence of data.

 If colors were mapped, the arrow on the button will appear solid.

### 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 can get to the **Configure Mapping** dialog box. It allows to assign font colors in dependence of data.

 If colors were mapped, the arrow on the button will appear solid.

## 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 can get to the **Configure Mapping** dialog box. It allows to assign fonts in dependence of data.

 If fonts were mapped, the arrow on the button will appear solid.

## 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.

## 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.

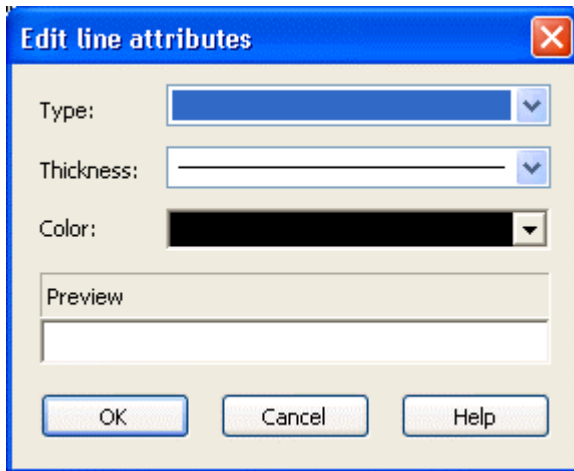
 With the help of the buttons above the preview window you can add new fields or delete the marked field. You also can use the Del button to delete fields.


## **234** The "Edit Table Format" Dialog Box

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.

---

## 4.31 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

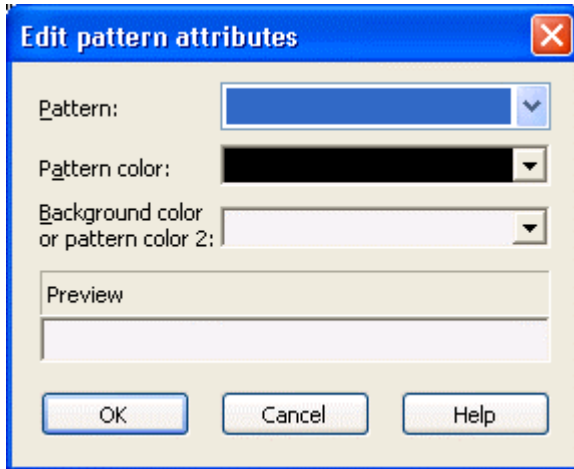
Select the line color.


### Preview

The line appearance based on the current settings is displayed in this field.

---

## 4.32 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, for calendar grids, for the title of a group and for node lines.

### **Pattern**

Here you can select a fill pattern.

### **Pattern color**

Select the foreground color of the fill pattern.

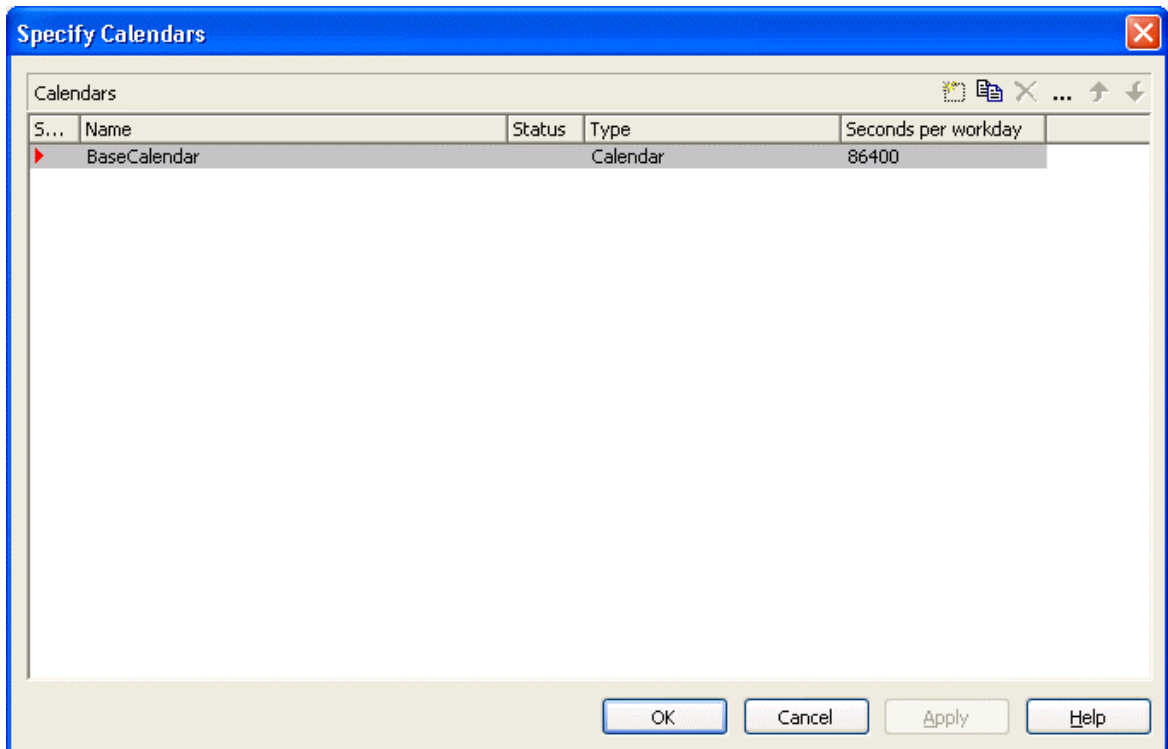
### **Background color or pattern color 2**

Select the background color or a second pattern color.

### **Preview**

The pattern based on the current settings is displayed in this field.

## 4.33 The "Specify Calendars" Dialog Box



You can get to this dialog via the **Objects** property page. Define one calendar per line in 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**



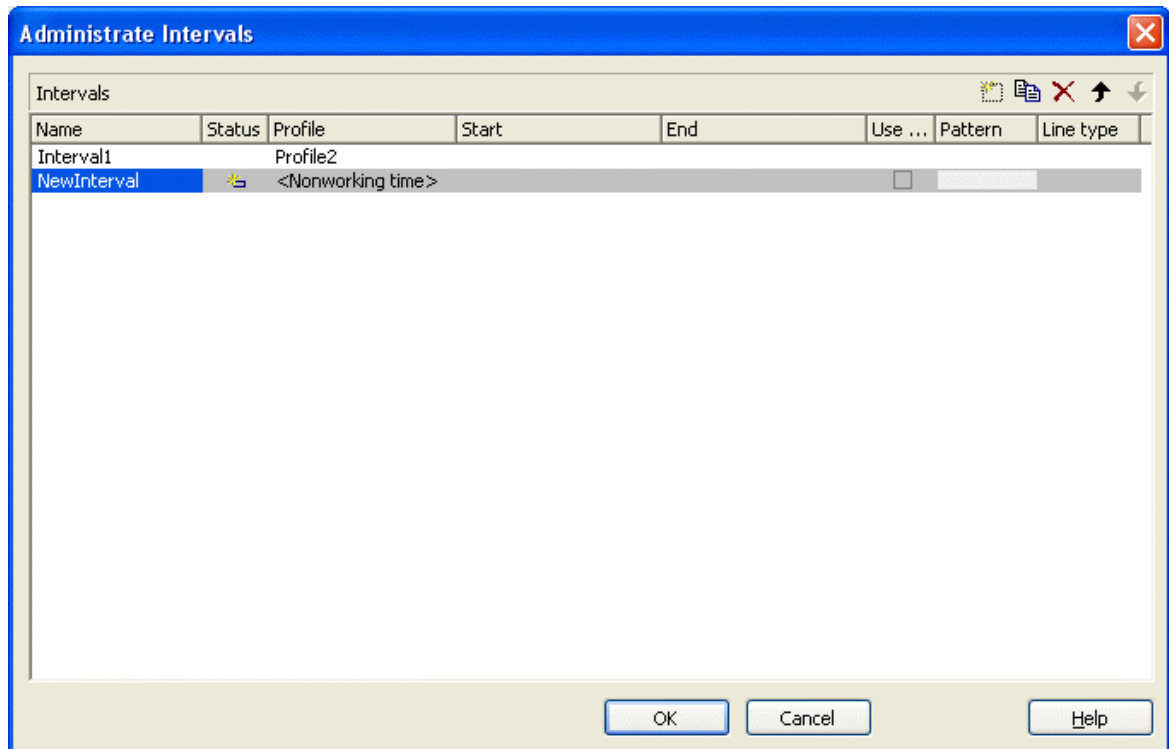
The marked calendar is deleted.

## **Edit calendar**



You will reach the **Edit Calendar** dialog box.

## 4.34 The "Administrate Intervals" Dialog Box (Calendar)



In this dialog box you can edit 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.

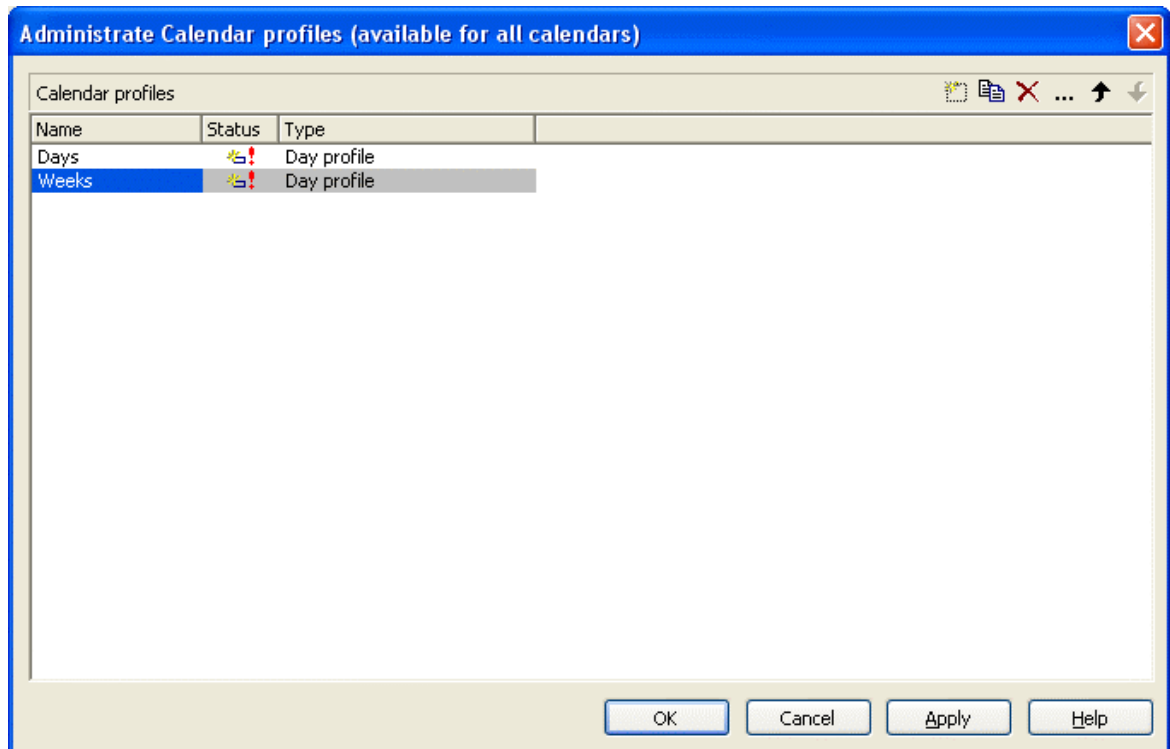
## **Copy interval**

 Click on this button to copy the marked interval.

## **Delete interval**

 Click on this button to delete the marked interval.

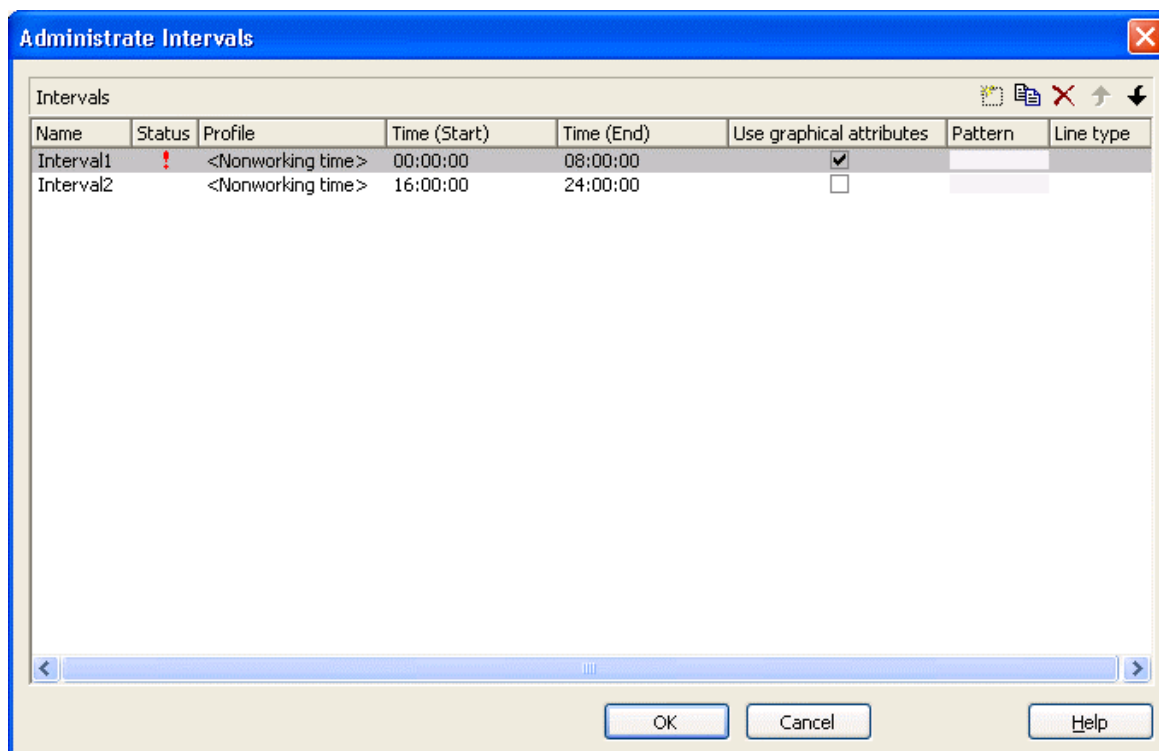
## 4.35 The "Administrate Calendar Profiles" Dialog Box



In this dialog you can create and modify calendar profiles.

**242** The "Administrate Intervals" Dialog Box (Calendar Profiles, Profile Type <Day Profile>)

## 4.36 The "Administrate Intervals" Dialog Box (Calendar Profiles, Profile Type <Day Profile>)

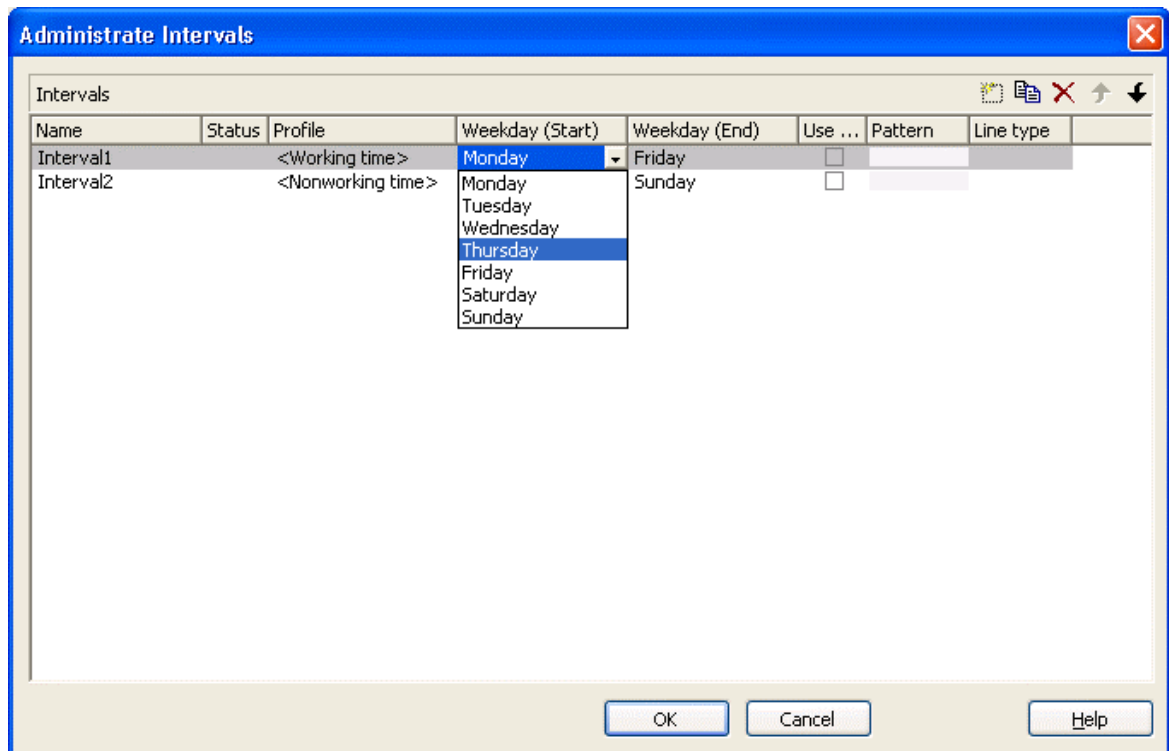


In this dialog box you can create and modify intervals. The properties that are offered vary depending on the profile type that has been selected in the <Administrate Calendar Profiles dialog.

### Name


Lists the names of all intervals. All names can be edited.

## 4.37 The "Administrate Intervals" Dialog Box (Calendar Profiles, Profile Type <Week Profile>)




In this dialog box you can create and modify intervals.

### Weekday Start/Weekday End

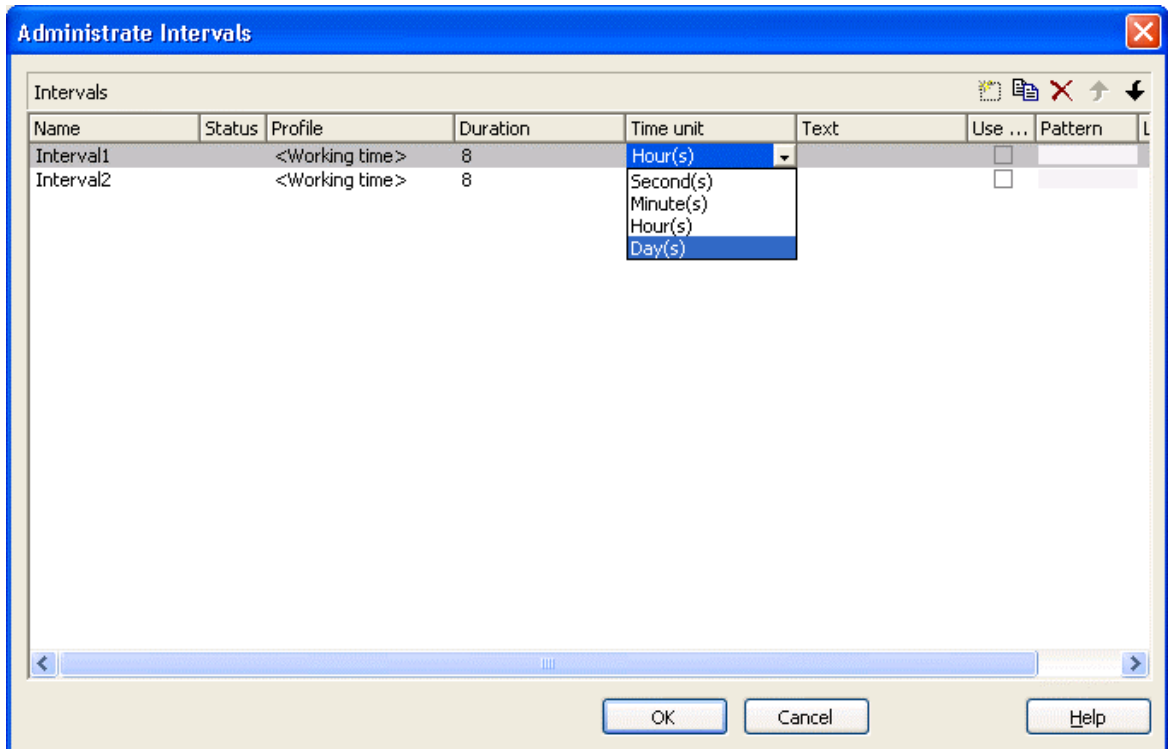
By clicking  you can set the first/last weekday of the interval.

### Weekday Start/Weekday End

By clicking  you can set the first/last weekday of the interval.

**244** The "Administrate Intervals" Dialog Box (Calendar Profiles, Profile Type <Variable Profile>)

## 4.38 The "Administrate Intervals" Dialog Box (Calendar Profiles, Profile Type <Variable Profile>)



In this dialog box you can create and modify intervals.

### Duration

Here you can specify the duration of the interval. This feature can also be set by the property **VcInterval.Duration**

### Duration

Here you can specify the duration of the interval. This feature can also be set by the property **VcInterval.Duration**

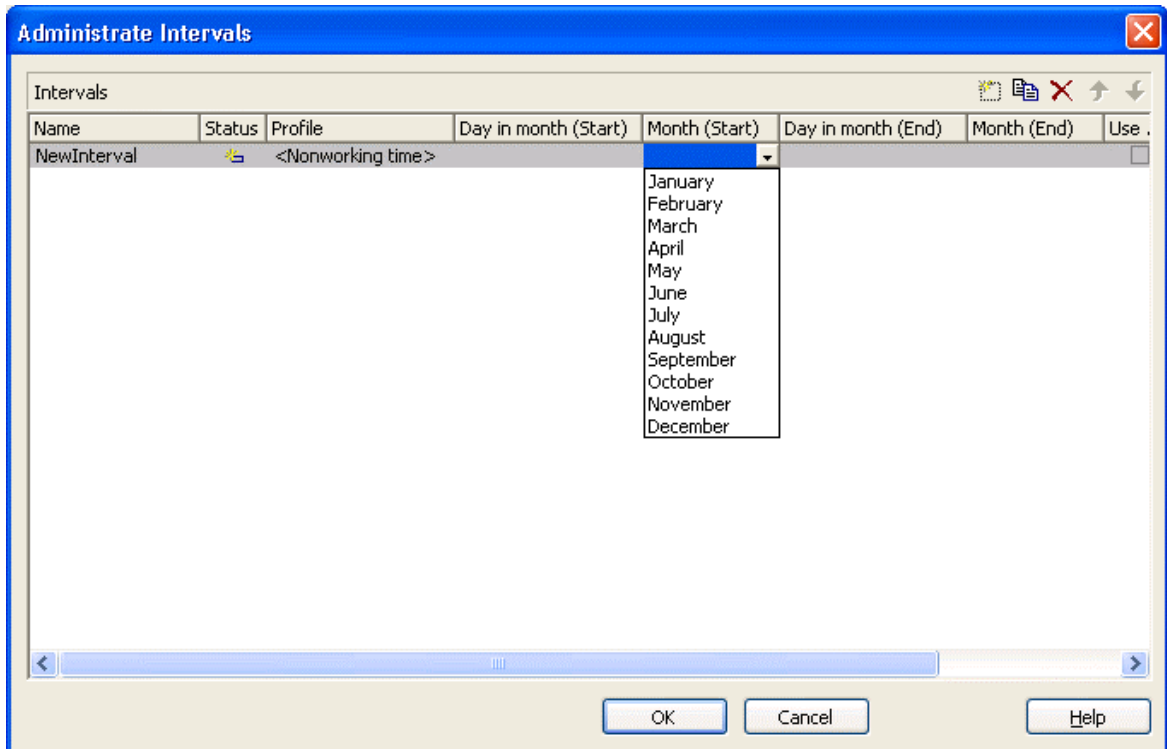
### Time unit

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**

## 4.39 The "Administrate Intervals" Dialog Box (Calendar Profiles, Profile Type <Year Profile>)



In this dialog box you can create and modify intervals.

### 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**


### 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)

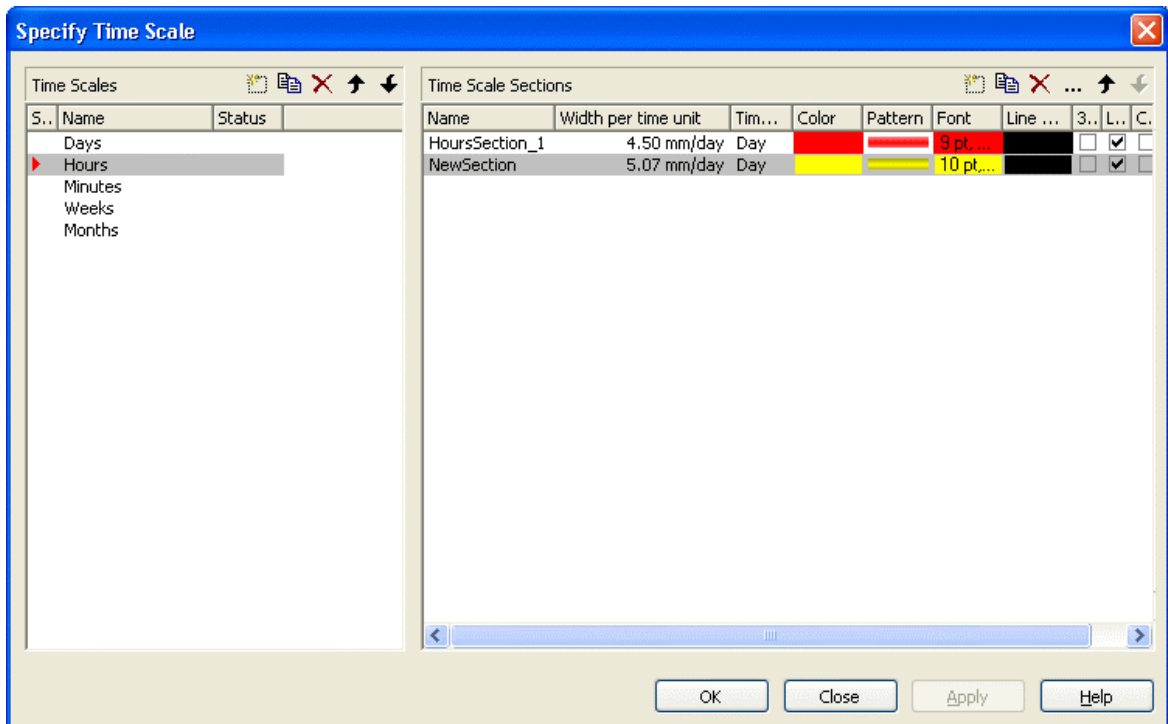
By clicking  you can set the day in the start/end month of the interval. This feature can also be set by the property **VcInterval.Start/EndMonth**

## 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**





## 4.40 The "Specify Time Scale" Dialog Box








You can reach this dialog box via the **Objects** property page. It allows to establish and modify time scales.

### 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.
- **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 / promote / demote time scale

     By these buttons you can create, copy or delete time scales or move them by one position up or down in the list, 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.
- **Color:** Click the button of this field () to open the Color Picker box, and choose a color for the section. If the ribbons had different colors, the color selected here will be applied to all sections.
- **Pattern** Select the fill pattern for the section.
- **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.
- **Line color:** Select a frame color for the time scale.
- **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

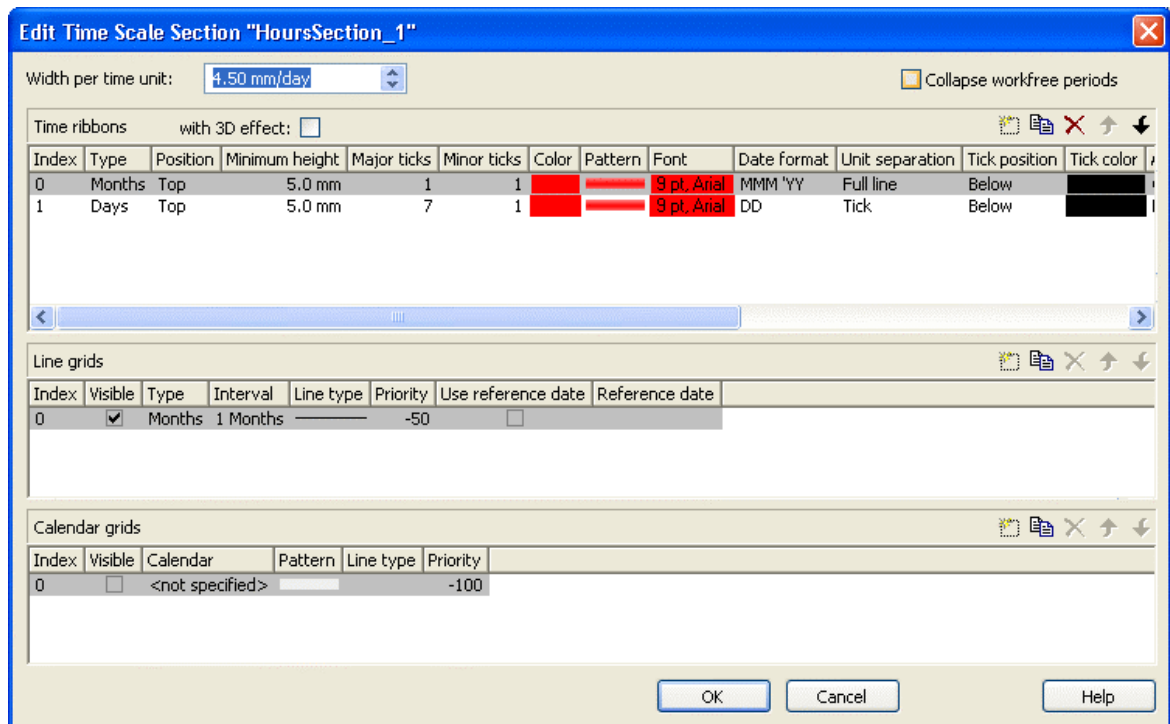


By these buttons you can create, copy, delete or edit time scales or move them in the table, respectively.

The position of the time scale sections in the table corresponds with their position in the diagram.

When creating a new section, all sections will be displayed with nearly the same extent. You can modify their size by using the mouse. The start dates of sections can be set and modified by API calls.

## 4.41 The "Edit Time Scale Section" Dialog Box



### Width per time unit

Specify the width allocated to each unit of the selected section. The new value is transferred to the corresponding field in the **Time Scale Sections** table in the **Specify Time Scale** dialog.

### Collapse workfree periods

Here you can set whether or not workfree periods should be displayed in this section. The calendar that defines the workfree periods is the one selected in the **Specify Calendars** dialog box.

When setting this feature, the new value will be copied to the corresponding field in the **Time Scale Sections** table in the **Specify Time Scale** dialog.





### Time 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.
- **Color:** Shows the background color of the ribbon. If not set, the ribbons of the section by default show the background color set in the **Specify Time Scale** dialog. To assign a different color to a ribbon, please click on the  button of the **Color** field of the ribbon to get to the color picker. The color that you set to the first ribbon of a section will be copied to the **Color** field in the **Time Scale Sections** table in the **Specify Time Scale** dialog.
- **Pattern:** Lets you set the pattern of the ribbon. If not set, the ribbons of the section by default show the pattern set in the **Specify Time Scale** dialog. To assign a different pattern to a ribbon, click on  button in the **Pattern** field of the ribbon to get to the **Pattern dialog**. The pattern that you set to the first ribbon of a section 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 <b>VcTextEntrySupplying</b> )
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 <b>VcTextEntrySupplying</b> )
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 <b>VcTextEntrySupplying</b> )
Q:	one-digit figure for the quarter: 1-4
TQ:	name of quarter (adjustable by using the event <b>VcTextEntrySupplying</b> )
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 <b>VcTextEntrySupplying</b> )
TH:	"am" or "pm" (adjustable by using the event <b>VcTextEntrySupplying</b> )
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

## 254 The "Edit Time Scale Section" Dialog Box

- 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 separator 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 checkbox 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.

## Line grids

In the diagram area and in the histogram, one or more line grids 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 checkbox 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.

## Calendar grids



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.

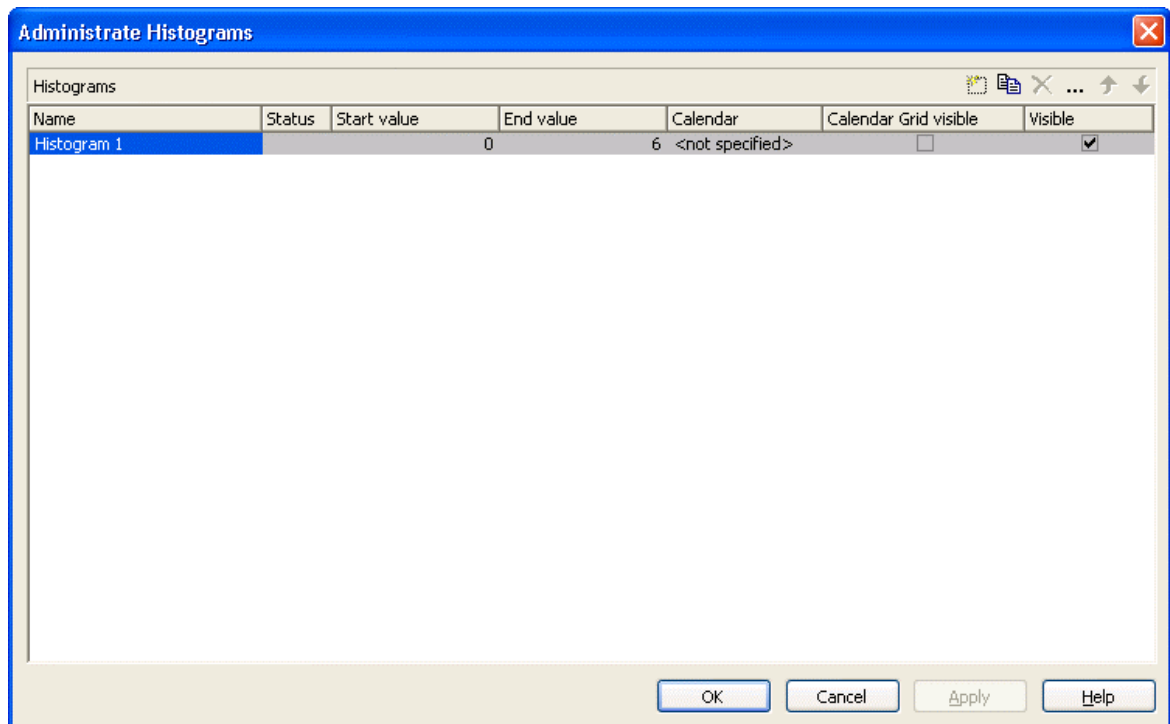
The table lets you modify the settings of the calendar grids:

- **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).

## 4.42 The "Administrate Histograms" Dialog Box



You can get to this dialog box via the **Layout** property page.

You can establish and modify histograms and select which ones 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.

### 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.

## Visible

Tick this box if you want the selected histogram to be displayed.

## Add histogram



A new histogram is created.

## Copy histogram



Copies the selected histogram.

## Delete histogram



The marked histogram is deleted.

## Edit histogram



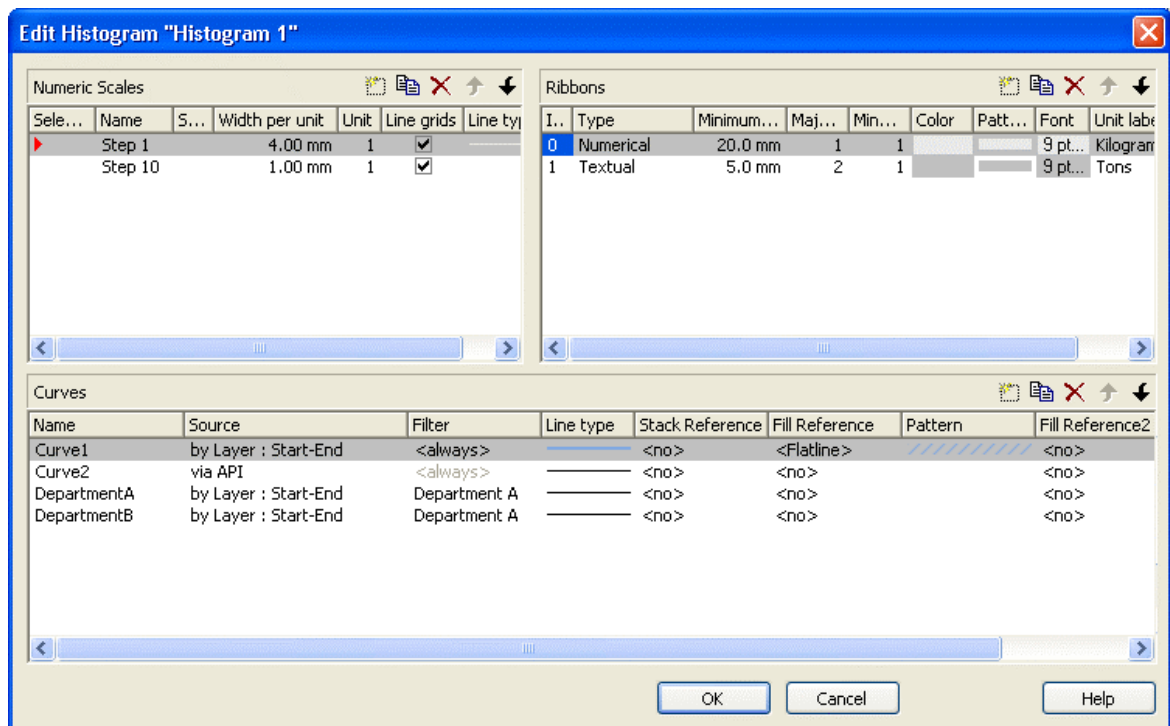
The **Edit Histogram** dialog box will appear.

## Promote / demote histogram



By these buttons you can create, copy or delete the histogram or move it by one position up or down in the list, respectively. The order of the histograms in the list equals their order of output.

## 4.43 The "Edit Histogram" Dialog Box



This dialog box will appear if in the **Administrate 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. Via 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 opening the dialog box is marked by a symbol.
- **Width per Unit** in mm
- **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 ( ... ). 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** minimum 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.
- **Color:** of the ribbon
- **Pattern:** for the pattern, you can select various types of color gradients.
- **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.
- **Unit label:** annotation of the label units of the numeric scale.

## Curves

- **Name:** In this column, the names of the curves available are listed.
- **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. With the help of a filter these activities can be specified further.
  2. **by API:** By this option, the values are set by the API. In the API, the values for a histogram curve can be freely defined using 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, specify a filter for each curve to specify the activities that contribute to 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:** To stack histogram curves, for each curve, in the **Stack Reference** field specify the curve on which you want the current curve to be stacked. If you do not want to stack a particular curve, select the entry <No> for that curve blank. If you select the entry <No> in the **Stack Reference** field for all curves, they will not be stacked, but will overlap each other instead. In order to still be able to differentiate between the curves, assign them different patterns.
- **Fill Reference:** This field allows you to specify how far down the fill pattern below a 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 a curve in the **Fill Reference** field, the fill pattern will will the area down to the curve.
- **Pattern:** Specify the pattern below a curve. By the **Edit** button (  ) you can open the **Pattern** dialog box to set 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) exceed the y values of the second reference curve.
- **Pattern 2:** Set the pattern and the color of the filling above the second reference curve.

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**

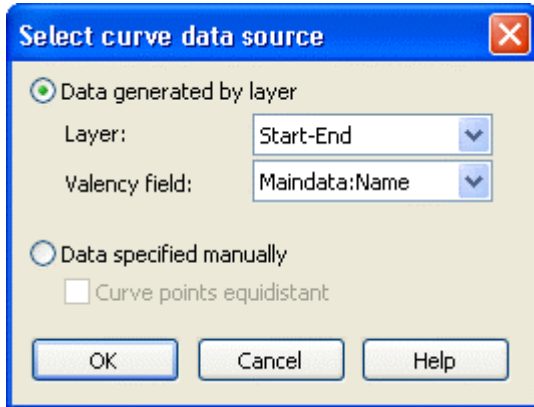
 The selected object is deleted.

### **Promote/demote numeric scale/ribbon/curve**

  By these buttons you can move the selected object by one position up or down in the list, respectively.

---

## 4.44 The "Select Curve Data Source" Dialog Box



You can get to this dialog via 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

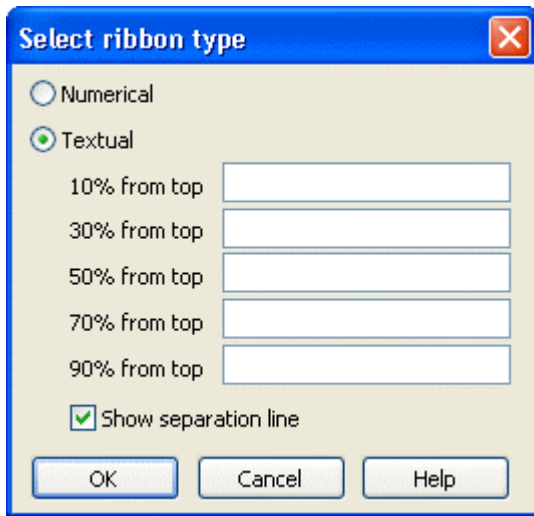
Select this option, if the data are to be specified manually. For this option you may choose the option **Curve points equidistant**. Otherwise the curve points will be created only in those points where the y values are changing.

For further information please see the chapter "Important Concepts: Histograms".



---

## 4.45 The "Select Ribbon Type" Dialog Box



You can get to this dialog via the **Edit Histogram** dialog.

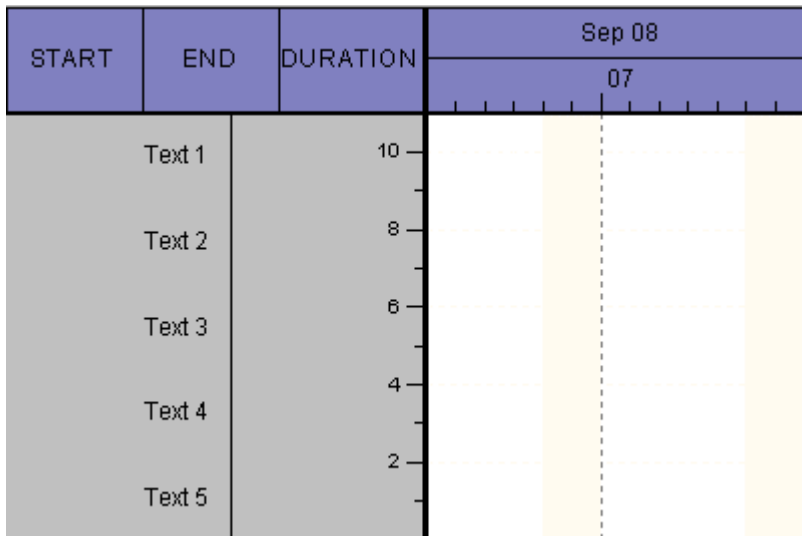
### Numerical

Select this option if the current ribbon of the numeric scale is to be annotated with numbers.

### Textual

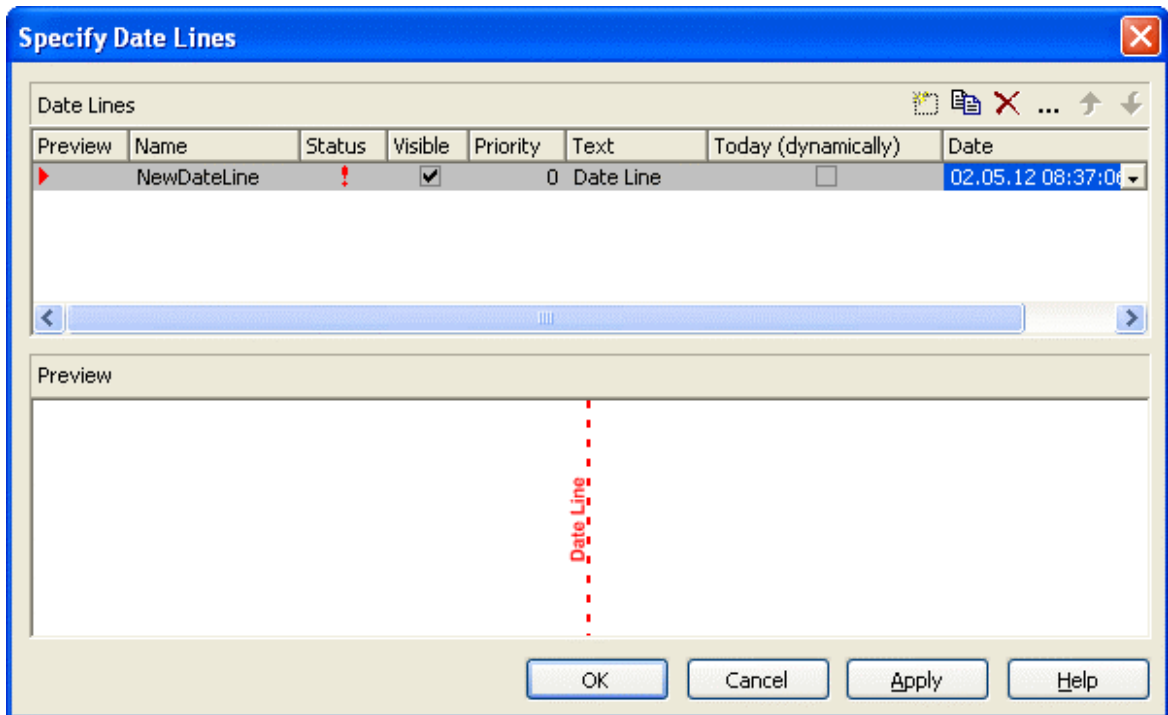
Select this option if the current ribbon of the numeric scale is to be annotated with texts which you can specify for five predefined positions (10%, 30%, 50%, 70 %, 90 % from top).

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*

## 4.46 The "Specify Date Lines" Dialog Box



Date lines (vertical lines in the diagram) let you highlight specific dates (the actual date or any other date) in your diagram. This dialog box allows to create or delete date lines in your chart and to set options to them. You can invoke this dialog on the **Objects** property page.

### 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.

### Visible

Activate this check box, if 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.

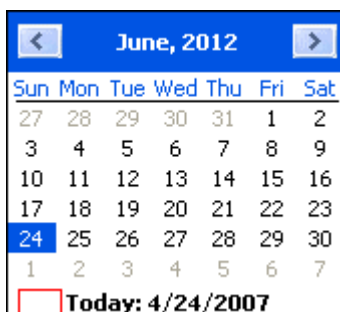
## Today (dynamically)

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.

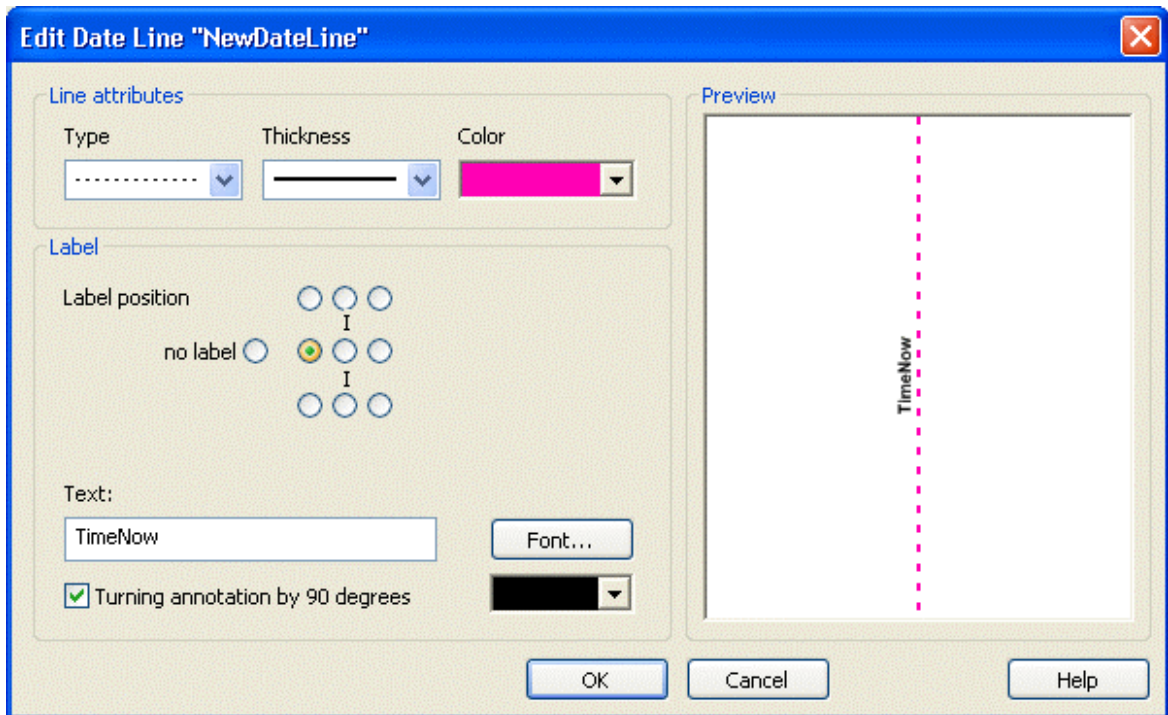
Alternatively, you can set the date by the date control. For this, please click on the arrow button (). The **date** dialog box will appear where the selected date is highlighted. If no date was selected, the current date is highlighted. Select a day from the month displayed. You can flip through the 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.



## Add / copy / delete / edit / promote / demote date line

     By these buttons you can create, copy or delete a date line or move it by one position upward or downward in the list.

## 4.47 The "Edit Date Line" Dialog Box



### 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.

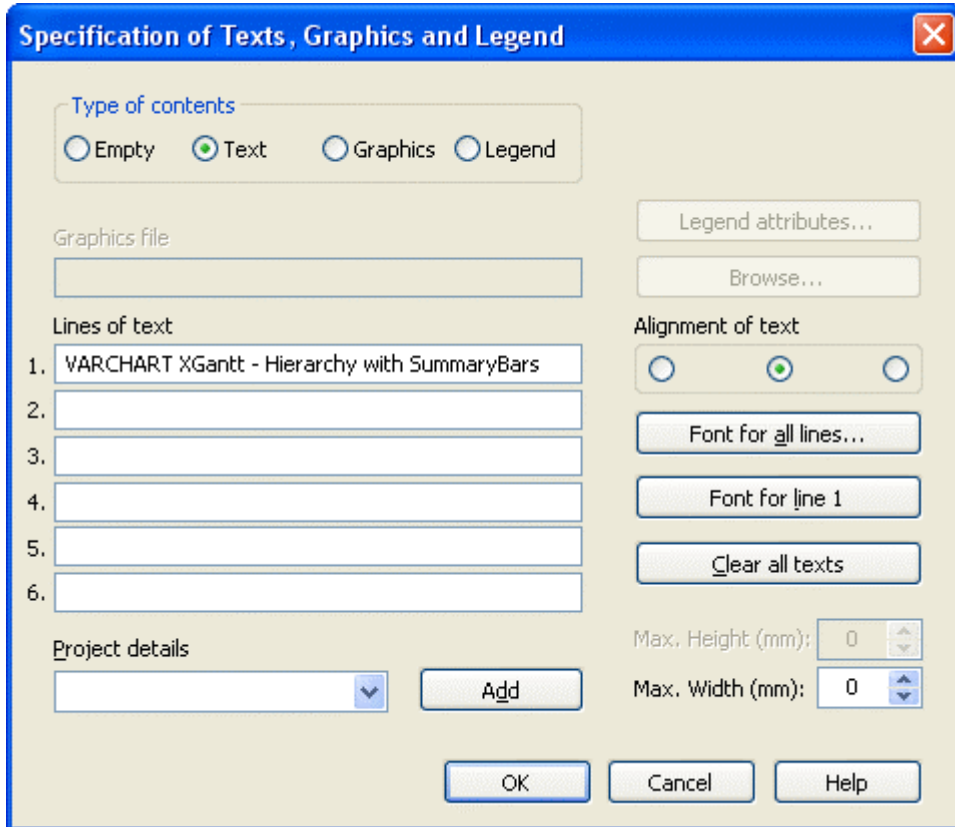
## **Font**

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.

## **Rotating an annotation by 90 degrees**

Activate this check box, if the annotation should be displayed in vertical direction.

## 4.48 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 you want to display at the chosen position:

**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 file (selected by the **Browse** button) will be displayed at the chosen location. Graphics are always positioned in the center.

**Legend:** A legend will be displayed at the chosen location. It describes the layers used in the 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 more attributes for the legend.

## **Graphics file**

*Only activated if the check box **Graphics** was ticked.* Select the graphics file to be displayed by clicking on the **Browse** button or enter the file name in the field manually. If the selected graphics file is not stored in the installation directory of the VARCHART web server, please also specify the drive and the directory.

## **Browse**

*Only activated if the check box **Graphics** was 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 if the check box **Text** was 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 if the check box **Text** was 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 clicking on the **Add** button.

The place holders will be replaced by the corresponding and up-to-date data in the print preview or on the printout.

## **Add**

*Only activated if the check box **Text** was 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 if the check box **Text** was ticked.* Specify whether the text lines should be output left-aligned, centred or right-aligned.

## **Font for all lines**

*Only activated if the check box **Text** was 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 if the check box **Text** was 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 if the check box **Text** was ticked.* Click on this button to delete the contents of all six lines of text.

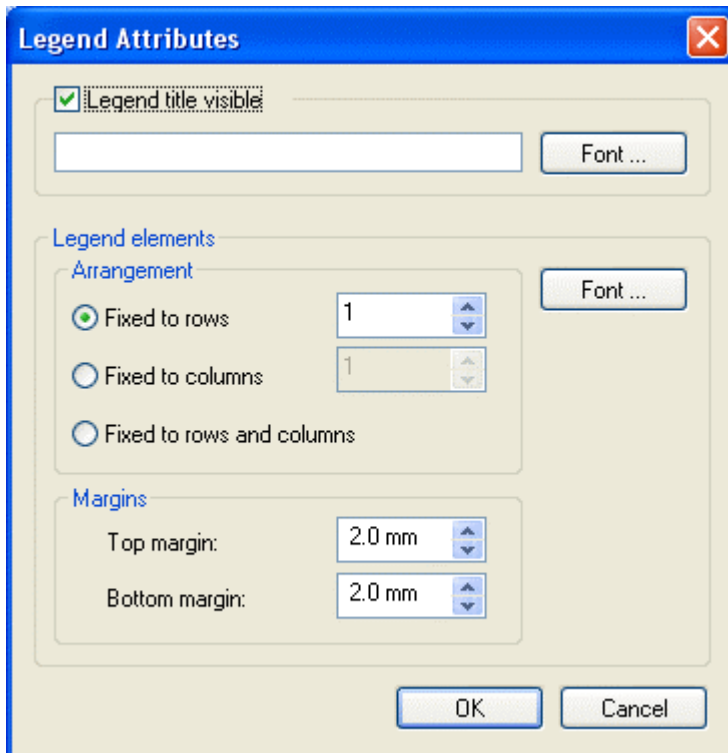
## **Max. Height (mm)**

*Only activated if the check box **Graphics** was 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 if the check box **Text** or **Graphics** was 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.

## 4.49 The "Legend Attributes Dialog Box"



You can reach this dialog at runtime by clicking the corresponding item of the legend's contextmenu or at designtime 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.

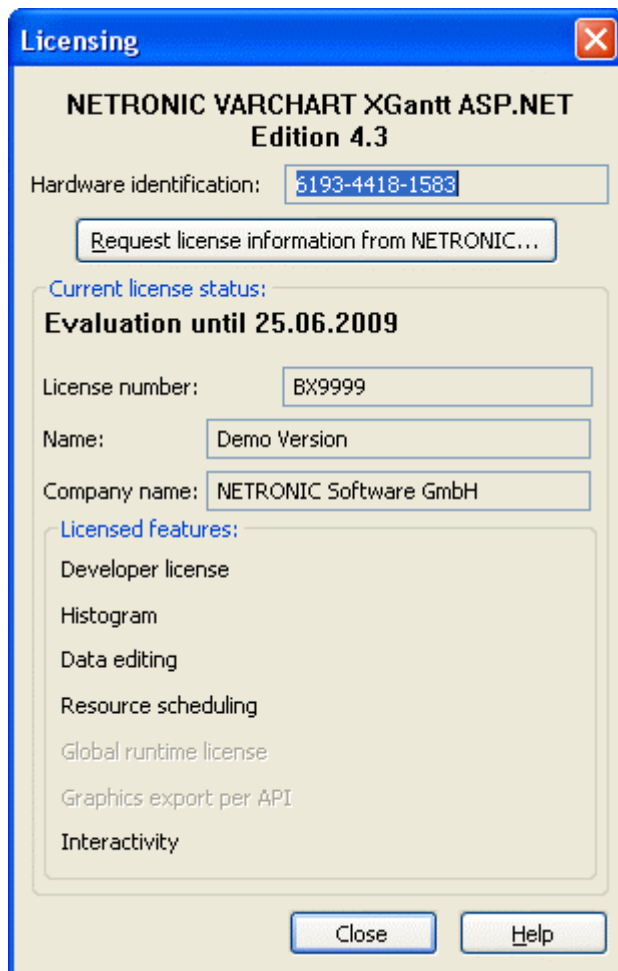
## **Margins**

- 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.

## 4.50 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" water mark.

### Hardware identification

*(cannot be edited)* The number indicated in this field is calculated from your hardware configuration. It is required by NETRONIC Software GmbH for the licensing procedure. When changing your hardware, you need to renew your license. Please do not hesitate to contact the support team of NETRONIC.

## Request license information from NETRONIC

For licensing, click on this button, which will get you to the **Request License Information** dialog.

### License number/Name/Company name

*(cannot be edited)* Indicates your license number, your name and the name of your company.

### Licensed features

The modules that have been licensed are indicated. If the licensing procedure was successful, the licensed modules are activated.

- **Developer license**
- **Histogram**
- **Data editing** (provides all functions of editing application data)
- **Resource scheduling** (requires all other modules and provides all functions for resource scheduling)
- **Global runtime license** (In the runtime mode, the VARCHART web server control can be run on any computer.)
- **Single-place runtime licenses** (The VARCHART web server control has to be licensed individually for the computer on which it should run.)
- **Graphics export per API**
- **Interactivity**

### Close

Quits the dialog box.

## 4.51 The "Request License Information" Dialog Box

**Request License Information**

**NETRONIC VARCHART XGantt ASP.NET Edition 4.3**

Hardware identification: 6193-4418-1583

First step: Enter your user information below:

License number: |

Name: |

Company name: |

Second step: Request your license information:

If you cannot send emails from your computer, contact NETRONIC Software GmbH by stating the four entries above:

email: license@netronic.com  
phone: +49/2408/141-0  
fax: +49/2408/141-33

Third step: After receiving the license information file, copy it into the directory of the DLL file.

Please 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 (**NETRONIC.XGantt.Web.VcGantt.lic**) and mail it back to you.

After having received the file, please copy it to the directory in which the file **NETRONIC.XGantt.Web.dll** 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 Frequently Asked Questions**

---

### **5.1 What Needs to be Done if the .NET-Framework was Installed, but still an Error Message Occurs Saying that the Wrong Version of .NET was Installed?**

This may occur for instance if the Internet Information server was installed after the .NET framework. In this case it will be sufficient to register anew the file aspnet\_isapi.dll (located in the .NET framework folder below the windows folder) by the call regsvr32 aspnet\_isapi.dll. Or you run the file aspnet\_regiis.exe in the directory "%SystemRoot%\Microsoft.NET\Framework\>"



---

## 5.2 Does VARCHART XGantt ASP.NET Require the Session State?

It positively does. Each page re-load – for instance after a mouse click – involves a re-load of the controls. The HTTP protocol lacks a memory; so in the first instance, nothing is stored to the server. This is why the kernel of VARCHART XGantt ASP.NET is stored in a session variable. It ensures the conservation of settings and loaded data from one HTTP request to the next one, so that intermediate updates are not needed.

---

## 5.3 What Type of Storage of the Session State does VARCHART XGantt ASP.NET Require?

ASP.NET supports three process models to store the session state:

1. In-Proc
2. State Server
3. SQL Server

VARCHART XGantt ASP.NET only works with the model „In-Proc“. The session state can be set for the server in the file "Machine.config" and separately for your application in the file "Web.config" by the entry `<sessionState mode="InProc"/>`.

---

## 5.4 What Is the Relation Between Sessions and Cookies ?

A VARCHART XGantt ASP.NET session requires cookies. If cookies are deactivated in the browser, the access to to the session state can only work if in the file **Web.config** of the application the attribute **cookieless** in the object **sessionState** was set to **true**.

---

## 5.5 What Needs to be Considered When Loading Data?

In a web application, the PageLoad method is invoked by the framework each time before a page is composed. Therefore, after a click on a page and after the associated request was launched the code of that method is executed. If you integrate the loading of data in the PageLoad method, you need to ensure not to load the data each time the PageLoad method is invoked, since modifications - such as new nodes or different dates - might get lost. Therefore, loading within the PageLoad method needs to be implemented as shown below:

### Example Code

```
if ( ! IsPostBack )
{
    vcGanttASP1.InsertNodeRecord(...);
    vcGanttASP1.InsertNodeRecord(...);
    vcGanttASP1.InsertNodeRecord(...);
    vcGanttASP1.EndLoading(...);
}
```

Please make sure that the EndLoading method is invoked only once.

---

## 5.6 What Can I do if in the Web Application or the Programming Environment a Fatal Error Occurs?

The aspnet\_wp process should be finished by using the TaskManager or a similar tool (for instance the ProcessExplorer), preferably immediately before starting the application.

## 5.7 How can I Move a Bar into the Visible Area by Clicking on the Table?

The event **VcNodeLeftClicking** 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 VcGantt object using the **ScrollToDate** method.

### Example Code VB.NET

```
Private Sub VcGantt1_VcNodeLeftClicking(ByVal sender As Object, ByVal e
As NETRONIC.XGantt.VcNodeClickingEventArgs) Handles
VcGantt1.VcNodeLeftClicking

    Dim myDataDef As VcDataDefinition
    Dim myDataDefTable As VcDataDefinitionTable
    Dim myDataField As VcDataDefinitionField
    Dim myIndex As Integer
    Dim location As VcLocation

    If (location = VcLocation.vcInTable) Then
        ' in case the Index of the "Start" field is not known
        myDataDef = VcGantt1.DataDefinition
        myDataDefTable =
myDataDef.DefinitionTable(VcDataTableType.vcMaindata)
        myDataField = myDataDefTable.DataDefinitionFieldByName("Start")
        myIndex = myDataField.ID
        VcGantt1.ScrollToDate(e.Node.DataField(myIndex),
VcHorizontalAlignment.vcLeftAligned, 2)
    End If

End Sub
```

### Example Code C#

```
private void VcGantt1_VcNodeLeftClicking(object sender,
NETRONIC.XGantt.VcNodeClickingEventArgs e)
{
    VcDataDefinition myDataDef = VcGantt1.DataDefinition;
    VcDataDefinitionTable myDataDefTable =
myDataDef.get_DefinitionTable(VcDataTableType.vcMaindata);
    VcDataDefinitionField myDataField =
myDataDefTable.DataDefinitionFieldByName("Start");
    int myIndex = myDataField.ID;
    VcLocation location = VcLocation.vcInTable;

    if (Location.ToString().Length > 0)
        VcGantt1.ScrollToDate(Convert.ToDateTime(e.Node.get_DataField(2)),
VcHorizontalAlignment.vcLeftAligned, 2);
}
```

---

## 5.8 How can I Make Overlapping Activities in a Group Visible?

To avoid bottlenecks in holiday rosters or for machine allocations, overlapping activities in a group can be made visible.

Activities can overlap if the activities were grouped and in the **Grouping** dialog the **Nodes in separate rows** option is **not** selected. By using the **Nodes in one line** option, the activity groups can be collapsed and expanded. If a group is collapsed, overlapping activities are invisible while in expanded groups, they are displayed as staggered piles to indicate the overlapping.

To make overlapping activities in a group visible, select the **Nodes in one line** option on the **Sorting** property page 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 not activated, allowing to detect any collisions at a glance.

If the activities are collapsed, overlapping activities cannot be detected. If 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.

---

## 5.9 How can I Save and Reload the Order 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, please open the **Nodes** property page and select a data field from **Row number field**. XGantt will store the identification to this data field. If the order of the nodes was modified interactively, you can update it by using the method **UpdateRowNumberField**. It requires grouping and the hierarchy to be deactivated.

Finally, please add the below code:

### Example Code VB.NET

```
Private Sub Form_Unload ()
    VcGantt1.UpdateRowNumberField
    VcGantt1.SaveAs ("file name.csv")
End Sub
```

### Example Code C#

```
private void Form_Unload()
{
    VcGantt1.UpdateRowNumberFields;
    VcGantt1.SaveAs("file name.csv");
}
```



---

## 5.10 How can I Improve the Performance?

### ► Suspend update

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 improve the overall performance considerably.

#### Example Code VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
dataRecordCltn = dataTable.DataRecordCollection

VcGantt1.SuspendUpdate(True)

For Each dataRecord In dataRecordCltn
    dataRecord.DataField(4) = "10"
    dataRecord.Update()
Next

VcGantt1.SuspendUpdate(False)
```

#### Example Code C#

```
VcDataTable dataTable =
vcGantt1.DataTableCollection.DataTableByName("Maindata");
VcDataRecordCollection dataRecordCltn = dataTable.DataRecordCollection;

vcGantt1.SuspendUpdate(true);

foreach (VcDataRecord dataRecord in dataRecordCltn)
{
    dataRecord.set_DataField(4, "10");
    dataRecord.Update();
}

vcGantt1.SuspendUpdate(false);
```

You can also accelerate the updating procedure of links via the **SuspendUpdate** method.

If you modify table formats in large projects, you also should use the **SuspendUpdate** method.

**Example Code VB.NET**

```

Private Sub ModifyTable_Click()

    Dim formatCltn As VcTableFormatCollection
    Dim aFormat As VcTableFormat
    Dim index As Integer

    VcGantt1.SuspendUpdate(True)

    formatCltn = VcGantt1.LeftTable.TableFormatCollection
    For Each aFormat In formatCltn
        For index = 1 To aFormat.FormatFieldCount
            aFormat.FormatField(index).BackColor = Color.Green
            aFormat.FormatField(index).TextFontColor = Color.Red
            aFormat.FormatField(index).Alignment =
VcFormatFieldAlignment.vcFFACenter
        Next
    Next

    VcGantt1.SuspendUpdate(False)

End Sub

```

**Example Code C#**

```

private void ModifyTable_Click()
{
    VcTableFormatCollection formatCltn =
vcGantt1.LeftTable.TableFormatCollection;

    vcGantt1.SuspendUpdate(true);

    foreach (VcTableFormat aFormat in formatCltn)
    {
        for (int index=1; index <= aFormat.FormatFieldCount; index++)
        {
            aFormat.get_FormatField(index).BackColor = Color.Green;
            aFormat.get_FormatField(index).TextFontColor = Color.Red;
            aFormat.get_FormatField(index).Alignment =
VcFormatFieldAlignment.vcFFACenter;
        }
    }

    vcGantt1.SuspendUpdate(false);
}

```

**► Graphics**

Another reason for a low performance may be graphics in table, node or box fields that are too large or that have too many pixels.

---

## 5.11 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 "Delivery" at the beginning of this book.

---

## 5.12 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 the VARCHART control. 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.

---

## 5.13 How to find the Name of the IIS Worker Process

When using IIS (Internet Information Service) 5.0, the name of the worker process is aspnet\_wp.exe while when using IIS 6.0, the name is w3wp.exe. If the IIS 6.0 is run in the IIS 5.0 isolation mode, the name is aspnet\_wp.exe. You can set the isolation mode in the IIS manager by selecting web sites > properties > services.

---

## 5.14 What if an ASPX Page is Displayed as a HTML Page?

In this case the IIS (Internet Information Service) does not know that the ASPX suffix refers to an ASP.NET page. This may happen for example, if the .NET framework was installed prior to installing the IIS. Please execute the below command:

**C:\Windows\system32\inetsrv\iisrstat.exe /RegServer**

The command **regsvr32 aspnet\_isapi.dll** may also help.

Or you run the file `aspnet_regiis.exe` in the directory `"%SystemRoot-%\Microsoft.NET\Framework\>"`.

---

## 5.15 How can Jittering and "Scrolling Away" be Avoided When Clicking on the Control?

Jittering only occurs with the Internet Explorer, with other browsers (for example Mozilla Firefox) the display remains stable. To stop the jittering when using ASP.NET 1.1, the property **SmartNavigation** has to be activated. For this, in the `@Page` directive of the corresponding ASPX page you can set the attribute **SmartNavigation** to **true**.

SmartNavigation also keeps the position stable by avoiding the website to scroll to the top after clicking. This feature is not supported by Mozilla Firefox. If SmartNavigation does not work although activated, one of the below options may be the cause:

1. The name of the ASPX page contains white blanks or special characters.
2. The server does not have a read permission of the file `SmartNav.js` in the folder `wwwroot\aspnet_client\system_web\1_1_4322\SmartNav.js`. To whom of the user the permission is to be given, can be found for example by a file monitor.

From ASP.NET 2.0 onward, you should use an Ajax framework as for example Microsoft's ASP. NET AJAX, to get a jitter-free display.

Activating the AJAX extensions also avoids scrolling and flashing of a page, if the `VARCHART XGantt` ASP.NET control is clicked upon. If other sections are affected by this (for example a calendar control), they need to be put in an update panel. When using AJAX with ASP.NET 1.1, you should not use SmartNavigation simultaneously.

---

## 5.16 Why Does the Message "Page not found" Occur When Invoking a HTML or an ASPX Page, Although the Page Does Exist?

The IIS (Internet Information Service) does not work. By the menu items control panel > Administrative Tools > Component Services the service **WWW Publishing** needs to have the state **started**. If it doesn't, it needs to be started anew.



---

### 5.17 How Can a URL be Put on a Node?

The below code sample demonstrates, how by clicking on a node a different website is invoked. You can certainly use the same way to link other objects to URLs, for example layers.

#### Example Code

```
private void vcGanttASP1_VcNodeLeftClicking(object sender,
NETRONIC.XGantt.Web.VcNodeClickingEventArgs e)
{
    // Here you can e.g. retrieve a data record entry from the clicked
node,
    // which is interpreted as a URL.
    this.Response.Clear();
    if (vcGanttASP1.IsCallBack)
    {
        this.Response.Write("{\"controls\" : [{ \"cmd\" :
\"window.location='http://www.netronic.de'\"}] }");
        this.Response.End();
    }
    else
        this.Response.Redirect("http://www.netronic.de");
}
```

## 5.18 How Can a Reload of a Page be Initialized When a Session Timeout Occurs?

If the session state of a page does not exist any more because, for instance, the validity period has expired, the control throws an exception (**NETRONIC.XGantt.Web.VcSessionStateExpiredException**) so that the page cannot be displayed properly any more. Yet, by adding a few lines of code, you may at least induce a reloading of the page with initial values:

### Example Code

```
protected override void OnError(System.EventArgs e)
{
    System.Exception lastError = Server.GetLastError();
    String type = lastError.GetType().ToString();
    if
    (type.Equals("NETRONIC.XGantt.Web.VcSessionStateExpiredException"))
    {
        if (this.IsPostBack)
        {
            this.Server.ClearError();
            this.Response.Clear();
            if (VcGantt1.IsCallBack)
            {
                this.Response.Write("{ \"controls\" :
                [{cmd: \"alert('SessionState
                expired!');window.location.reload();\"}]})");
            }
            else
            {
                this.Response.Redirect(this.Request.Url.AbsolutePath);
            }
        }
    }
}
```

---

## 5.19 How to Use XGantt Best Together With Microsoft's ASP .NET AJAX ?

Instances of the XGantt ASP .NET control should not be placed in an UpdatePanel of the ASP .NET Ajax extensions when using Microsoft's ASP .NET AJAX, because they will jitter when an Ajax callback is carried out. If you want your chart to be updated as well when a callback is carried out, please enter the following code in the **Page\_Load**- method of your ASP .NET page and replace "vcGantt1" with the name of your XGantt instance.

### Example Code

```
protected void Page_Load(object sender, System.EventArgs e)
{
    ...
    ClientScript.RegisterClientScriptBlock(this.GetType(), "jsUpdateGantt",
    "function EndRequestHandler(sender, args) {" +
    "if (args.get_error() == undefined) {" +
    "vcGantt1.update();}" +
    "}\n" +
    "function updateGantt() {" +
    "Sys.WebForms.PageRequestManager.getInstance().add_endRequest(EndRequest
    Handler);" +
    "}\n" +
    "window.onload = updateGantt;", true);
    ...
}
```

---

---

## 6 API Reference

---

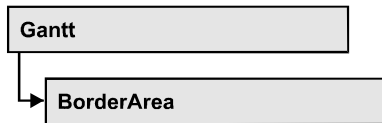
### 6.1 Object Types

- VcBorderArea
- VcBorderBox
- VcBox
- VcBoxCollection
- VcBoxFormat
- VcBoxFormatCollection
- VcBoxFormatField
- VcCalendar
- VcCalendarCollection
- VcCalendarGrid
- VcCalendarProfile
- VcCurve
- VcCurveCollection
- VcDataDefinition
- VcDataDefinitionField
- VcDataDefinitionTable
- VcDataRecord
- VcDataRecordCollection
- VcDataTable
- VcDataTableCollection
- VcDataTableField
- VcDataTableFieldCollection
- VcDateLine
- VcDateLineCollection
- VcDateLineGrid
- VcFilter
- VcFilterCollection
- VcFilterSubCondition
- VcGantt
- VcGroup
- VcGroupCollection
- VcGroupLevelLayout

## 300 API Reference: Object Types

- VcGroupLevelLayoutCollection
- VcHistogram
- VcHistogramCollection
- VcInterval
- VcIntervalCollection
- VcLayer
- VcLayerCollection
- VcLayerFormat
- VcLayerFormatField
- 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

## 6.2 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
<b>Parameter:</b> boxPosition	VcBorderBoxPosition  <b>Possible Values:</b> .vcBBXPBottomBottomCentered 8 .vcBBXPBottomBottomLeft 7 .vcBBXPBottomBottomRight 9 .vcBBXPBottomTopCentered 5 .vcBBXPBottomTopLeft 4 .vcBBXPBottomTopRight 6 .vcBBXPLegend 51 .vcBBXPTopCentered 2 .vcBBXPTopLeft 1 .vcBBXPTopRight 3	Box position  second line in the bottom area, centered second line in the bottom area, left second line in the bottom area, right first line in the bottom area, centered first line in the bottom area, left first line in the bottom area, right legend top centered top left top right
<b>Return value</b>	VcBorderBox	Box of the title and legend area

### Example Code VB.NET

```

Dim boardArea As VcBorderArea
Dim bBoxBBL As VcBorderBox

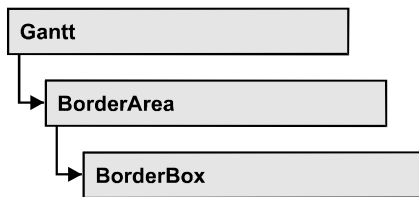
boardArea = VcGanttASP1.BorderArea
bBoxBBL = boardArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomBottomLeft)
bBoxBBL.LegendTitle = "Explanation"
  
```

## 302 API Reference: VcBorderArea

### Example Code C#

```
VcBorderArea boardArea = vcGanttASP1.BorderArea;  
  
VcBorderBox bBoxBBL =  
boardArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomBottomLeft);  
bBoxBBL.LegendTitle = "Explanation";
```

## 6.3 VcBoundingBox



An object of the type **VcBoundingBox** designates one of the boxes in the title or legend area of the graphics.

### Properties

- GraphicsFileName
- LegendElementsArrangement
- LegendElementsBottomMargin
- LegendElementsMaximumColumnCount
- LegendElementsMaximumRowCount
- LegendElementsTopMargin
- LegendFont
- LegendTitle
- LegendTitleFont
- LegendTitleVisible
- Text
- TextFont
- Type

---

## Properties

### GraphicsFileName

**Property of VcBoundingBox**

This property lets you set or retrieve the name of the graphics file used in the VcBoundingBox object. *Available formats:*

- \*.BMP (Microsoft Windows Bitmap)
- \*.EMF (Enhanced Metafile)
- \*.EMF, with EMF+ included



## 304 API Reference: VcBorderBox

- \*.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, 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	System.String	Name of the graphics file

### Example Code VB.NET

```
Dim borderArea As VcBorderArea
Dim borderBox As VcBorderBox

borderArea = VcGanttASP1.BorderArea
borderBox = borderArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomTopRight)
borderBox.Type = VcBorderBoxType.vcBBXTGraphics
borderBox.GraphicsFileName = "C:\Asterix.jpg"
```

### Example Code C#

```
VcBorderArea borderArea = vcGanttASP1.BorderArea;
VcBorderBox borderBox =
borderArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomTopRight);
borderBox.Type = VcBorderBoxType.vcBBXTGraphics;
borderBox.GraphicsFileName = @"C:\Asterix.jpg";
```

## LegendElementsArrangement

Property of VcBorderBox

This property lets you set or retrieve the arrangement of the elements in the legend.

	Data Type	Explanation
<b>Property value</b>	VcLegendElementsArrangement	Type of arrangement of the legend elements
	<b>Possible Values:</b> .vcLEAFixedToColumns 0 .vcLEAFixedToRows 1 .vcLEAFixedToRowsAndColumns 2	The legend elements are merely aligned along columns. The legend elements are merely aligned along rows. The legend elements are aligned along rows and columns.

## LegendElementsBottomMargin

Property of VcBoundingBox

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
<b>Property value</b>	System.Int16	Width of bottom margin

## LegendElementsMaximumColumnCount

Property of VcBoundingBox

This property lets you set or retrieve the number of columns to which the elements in the legend should disperse.

	Data Type	Explanation
<b>Property value</b>	System.Int16	Number of columns

## LegendElementsMaximumRowCount

Property of VcBoundingBox

This property lets you set or retrieve the number of rows to which the elements in the legend should disperse.

	Data Type	Explanation
<b>Property value</b>	System.Int16	Number of rows

## 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	System.Int16	Width of top margin

## LegendFont

Property of VcBorderBox

This property lets you set or retrieve the font attributes of the legend.

	Data Type	Explanation
Property value	Font	Font attributes of the legend

### Example Code VB.NET

```
Dim borderArea As VcBorderArea
Dim borderBox As VcBorderBox

borderArea = VcGanttASP1.BorderArea
borderBox = borderArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomBottomLeft)
borderBox.Type = VcBorderBoxType.vcBBXTLegend
MsgBox(borderBox.LegendFont.Name)
```

### Example Code C#

```
VcBorderArea borderArea = vcGanttASP1.BorderArea;
VcBorderBox borderBox =
borderArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomBottomLeft);
borderBox.Type = VcBorderBoxType.vcBBXTLegend;
MessageBox.Show(borderBox.LegendFont.Name);
```

## LegendTitle

Property of VcBorderBox

This property lets you set or retrieve the legend title.

	Data Type	Explanation
Property value	System.String	Legend title

**Example Code VB.NET**

```
Dim borderArea As VcBorderArea
Dim borderBox As VcBorderBox

borderArea = VcGanttASP1.BorderArea
borderBox = borderArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomBottomLeft)
borderBox.LegendTitle = "Explanation"
```

**Example Code C#**

```
VcBorderArea borderArea = vcGanttASP1.BorderArea;

VcBorderBox borderBox =
borderArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomBottomLeft);
borderBox.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	Font	Font attributes of the legend title

**Example Code VB.NET**

```
Dim borderArea As VcBorderArea
Dim borderBox As VcBorderBox

borderArea = VcGanttASP1.BorderArea
borderBox = borderArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomBottomLeft)
borderBox.Type = VcBorderBoxType.vcBBXTLegend
MsgBox(borderBox.LegendTitleFont.Name)
```

**Example Code C#**

```
VcBorderArea borderArea = vcGanttASP1.BorderArea;
VcBorderBox borderBox =
borderArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomBottomLeft);
borderBox.Type = VcBorderBoxType.vcBBXTLegend;
MessageBox.Show(borderBox.LegendTitleFont.Name);
```

**LegendTitleVisible****Property of VcBorderBox**

This property lets you set or retrieve whether the legend title is visible.

	Data Type	Explanation
Property value	System.Boolean	Legend title visible (True)/ not visible (False)

### Example Code VB.NET

```
Dim borderArea As VcBorderArea
Dim borderBox As VcBorderBox

borderArea = VcGanttASP1.BorderArea
borderBox = borderArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomBottomLeft)
borderBox.LegendTitleVisible = False
```

### Example Code C#

```
VcBorderArea borderArea = vcGanttASP1.BorderArea;

VcBorderBox borderBox =
borderArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomBottomLeft);
borderBox.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

The property Text is an Indexed Property, which in C# is addressed by the methods set\_Text (rowIndex, pvn) and get\_Text (rowIndex).

	Data Type	Explanation
<b>Parameter:</b> rowIndex	System.Int16	Row index {0...6}
<b>Property value</b>	System.String	Text in text boxes

**Example Code VB.NET**

```
Dim borderArea As VcBorderArea
Dim borderBox As VcBorderBox

borderArea = VcGanttASP1.BorderArea
borderBox = borderArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomBottomLeft)
borderBox.Type = VcBorderBoxType.vcBBXTText
borderBox.Text(index) = "Department A"
```

**Example Code C#**

```
VcBorderArea borderArea = vcGanttASP1.BorderArea;

VcBorderBox borderBox =
borderArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomBottomLeft);
borderBox.Type = VcBorderBoxType.vcBBXTText;
borderBox.set_Text(index, "DepartmentA");
```

## 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)**.

The property TextFont is an Indexed Property, which in C# is addressed by the methods **set\_TextFont (rowIndex, pvn)** and **get\_TextFont (rowIndex)**.

	Data Type	Explanation
<b>Parameter:</b>		
rowIndex	System.Int16	Row index {0...6}
<b>Property value</b>	Font	Font attributes of the text

**Example Code VB.NET**

```
Dim borderArea As VcBorderArea
Dim bBoxTL As VcBorderBox

Set borderArea = VcGanttASP1.BorderArea
Set bBoxBBL = borderArea.BorderBox(vcBBXPBottomBottomLeft)

bBoxTL.TextFont(i).Bold = False
bBoxTL.TextFont(i).Italic = False
bBoxTL.TextFont(i).Name = "Symbol"
```

## 310 API Reference: VcBoundingBox

### Example Code C#

```
// Text for Title
VcBoundingBox borderBox =
VcGanttASP1.BorderArea.BorderBox(VcBoundingBoxPosition.vcBBXPTopCentered);
borderBox.Type = VcBoundingBoxType.vcBBXTText;

Font titleFont1 = new Font("Arial", 20, FontStyle.Bold);

borderBox.set_Text(1, "Time Scheduler");
borderBox.set_TextFont(1, titleFont1);
```

## Type

### Property of VcBoundingBox

This property lets you set or retrieve the type of the BorderBox object.

	Data Type	Explanation
Property value	VcBoundingBoxType  <b>Possible Values:</b> .vcBBXTGraphics 3 .vcBBXTLegend 4 .vcBBXTNothing 0 .vcBBXTText 1 .vcBBXTTextWithGraphics 2	Box type  graphics legend nothing text text and graphics

### Example Code VB.NET

```
Dim bBoxBBL As VcBoundingBox

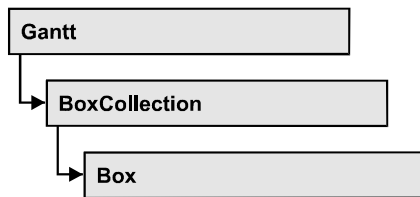
bBoxBBL = boardArea.BorderBox(VcBoundingBoxPosition.vcBBXPBottomLeft)
bBoxBBL.Type = VcBoundingBoxType.vcBBXTGraphics
```

### Example Code C#

```
VcBorderArea boardArea = vcGanttASP1.BorderArea;

VcBoundingBox bBoxBBL =
boardArea.BorderBox(VcBoundingBoxPosition.vcBBXPBottomBottomLeft);
bBoxBBL.Type = VcBoundingBoxType.vcBBXTGraphics;
```

## 6.4 VcBox



An object of the type **VcBox** designates a box to display texts or graphics.

### Properties

- FieldText
- FormatName
- LineColor
- LineThickness
- LineType
- Marked
- Name
- Origin
- Priority
- ReferencePoint
- Specification
- Visible

### Methods

- GetXYOffset
- SetXYOffset
- SetXYOffsetByTopLeftPixel

---

## Properties

### 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.



## 312 API Reference: VcBox

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.

The property `FieldText` is an Indexed Property, which in C# is addressed by the methods `set_FieldText (fieldIndex, pvn)` and `get_FieldText (fieldIndex)`.

	Data Type	Explanation
<b>Parameter:</b> ⇒ fieldIndex	System.Int16	Field index
<b>Property value</b>	System.String	Field content

### Example Code VB.NET

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcGanttASP1.BoxCollection
box = boxCltn.FirstBox
box.FieldText(0) = "User: "
```

### Example Code C#

```
VcBoxCollection boxCltn = vcGanttASP1.BoxCollection;
VcBox box = boxCltn.FirstBox();
box.set_FieldText(0, "User: ");
```

## FormatName

Property of VcBox

This property lets you set or retrieve the name of the box format.

	Data Type	Explanation
<b>Property value</b>	VcBoxFormat	BoxFormat object or <b>Nothing</b>

### Example Code VB.NET

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcGanttASP1.BoxCollection
box = boxCltn.FirstBox
box.FormatName = "Standard"
```

### Example Code C#

```
VcBoxCollection boxCltn = vcGanttASP1.BoxCollection;
VcBox 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	System.Drawing.Color	RGB color values {0...255},{0...255},{0...255}

### Example Code VB.NET

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcGanttASP1.BoxCollection
box = boxCltn.FirstBox
box.LineColor = System.Drawing.Color.Blue
```

### Example Code C#

```
VcBoxCollection boxCltn = vcGanttASP1.BoxCollection;
VcBox box = boxCltn.FirstBox();
box.LineColor = System.Drawing.Color.Blue;
```

## 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.



.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

**Example Code VB.NET**

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcGanttASP1.BoxCollection
box = boxCltn.FirstBox
box.LineType = VcLineType.vcDotted
```

**Example Code C#**

```
VcBoxCollection boxCltn = vcGanttASP1.BoxCollection;
VcBox box = boxCltn.FirstBox();
box.LineType = VcLineType.vcDotted;
```

**Marked****Property of VcBox**

This property lets you set or retrieve whether a text box is marked.

	Data Type	Explanation
<b>Property value</b>	System.Boolean	<b>True:</b> box marked; <b>false:</b> box unmarked

**Example Code VB.NET**

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcGanttASP1.BoxCollection
box = boxCltn.FirstBox
box.Marked = True
```

**Example Code C#**

```
VcBoxCollection boxCltn = vcGanttASP1.BoxCollection;
VcBox box = boxCltn.FirstBox();
box.Marked = true;
```

**Name****Property of VcBox**

This property lets you set or retrieve the name of a box. You can also specify the name in the **Administrative Boxes** dialog box.

	Data Type	Explanation
<b>Property value</b>	System.String	Box name

**Example Code VB.NET**

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox
Dim boxName As String

boxCltn = VcGanttASP1.BoxCollection
box = boxCltn.FirstBox
MsgBox(box.Name)
```

**Example Code C#**

```
VcBoxCollection boxCltn = vcGanttASP1.BoxCollection;
VcBox box = boxCltn.FirstBox();

MessageBox.Show(box.Name);
```

**Origin****Property of VcBox**

This property lets you set or retrieve the 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.

With the help of the properties **Origin**, **ReferencePoint** and the method **GetXYOffset** you can position each box in the diagram area. The relative position of the boxes is independent of the current diagram size.

	Data Type	Explanation
<b>Property value</b>	VcBoxOrigin  <b>Possible Values:</b> .vcBOBottomCenter 28 .vcBOBottomLeft 27 .vcBOBottomRight 29 .vcBOCenterCenter 25	Origin of the box  bottom center bottom left bottom right center center

.vcBOCenterLeft	24	center left
.vcBOCenterRight	26	center right
.vcBOTopCenter	22	top center
.vcBOTopLeft	21	top left
.vcBOTopRight	23	top right

**Example Code VB.NET**

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcGanttASP1.BoxCollection
box = boxCltn.FirstBox
box.Origin = VcBoxOrigin.vcBOTopCenter
```

**Example Code C#**

```
VcBoxCollection boxCltn = vcGanttASP1.BoxCollection;
VcBox box = boxCltn.FirstBox();
box.Origin = VcBoxOrigin.vcBOTopCenter;
```

## Priority

**Property of VcBox**

This property lets you set or retrieve the priority of the box.

	Data Type	Explanation
Property value	System.Int16	Priority value

**Example Code VB.NET**

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcGanttASP1.BoxCollection
box = boxCltn.FirstBox
box.Priority = 3
```

**Example Code C#**

```
VcBoxCollection boxCltn = vcGanttASP1.BoxCollection;
VcBox box = boxCltn.FirstBox();
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
<b>Property value</b>	VcBoxReferencePoint  <b>Possible Values:</b> .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 VB.NET**

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcGanttASP1.BoxCollection
box = boxCltn.FirstBox
box.ReferencePoint = VcBoxReferencePoint.vcBRPCenterRight
```

**Example Code C#**

```
VcBoxCollection boxCltn = vcGanttASP1.BoxCollection;
VcBox box = boxCltn.FirstBox();
box.ReferencePoint = VcBoxReferencePoint.vcBRPCenterRight;
```

**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
<b>Property value</b>	System.String	Specification of the box

**Example Code VB.NET**

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcGantt1.BoxCollection
box = boxCltn.FirstBox
MsgBox(box.Specification)
```

**Example Code C#**

```
VcBoxCollection boxCltn = vcGantt1.BoxCollection;
VcBox box = boxCltn.FirstBox();
MessageBox.Show(box.Specification);
```

## 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
<b>Property value</b>	System.Boolean	Box visible/invisible <b>Default value:</b> True

### Example Code VB.NET

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcGanttASP1.BoxCollection
box = boxCltn.FirstBox
box.Visible = False
```

### Example Code C#

```
VcBoxCollection boxCltn = vcGanttASP1.BoxCollection;
VcBox box = boxCltn.FirstBox();
box.Visible = false;
```

---

## Methods

### GetXYOffset

Method of VcBox

This method lets you retrieve the distance between origin and reference point in x and y direction (unit: 1/100 mm).

	Data Type	Explanation
<b>Parameter:</b>		
↔ xOffset	System.Int32	X value of the offset
↔ yOffset	System.Int32	Y value of the offset
<b>Return value</b>	System.Boolean	Offset is returned/not returned



## 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.

	Data Type	Explanation
<b>Parameter:</b>		
⇒ xOffset	System.Int32	X value of the offset
⇒ yOffset	System.Int32	Y value of the offset
<b>Return value</b>	System.Boolean	Offset is set (True)/not set (False)

### Example Code VB.NET

```
Dim offSet As Boolean
offSet = VcGanttASP1.BoxCollection.FirstBox.SetXYOffset(100, 100)
```

### Example Code C#

```
bool offSet = vcGanttASP1.BoxCollection.FirstBox().SetXYOffset(100, 100);
```

## SetXYOffsetByTopLeftPixel

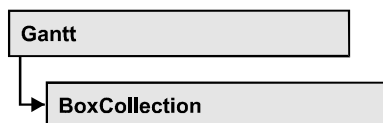
**Method of VcBox**

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
<b>Parameter:</b>		
⇒ x	System.Int32	X value of the offset
⇒ y	System.Int32	Y value of the offset
<b>Return value</b>	System.Boolean	Offset is set (True) / not set (False)

## 6.5 VcBoxCollection



The VcBoxCollection object contains all boxes available. You can access all objects in an iterative loop by **For Each box In VcBoxCollection** or by the methods **First...** and **Next...**. You can access a single box by the method **BoxByName**. 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

- Count

### Methods

- Add
- AddBySpecification
- BoxByIndex
- BoxByName
- Copy
- FirstBox
- GetEnumerator
- NextBox
- Remove
- Update

---

## Properties

### Count

**Read Only Property of VcBoxCollection**

This property lets you retrieve the number of boxes in the box collection.

	Data Type	Explanation
Property value	System.Int32	Number of boxes

### Example Code VB.NET

```
Dim boxCltn As VcBoxCollection
Dim numberOfBoxes As Integer

boxCltn = VcGanttASP1.BoxCollection
numberOfBoxes = boxCltn.Count
```

### Example Code C#

```
VcBoxCollection boxCltn = vcGanttASP1.BoxCollection;
int 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 has not been 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
<b>Parameter:</b> ⇒ boxName	System.String	Box name
<b>Return value</b>	VcBox	New box object

### Example Code VB.NET

```
newBox = VcGanttASP1.BoxCollection.Add("box1")
```

### Example Code C#

```
newBox = vcGanttASP1.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
<b>Parameter:</b>		
⇒ specification	System.String	Box specification
<b>Return value</b>	VcBox	New box object

#### Example Code VB.NET

```
Dim boxCltn As VcBoxCollection

boxCltn = VcGanttASP1.BoxCollection
boxCltn.AddBySpecification(textSpecification)
boxCltn.Update()
```

#### Example Code C#

```
VcBoxCollection boxCltn = vcGanttASP1.BoxCollection;
boxCltn.AddBySpecification(textSpecification);
boxCltn.Update();
```

## BoxByIndex

#### Method of VcBoxCollection

This method lets you access a box by its index. If a box does not exist at the index specified, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
<b>Parameter:</b>		
⇒ index	System.Int16	Index of the box
<b>Return value</b>	VcBox	Box object returned

#### Example Code VB.NET

```
Dim boxCltn As VcBoxCollection

boxCltn = VcGanttASP1.BoxCollection
box = boxCltn.BoxByIndex(0)
box.LineThickness = 2
```

#### Example Code C#

```
VcBoxCollection boxCltn = vcGanttASP1.BoxCollection;
VcBox box = boxCltn.BoxByIndex(0);
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
<b>Parameter:</b> ⇒ boxName	System.String	Box name
<b>Return value</b>	VcBox	Box

### Example Code VB.NET

```
Dim boxCltn As VcBoxCollection

boxCltn = VcGanttASP1.BoxCollection
box = boxCltn.BoxByName("BoxOne")
box.LineThickness = 3
```

### Example Code C#

```
VcBoxCollection boxCltn = vcGanttASP1.BoxCollection;
VcBox box = boxCltn.BoxByName("BoxOne");
box.LineThickness = 3;
```

## 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
<b>Parameter:</b> ⇒ boxName	System.String	Name of the box to be copied
⇒ newBoxName	System.String	Name of the new box
<b>Return value</b>	VcBox	Box object

**Example Code VB.NET**

```
Dim boxCltn As VcBoxCollection

boxCltn = VcGanttASP1.BoxCollection
boxCltn.Copy("BoxOne", "NewBox")
boxCltn.Update()
```

**Example Code C#**

```
VcBoxCollection boxCltn = vcGanttASP1.BoxCollection;
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 VB.NET**

```
Dim boxCltn As VcBoxCollection

boxCltn = VcGanttASP1.BoxCollection
box = boxCltn.FirstBox
```

**Example Code C#**

```
VcBoxCollection boxCltn = vcGanttASP1.BoxCollection;
VcBox box = boxCltn.FirstBox();
```

## GetEnumerator

**Method of VcBoxCollection**

This method returns an Enumerator object which supports the iteration by language specific elements. It is implied in the For...Each construct of Visual Basic and C#. This object allows to iterate over the box objects included.

	Data Type	Explanation
Return value	VcObject	Reference object

## 326 API Reference: VcBoxCollection

### Example Code VB.NET

```
Dim box As VcBox

For Each box In VcGanttASP1.BoxCollection
    ListBox1.Items.Add(box.FormatName)
Next
```

### Example Code C#

```
foreach (VcBox box in vcGanttASP1.BoxCollection)
    listBox1.Items.Add(box.FormatName);
```

## 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	Succeeding box

### Example Code VB.NET

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcGanttASP1.BoxCollection
box = boxCltn.FirstBox

While Not box Is Nothing
    ListBox1.Items.Add(box.Name)
    box = boxCltn.NextBox
End While
```

### Example Code C#

```
VcBoxCollection boxCltn = vcGanttASP1.BoxCollection;
VcBox box = boxCltn.FirstBox();

while (box != null)
{
    listBox.Items.Add(box.Name);
    box = boxCltn.NextBox();
}
```

## 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
<b>Parameter:</b> ⇒ <code>boxName</code>	System.String	Box name
<b>Return value</b>	System.Boolean	Box deleted (True)/not deleted (False)

**Example Code VB.NET**

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcGanttASP1.BoxCollection
box = boxCltn.BoxByIndex(0)
boxCltn.Remove(box.Name)
boxCltn.Update()
```

**Example Code C#**

```
VcBoxCollection boxCltn = vcGanttASP1.BoxCollection;
VcBox box = boxCltn.BoxByIndex(0);
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
<b>Return value</b>	System.Boolean	Update successful (True)/ not successful (False)

**Example Code VB.NET**

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

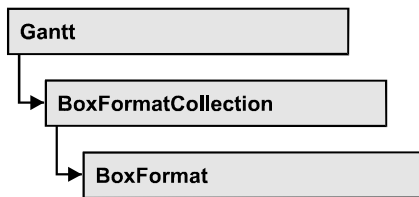
boxCltn = VcGanttASP1.BoxCollection
box = boxCltn.BoxByIndex(0)
boxCltn.Remove(box.Name)
boxCltn.Update()
```

**Example Code C#**

```
VcBoxCollection boxCltn = vcGanttASP1.BoxCollection;
VcBox box = boxCltn.BoxByIndex(0);
boxCltn.Remove(box.Name);
boxCltn.Update();
```



## 6.6 VcBoxFormat



An object of the type **VcBoxFormat** defines the formats of boxes. With **ForEach formatField In BoxFormat** you can retrieve all box formats

### Properties

- FieldsSeparatedByLines
- FormatField
- FormatFieldCount
- Name
- Specification

### Methods

- CopyFormatField
- GetEnumerator
- RemoveFormatField

---

## Properties

### FieldsSeparatedByLines

Property of VcBoxFormat

This property lets you set or retrieve whether fields are to be separated by lines.

	Data Type	Explanation
Property value	System.Boolean	Box fields separated by lines (True)/ not separated by lines (False).

#### Example Code VB.NET

```

Dim boxFormat As VcBoxFormat

boxFormat = VcGanttASP1.BoxFormatCollection.FormatByIndex(0)
boxFormat.FieldsSeparatedByLines = True
  
```

**Example Code C#**

```
VcBoxFormat boxFormat = vcGanttASP1.BoxFormatCollection.FormatByIndex(0);
boxFormat.FieldsSeparatedByLines = true;
```

**FormatField****Read Only Property of VcBoxFormat**

This property gives access to a VcBoxFormatField object by its index. The index has to be in the range from 0 to FormatFieldCount-1.

The property FormatField is an Indexed Property, which in C# is addressed by the method `get_FormatField(index)`.

	Data Type	Explanation
<b>Parameter:</b>		
index	System.Int16	Index of the box format field
<b>Property value</b>	VcBoxFormatField	Nox format field

**Example Code VB.NET**

```
Dim boxFormat As VcBoxFormat
Dim formatField As VcBoxFormatField

boxFormat = VcGanttASP1.BoxFormatCollection.FirstFormat
formatField = boxFormat.FormatField(0)
MsgBox(formatField.FormatName)
```

**Example Code C#**

```
VcBoxFormat boxFormat = vcGanttASP1.BoxFormatCollection.FirstFormat();
VcBoxFormatField formatField = boxFormat.get_FormatField(0);
MessageBox.Show(formatField.FormatName);
```

**FormatFieldCount****Read Only Property of VcBoxFormat**

This property allows to determine the number of fields in a box format.

	Data Type	Explanation
<b>Property value</b>	System.Int16	Number of fields of the box format

**Example Code VB.NET**

```
Dim boxFormat As VcBoxFormat
Dim formatField As VcBoxFormatField

boxFormat = VcGanttASP1.BoxFormatCollection.FirstFormat
MsgBox(boxFormat.FormatFieldCount)
```

### Example Code C#

```
VcBoxFormat boxFormat = vcGanttASP1.BoxFormatCollection.FirstFormat();  
MessageBox.Show(boxFormat.FormatFieldCount.ToString());
```

## 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	System.String	Box format name

### Example Code VB.NET

```
Dim boxFormat As VcBoxFormat  
  
For Each boxFormat In VcGanttASP1.BoxFormatCollection  
    ListBox1.Items.Add(boxFormat.Name)  
Next
```

### Example Code C#

```
foreach (VcBoxFormat boxFormat in vcGanttASP1.BoxFormatCollection)  
    listBox1.Items.Add(boxFormat.Name);
```

## 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	System.String	Specification of the box format

## 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
<b>Parameter:</b> ⇒ position	VcFormatFieldInnerPosition	Position of the new box format field
	<b>Possible Values:</b> .vcInnerAbove 1 .vcInnerBelow 3 .vcInnerLeftOf 0 .vcInnerRightOf 4	above below left of right of
⇒ refIndex	System.Int16	Index of the reference box format field
<b>Return value</b>	VcBoxFormatField	Box format field object

#### Example Code VB.NET

```
Dim boxFormat As VcBoxFormat
Dim formatField As VcBoxFormatField

boxFormat = VcGanttASP1.BoxFormatCollection.FormatByIndex(2)
boxFormat.CopyFormatField(VcFormatFieldInnerPosition.vcInnerRightOf, 0)
```

#### Example Code C#

```
VcBoxFormat boxFormat = vcGanttASP1.BoxFormatCollection.FormatByIndex(0);
VcBoxFormatField formatField =
boxFormat.CopyFormatField(VcFormatFieldInnerPosition.vcInnerRightOf, 0);
```

### GetEnumerator

Method of VcBoxFormat

This method returns an Enumerator object which supports the iteration by language specific elements. It is implied in the For...Each construct of Visual Basic and C#. This object allows to iterate over the box format fields included.

	Data Type	Explanation
<b>Return value</b>	VcObject	Reference object

### Example Code VB.NET

```
Dim boxFormat As VcBoxFormat
Dim formatField As VcBoxFormatField

boxFormat = VcGanttASP1.BoxFormatCollection.FirstFormat
For Each formatField In boxFormat
    ListBox1.Items.Add(formatField.FormatName)
Next
```

### Example Code C#

```
VcBoxFormat boxFormat = vcGanttASP1.BoxFormatCollection.FirstFormat();
foreach(VcBoxFormatField formatField in boxFormat)
    listBox1.Items.Add(formatField.FormatName);
```

## RemoveFormatField

Method of VcBoxFormat

This method lets you remove a box format field by its index. After that, the program will set all box format field indexes newly in order to number them consecutively.

	Data Type	Explanation
<b>Parameter:</b> ⇒ index	System.Int16	Index of the box format field to be deleted

### Example Code VB.NET

```
Dim boxFormat As VcBoxFormat
Dim i As Integer

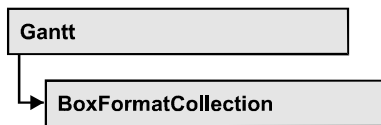
boxFormat = VcGanttASP1.BoxFormatCollection.FirstFormat

For i = 0 To boxFormat.FormatFieldCount - 1
    boxFormat.RemoveFormatField(i)
Next
```

### Example Code C#

```
VcBoxFormat boxFormat = vcGanttASP1.BoxFormatCollection.FirstFormat();
for (short i=0; i<boxFormat.FormatFieldCount-1; i++)
    boxFormat.RemoveFormatField(i);
```

## 6.7 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 method **BoxFormatByName**. 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 box formats in the corresponding way.

### Properties

- Count

### Methods

- Add
- AddBySpecification
- Copy
- FirstFormat
- FormatByIndex
- FormatByName
- GetEnumerator
- NextFormat
- Remove

---

## Properties

### 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	System.Int32	Number of box formats

### Example Code VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim numberOfBoxformats As Integer

boxFormatCltn = VcGanttASP1.BoxFormatCollection
numberOfBoxformats = boxFormatCltn.Count
```

### Example Code C#

```
VcBoxFormatCollection boxFormatCltn = vcGanttASP1.BoxFormatCollection;
int 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 has not been used before, the new box object will be returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
<b>Parameter:</b> ⇒ formatName	System.String	Box format name
<b>Return value</b>	VcBoxFormat	New box format object

### Example Code VB.NET

```
Dim newBoxFormat = VcGanttASP1.BoxFormatCollection.Add("boxFormat1")
```

### Example Code C#

```
newBoxFormat = vcGanttASP1.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
<b>Parameter:</b> ⇒ formatSpecification	System.String	Box format specification
<b>Return value</b>	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
<b>Parameter:</b> ⇒ FormatName	System.String	Name of the box format to be copied
⇒ newFormatName	System.String	Name of the new box format
<b>Return value</b>	VcBoxFormat	Box format object

### Example Code VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormat As VcBoxFormat

boxFormatCltn = VcGanttASp1.BoxFormatCollection
boxFormat = boxFormatCltn.Copy("CurrentBoxFormat", "NewBoxFormat")
```

### Example Code C#

```
VcBoxFormatCollection boxFormatCltn = vcGanttASp1.BoxFormatCollection;
VcBoxFormat 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).



## 336 API Reference: VcBoxFormatCollection

	Data Type	Explanation
Return value	VcBoxFormat	First box format

### Example Code VB.NET

```
Dim format As VcBoxFormat  
  
format = VcGanttASp1.BoxFormatCollection.FirstFormat
```

### Example Code C#

```
VcBoxFormat format = vcGanttASp1.BoxFormatCollection.FirstFormat();
```

## FormatByIndex

### Method of VcBoxFormatCollection

This method lets you access a box format by its index. If a box format does not exist at the index specified, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
<b>Parameter:</b> ⇒ index	System.Int16	Index of the box format
<b>Return value</b>	VcBoxFormat	Box format object returned

### Example Code VB.NET

```
Dim formatBoxCltn As VcBoxFormatCollection  
Dim formatBox As VcBoxFormat  
  
formatBoxCltn = VcGanttASp1.BoxFormatCollection  
formatBox = formatBoxCltn.FormatByIndex(2)
```

### Example Code C#

```
VcBoxFormatCollection boxFormatCltn = vcGanttASp1.BoxFormatCollection;  
VcBoxFormat 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
<b>Parameter:</b> ⇒ formatName	System.String	Name of the box format
<b>Return value</b>	VcBoxFormat	Box format

**Example Code VB.NET**

```
Dim formatBoxCltn As VcBoxFormatCollection
Dim formatBox As VcBoxFormat

formatBoxCltn = VcGanttASp1.BoxFormatCollection
formatBox = formatBoxCltn.FormatByName("Standard")
```

**Example Code C#**

```
VcBoxFormatCollection boxFormatCltn = vcGanttASp1.BoxFormatCollection;
VcBoxFormat format = boxFormatCltn.FormatByName("Standard");
```

## GetEnumerator

**Method of VcBoxFormatCollection**

This method returns an Enumerator object which supports the iteration by language specific elements. It is implied in the For...Each construct of Visual Basic and C#. This object allows to iterate over the box format objects included.

	Data Type	Explanation
<b>Return value</b>	VcObject	Reference object

**Example Code VB.NET**

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormat As VcBoxFormat

boxFormatCltn = VcGanttASp1.BoxFormatCollection
For Each boxFormat In boxFormatCltn
    ListBox1.Items.Add(boxFormat.Name)
Next
```

**Example Code C#**

```
VcBoxFormatCollection boxFormatCltn = vcGanttASp1.BoxFormatCollection;
foreach (VcBoxFormat boxFormat in boxFormatCltn)
    listBox1.Items.Add(boxFormat.Name);
```

## 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
<b>Return value</b>	VcBoxFormat	Subsequent box format

### Example Code VB.NET

```
Dim formatBoxCltn As VcBoxFormatCollection
Dim formatBox As VcBoxFormat

formatBoxCltn = VcGanttASP1.BoxFormatCollection
formatBox = formatBoxCltn.FirstFormat

While Not formatBox Is Nothing
    ListBox1.Items.Add(formatBox.Name)
    formatBox = formatBoxCltn.NextFormat
End While
```

### Example Code C#

```
VcBoxFormatCollection boxFormatCltn = vcGanttASP1.BoxFormatCollection;
VcBoxFormat boxFormat = boxFormatCltn.FirstFormat();

while (boxFormat != null)
{
    ListBox.Items.Add(boxFormat.Name);
    boxFormat = boxFormatCltn.NextFormat();
}
```

## 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
<b>Parameter:</b> ⇒ FormatName	System.String	Box format name
<b>Return value</b>	System.Boolean	Box format deleted (True)/not deleted (False)

### Example Code VB.NET

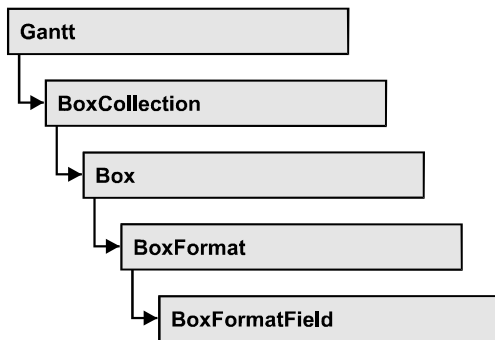
```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormat As VcBoxFormat

boxFormatCltn = VcGanttASP1.BoxFormatCollection
boxFormat = boxFormatCltn.FormatByIndex(1)
boxFormatCltn.Remove(boxFormat.Name)
```

**Example Code C#**

```
VcBoxFormatCollection boxFormatCltn = vcGanttASP1.BoxFormatCollection;  
VcBoxFormat boxFormat = boxFormatCltn.FormatByIndex(1);  
boxFormatCltn.Remove(boxFormat.Name);
```

## 6.8 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
- BackgroundColor
- FormatName
- GraphicsHeight
- Index
- MaximumTextLineCount
- MinimumTextLineCount
- MinimumWidth
- Pattern
- 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
<b>Property value</b>	VcFormatFieldAlignment	Alignment of the field content
	<b>Possible Values:</b> .vcFFABottom 28 .vcFFABottomLeft 27 .vcFFABottomRight 29 .vcFFACenter 25 .vcFFALeft 24 .vcFFARight 26 .vcFFATop 22 .vcFFATopLeft 21 .vcFFATopRight 23	Bottom Bottom left Bottom right Center Left Right Top Top left Top right

### Example Code VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

boxFormatCltn = VcGanttASP1.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.Alignment = VcFormatFieldAlignment.vcFFACenter
```

### Example Code C#

```
VcBoxFormatCollection boxFormatCltn = vcGanttASP1.BoxFormatCollection;
VcBoxFormatField boxFormatField =
boxFormatCltn.FirstFormat().get_FormatField(0);
boxFormatField.Alignment = VcFormatFieldAlignment.vcFFACenter;
```

## BackgroundColor

### 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.

If the box format field shall have the background color of the box format, select the value **-1**.

	Data Type	Explanation
<b>Property value</b>	System.Drawing.Color	RGB color values {{0...255},{0...255},{0...255}} <b>Default value:</b> -1

## 342 API Reference: VcBoxFormatField

### Example Code VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

boxFormatCltn = VcGanttASPl.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.BackgroundColor = Color.Red
```

### Example Code C#

```
VcBoxFormatCollection boxFormatCltn = vcGanttASPl.BoxFormatCollection;
VcBoxFormatField boxFormatField =
boxFormatCltn.FirstFormat().get_FormatField(0);
boxFormatField.BackgroundColor = Color.Red;
```

## FormatName

Read Only Property of VcBoxFormatField

This property lets you retrieve the name of the box format to which this field belongs.

	Data Type	Explanation
Property value	System.String	Name of the box format

### Example Code VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

boxFormatCltn = VcGanttASPl.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
MsgBox(boxFormatField.FormatName)
```

### Example Code C#

```
VcBoxFormatCollection boxFormatCltn = vcGanttASPl.BoxFormatCollection;
VcBoxFormatField boxFormatField =
boxFormatCltn.FirstFormat().get_FormatField(0);
MessageBox.Show(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	System.Int16	Height (in mm) of the graphics 0...200

**Example Code VB.NET**

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

boxFormatCltn = VcGanttASP1.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.Type = VcFormatFieldType.vcFFTGraphics
boxFormatField.GraphicsHeight = 150
```

**Example Code C#**

```
VcBoxFormatCollection boxFormatCltn = vcGanttASP1.BoxFormatCollection;
VcBoxFormatField boxFormatField =
boxFormatCltn.FirstFormat().get_FormatField(0);
boxFormatField.Type = VcFormatFieldType.vcFFTGraphics;
boxFormatField.GraphicsHeight = 150;
```

## Index

**Read Only Property of VcBoxFormatField**

This property lets you retrieve the index of the box format field in the associated box format.

	Data Type	Explanation
Property value	System.Int16	Index of the box format field

**Example Code VB.NET**

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

boxFormatCltn = VcGanttASP1.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
MsgBox(boxFormatField.Index)
```

**Example Code C#**

```
VcBoxFormatCollection boxFormatCltn = vcGanttASP1.BoxFormatCollection;
VcBoxFormatField boxFormatField =
boxFormatCltn.FirstFormat().get_FormatField(0);
MessageBox.Show(boxFormatField.Index.ToString());
```

## 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	System.Int16	Maximum number of lines



### Example Code VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

boxFormatCltn = VcGanttASP1.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.Type = VcFormatFieldType.vcFFTTText
boxFormatField.MaximumTextLineCount = 5
```

### Example Code C#

```
VcBoxFormatCollection boxFormatCltn = vcGanttASP1.BoxFormatCollection;
VcBoxFormatField boxFormatField =
boxFormatCltn.FirstFormat().get_FormatField(0);
boxFormatField.Type = VcFormatFieldType.vcFFTTText;
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 **vcFFTTText**. 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	System.Int16	Minimum number of lines 0...20

### Example Code VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

boxFormatCltn = VcGanttASP1.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.Type = VcFormatFieldType.vcFFTTText
boxFormatField.MinimumTextLineCount = 3
```

### Example Code C#

```
VcBoxFormatCollection boxFormatCltn = vcGanttASP1.BoxFormatCollection;
VcBoxFormatField boxFormatField =
boxFormatCltn.FirstFormat().get_FormatField(0);
boxFormatField.Type = VcFormatFieldType.vcFFTTText;
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	System.Int16	Minimum width of the box format field 0...200

### Example Code VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

boxFormatCltn = VcGanttASp1.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.MinimumWidth = 100
```




### Example Code C#



```
VcBoxFormatCollection boxFormatCltn = vcGanttASp1.BoxFormatCollection;
VcBoxFormatField boxFormatField =
boxFormatCltn.FirstFormat().get_FormatField(0);
boxFormatField.MinimumWidth = 100;
```

## Pattern

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	VcFieldFillPattern	Pattern type
	<b>Possible Values:</b> .vcAeroGlassPattern 44  .vcFieldNoPattern 1276 .vcFieldVerticalBottomLightedConvexPattern 43  .vcFieldVerticalConcavePattern 40	Vertical color gradient in the color of the fill pattern  No fill pattern Vertical color gradient from bright to dark  Vertical color gradient from dark to bright to dark 

.vcFieldVerticalConvexPattern 41  .vcFieldVerticalTopLightedConvexPattern 42	Vertical color gradient from bright to dark to bright  Vertical color gradient from dark to bright 
--	--

### Example Code VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

boxFormatCltn = VcGanttASP1.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.Pattern = VcFillPatternSingleColored.vcSingleColoredNoPatter
```

### Example Code C#

```
VcBoxFormatCollection boxFormatCltn = vcGanttASP1.BoxFormatCollection;
VcBoxFormatField boxFormatField =
boxFormatCltn.FirstFormat().get_FormatField(0);
boxFormatField.Pattern = VcFillPatternSingleColored.vcSingleColoredNoPattern;
```

## 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	Font	Font type of the box format

### Example Code VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

boxFormatCltn = VcGanttASP1.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
MsgBox(boxFormatField.TextFont.FontFamily.ToString())
```

### Example Code C#

```
VcBoxFormatCollection boxFormatCltn = vcGanttASP1.BoxFormatCollection;
VcBoxFormatField boxFormatField =
boxFormatCltn.FirstFormat().get_FormatField(0);
MessageBox.Show(boxFormatField.TextFont.Name.ToString());
```

## 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
<b>Property value</b>	System.Drawing.Color	Font color of the box format <b>Default value:</b> Color.Black

### Example Code VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

boxFormatCltn = VcGanttASp1.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.TextFontColor = Color.Red
```

### Example Code C#

```
VcBoxFormatCollection boxFormatCltn = vcGanttASp1.BoxFormatCollection;
VcBoxFormatField boxFormatField =
boxFormatCltn.FirstFormat().get_FormatField(0);
boxFormatField.TextFontColor = Color.Red;
```

## Type

Property of VcBoxFormatField

This property lets you enquire the type of the box format field.

	Data Type	Explanation
<b>Property value</b>	VcFormatFieldType  <b>Possible Values:</b> .vcFFTGraphics 64 .vcFFTText 36	Type of the box format field  Graphics Text

### Example Code VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

boxFormatCltn = VcGanttASp1.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.Type = VcFormatFieldType.vcFFTGraphics
boxFormatField.GraphicsHeight = 150
```

## 348 API Reference: VcBoxFormatField

### Example Code C#

```
VcBoxFormatCollection boxFormatCltn = vcGanttASP1.BoxFormatCollection;  
VcBoxFormatField boxFormatField =  
boxFormatCltn.FirstFormat().get_FormatField(0);  
boxFormatField.Type = VcFormatFieldType.vcFFTGraphics;  
boxFormatField.GraphicsHeight = 150;
```

## 6.9 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. Bars and layers adapt to the time pattern provided by the calendar.

A calendar also is useful for scheduling, e.g. to count the work days between two set dates.

You also can use a calendar to interrupt nodes by workfree intervals.

Furthermore, calendars specify calendar grids.

### Properties

- CalendarProfileCollection
- IntervalCollection
- Name
- SecondsPerWorkday
- Specification
- Type

### Methods

- AddDuration
- CalcDuration
- GetEndOfPreviousWorktime
- GetNextIntervalBorder
- GetPreviousIntervalBorder
- GetStartOfInterval
- GetStartOfNextWorktime
- IsWorktime
- Update

## Properties

### CalendarProfileCollection

Read Only Property of VcCalendar

This property gives access to the CalendarProfileCollection 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	System.String	Name of the calendar

#### Example Code VB.NET

```
Dim calendar As VcCalendar
Dim calendarName As String

calendar = VcGanttASP1.CalendarCollection.FirstCalendar
calendarName = calendar.Name
```

#### Example Code C#

```
VcCalendar calendar = vcGanttASP1.CalendarCollection.FirstCalendar();
string calendarName = calendar.Name;
```

## SecondsPerWorkday

Read Only 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	System.Int32	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 **VcCalendarCollection.AddBySpecification**.

	Data Type	Explanation
Property value	System.String	Specification of the calendar

## 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	VcCalendarType  <b>Possible Values:</b> .vcNormalCalendar 139 .vcShiftCalendar 12	Calendar type

### Example Code VB.NET

```
Dim calendarCltn As VcCalendarCollection
Dim calendar As VcCalendar

calendarCltn = VcGantt1.CalendarCollection
calendar = calendarCltn.CalendarByIndex(0)
calendar.Type = VcCalendarType.vcNormalCalendar
```



### Example Code C#

```
VcCalendarCollection calendarCltn = vcGantt1.CalendarCollection;  
VcCalendar calendar = calendarCltn.CalendarByIndex(0);  
calendar.Type = VcCalendarType.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
<b>Parameter:</b>		
⇒ date	System.DateTime	Date the duration is to be inserted at
⇒ duration	System.Int32	Number of time units (e.g.days)
<b>Return value</b>	System.DateTime	Date the duration was inserted at

### Example Code VB.NET

```
Dim calendar As VcCalendar  
Dim newDate As Date  
  
calendar = VcGanttASP1.CalendarCollection.CalendarByName("WeekCalendar")  
newDate = calendar.AddDuration("16.06.2012", 3)
```

### Example Code C#

```
VcCalendar calendar =  
vcGanttASP1.CalendarCollection.CalendarByName("WeekCalendar");  
DateTime newDate = calendar.AddDuration(Convert.ToDateTime("16.06.2012"), 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
<b>Parameter:</b>		
⇒ fromDate	System.DateTime	Start date of the duration that the number of work time elements is to be retrieved of
⇒ toDate	System.DateTime	End date of the duration that the number of work time elements is to be retrieved of
<b>Return value</b>	System.Int32	Number of time units (e.g. days) of the duration

**Example Code VB.NET**

```
Dim calendar As VcCalendar
Dim duration As Integer

calendar = VcGanttASP1.CalendarCollection.CalendarByName("WeekCalendar")
duration = calendar.CalcDuration("01.01.2014", "31.12.2014")
```

**Example Code C#**

```
VcCalendar calendar =
vcGanttASP1.CalendarCollection.CalendarByName("WeekCalendar");
int duration = calendar.CalcDuration(Convert.ToDateTime("01.01.2014"),
Convert.ToDateTime("31.12.2014"));
```

## 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
<b>Parameter:</b>		
⇒ date	System.DateTime	Date that the previous work time refers to
<b>Return value</b>	System.DateTime	Final date of the previous work time

**Example Code VB.NET**

```
Dim calendar As VcCalendar
Dim endOfWork As Date

calendar = VcGanttASP1.CalendarCollection.CalendarByName("WeekCalendar")
endOfWork = calendar.GetEndOfPreviousWorktime("18.06.2014")
```

**Example Code C#**

```
VcCalendar calendar =
vcGanttASP1.CalendarCollection.CalendarByName("WeekCalendar");
DateTime endOfWork =
calendar.GetEndOfPreviousWorktime(Convert.ToDateTime("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
<b>Parameter:</b>		
⇒ date	System.DateTime	Date that the subsequent interval border refers to
<b>Return value</b>	System.DateTime	Start date of the subsequent interval border

### Example Code VB.NET

```
Dim calendar As VcCalendar
Dim nextIntervalBorder As Date

calendar = VcGanttASP1.CalendarCollection.CalendarByName("WeekCalendar")
nextIntervalBorder = calendar.GetNextIntervalBorder("18.06.2014")
```

### Example Code C#

```
VcCalendar calendar =
vcGanttASP1.CalendarCollection.CalendarByName("WeekCalendar");
DateTime nextIntervalBorder =
calendar.GetNextIntervalBorder(Convert.ToDateTime("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
<b>Parameter:</b>		
⇒ date	System.DateTime	Date that of the preceding interval border refers to
<b>Return value</b>	System.DateTime	End date of the interval border preceding

### Example Code VB.NET

```
Dim calendar As VcCalendar
Dim previousIntervalBorder As Date

calendar = VcGanttASP1.CalendarCollection.CalendarByName("WeekCalendar")
previousIntervalBorder = calendar.GetPreviousIntervalBorder("18.06.2014")
```

**Example Code C#**

```
VcCalendar calendar =
vcGanttASP1.CalendarCollection.CalendarByName("WeekCalendar");
DateTime previousIntervalBorder =
calendar.GetPreviousIntervalBorder(Convert.ToDateTime("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
<b>Parameter:</b> ⇒ date	System.DateTime	Reference date of the interval, that the start date is to be retrieved of
<b>Return value</b>	System.DateTime	Start date of the interval

**Example Code VB.NET**

```
Dim calendar As VcCalendar
Dim startOfInterval As Date

calendar = VcGanttASP1.CalendarCollection.CalendarByName("WeekCalendar")
startOfInterval = calendar.GetStartOfInterval("18.06.2014")
```

**Example Code C#**

```
VcCalendar calendar =
vcGanttASP1.CalendarCollection.CalendarByName("WeekCalendar");
DateTime startOfInterval =
calendar.GetStartOfInterval(Convert.ToDateTime("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
<b>Parameter:</b> ⇒ date	System.DateTime	Reference date, of which the start date of the subsequent work time is to be retrieved
<b>Return value</b>	System.DateTime	Start date of the subsequent work time

### Example Code VB.NET

```
Dim calendar As VcCalendar
Dim startOfNextWorktime As Date

calendar = VcGanttASP1.CalendarCollection.CalendarByName("WeekCalendar")
startOfNextWorktime = calendar.GetStartOfNextWorktime("18.06.2012")
```

### Example Code C#

```
VcCalendar calendar =
vcGanttASP1.CalendarCollection.CalendarByName("WeekCalendar");
DateTime startOfNextWorktime =
calendar.GetStartOfNextWorktime(Convert.ToDateTime("18.06.2012"));
```

## IsWorktime

### Method of VcCalendar

This method lets you retrieve whether or not the date passed is in a work time.

	Data Type	Explanation
<b>Parameter:</b> ⇒ date	System.DateTime	Date to be checked for being a work time
<b>Return value</b>	System.Boolean	Date passed does /does not belong to a work time

### Example Code VB.NET

```
Dim calendar As VcCalendar
Dim isWorktime As Boolean

calendar = VcGanttASP1.CalendarCollection.CalendarByName("WeekCalendar")
isWorktime = calendar.IsWorktime ("18.06.2014")
```

### Example Code C#

```
VcCalendar calendar =
vcGanttASP1.CalendarCollection.CalendarByName("WeekCalendar");
bool isWorktime = calendar.IsWorktime(Convert.ToDateTime("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
<b>Return value</b>	Void	

**Example Code VB.NET**

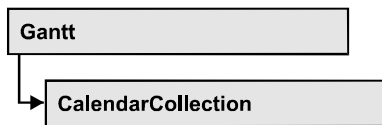
```
Dim calendar As VcCalendar

calendar = VcGanttASP1.CalendarCollection.CalendarByName("WeekCalendar")
calendar.Update()
```

**Example Code C#**

```
VcCalendar calendar =
vcGanttASP1.CalendarCollection.CalendarByName("WeekCalendar");
calendar.Update();
```

## 6.10 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 method **CalendarByName**. 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

- Active
- Count

### Methods

- Add
- AddBySpecification
- CalendarByIndex
- CalendarByName
- Copy
- FirstCalendar
- GetEnumerator
- NextCalendar
- Remove
- Update

---

## Properties

### Active

**Property of VcCalendarCollection**

This property lets you retrieve or set the default calendar for nodes, if no other calendar was assigned.

	Data Type	Explanation
Property value	VcCalendar	Currently used calendar

### Example Code VB.NET

```
Dim workday As VcWorkday
Dim freeday As VcWorkday
Dim workweek As VcWorkweek
Dim calendarCltn As VcCalendarCollection
Dim calendar As VcCalendar

workday = VcGantt1.WorkdayCollection.CreateWorkday("Work day")
workday.AddNonWorkInterval("00:00:00", "00:00:00")
workday.AddWorkInterval("08:00:00", "16:30:00")
freeday = VcGantt1.WorkdayCollection.CreateWorkday("Workfree day")
freeday.AddNonWorkInterval("00:00:00", "00:00:00")
calendarCltn = VcGantt1.CalendarCollection
calendar = calendarCltn.CreateCalendar("New calendar")
workweek = VcGantt1.WorkweekCollection.CreateWorkweek("Work week")
workweek.AddWorkday(workday, VcWeekday.vcMonday, VcWeekday.vcFriday)
workweek.AddWorkday(freeday, VcWeekday.vcSaturday, VcWeekday.vcSunday)
calendar.AddWorkweek(workweek, "01.01.13", "31.12.14")
calendar.Update()
calendarCltn.Active = calendar
```

### Example Code C#

```
VcWorkday workday = VcGantt1.WorkdayCollection.CreateWorkday("Work day");
workday.AddNonWorkInterval(Convert.ToDateTime("00:00:00"),
Convert.ToDateTime("00:00:00"));
workday.AddWorkInterval(Convert.ToDateTime("08:00:00"),
Convert.ToDateTime("16:30:00"));
VcWorkday freeday = VcGantt1.WorkdayCollection.CreateWorkday("Workfree day");
freeday.AddNonWorkInterval(Convert.ToDateTime("00:00:00"),
Convert.ToDateTime("00:00:00"));
VcCalendarCollection calendarCltn = VcGantt1.CalendarCollection;
VcCalendar calendar = calendarCltn.CreateCalendar("New calendar");
VcWorkweek workweek = VcGantt1.WorkweekCollection.CreateWorkweek("Work week");
workweek.AddWorkday(workday, VcWeekday.vcMonday, VcWeekday.vcFriday);
workweek.AddWorkday(freeday, VcWeekday.vcSaturday, VcWeekday.vcSunday);
calendar.AddWorkweek(workweek, Convert.ToDateTime("01.01.13"),
Convert.ToDateTime("31.12.14"));
calendar.Update();
calendarCltn.Active = calendar;
```

## Count

### Read Only Property of VcCalendarCollection

This property lets you retrieve the number of calendars in the CalendarCollection object.

	Data Type	Explanation
Property value	System.Int32	Number of calendars



### Example Code VB.NET

```
Dim calendarCltn As VcCalendarCollection
Dim numberOfCalendar As Integer

calendarCltn = VcGanttASP1.CalendarCollection
numberOfCalendar = calendarCltn.Count
```

### Example Code C#

```
VcCalendarCollection calendarCltn = vcGanttASP1.CalendarCollection;
int numberOfCalendar = calendarCltn.Count;
```

---

## 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
<b>Parameter:</b>		
⇒ calendarName	System.String	Calendar name
<b>Return value</b>	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
<b>Parameter:</b>		
⇒ Specification	System.String	Calendar specification
<b>Return value</b>	VcCalendar	New calendar object

## CalendarByIndex

Method of VcCalendarCollection

This method lets you access a calendar by its index. If a calendar does not exist at the index specified, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
<b>Parameter:</b> ⇒ index	System.Int16	Index of the calendar
<b>Return value</b>	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
<b>Parameter:</b> ⇒ calendarName	System.String	Name of the calendar
<b>Return value</b>	VcCalendar	Calendar

### Example Code VB.NET

```
Dim calendarCltn As VcCalendarCollection

calendarCltn = VcGanttASP1.CalendarCollection
calendarCltn.Active = calendarCltn.CalendarByName("Calendar_1")
```

### Example Code C#

```
VcCalendarCollection calendarCltn = vcGanttASP1.CalendarCollection;
calendarCltn.Active = calendarCltn.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.

## 362 API Reference: VcCalendarCollection

	Data Type	Explanation
<b>Parameter:</b>		
⇒ calendarName	System.String	Name of the calendar to be copied
⇒ newCalendarName	System.String	Name of the calendar
<b>Return value</b>	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, to continue in a forward iteration loop by the method **NextCalendar** for the calendars following. If there is no calendar in the calendar collection, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
<b>Return value</b>	VcCalendar	First calendar

### Example Code VB.NET

```
Dim calendarCltn As VcCalendarCollection
Dim calendar As VcCalendar

calendarCltn = VcGanttASP1.CalendarCollection
calendar = calendarCltn.FirstCalendar
```

### Example Code C#

```
VcCalendarCollection calendarCltn = vcGanttASP1.CalendarCollection;
VcCalendar calendar = calendarCltn.FirstCalendar();
```

## GetEnumerator

### Method of VcCalendarCollection

This method returns an Enumerator object which supports the iteration by language specific elements. It is implied in the For...Each construct of Visual Basic and C#. This object allows to iterate over the calendar objects included.

	Data Type	Explanation
<b>Return value</b>	VcObject	Reference object

**Example Code VB.NET**

```
Dim calendar As VcCalendar

For Each calendar In VcGanttASP1.CalendarCollection
    MsgBox(calendar.Name)
Next
```

**Example Code C#**

```
foreach (VcCalendar calendar in vcGanttASP1.CalendarCollection)
    MessageBox.Show(calendar.Name);
```

## 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
<b>Return value</b>	VcCalendar	Succeeding calendar

**Example Code VB.NET**

```
Dim calendarCltn As VcCalendarCollection
Dim calendar As VcCalendar
calendarCltn = VcGanttASP1.CalendarCollection
calendar = calendarCltn.FirstCalendar

While Not calendar Is Nothing
    ListBox1.Items.Add(calendar.Name)
    calendar = calendarCltn.NextCalendar
End While
```

**Example Code C#**

```
VcCalendarCollection calendarCltn = vcGanttASP1.CalendarCollection;
VcCalendar calendar = calendarCltn.FirstCalendar();

while (calendar != null)
{
    ListBox.Items.Add(calendar.Name);
    calendar = calendarCltn.NextCalendar();
}
```

## 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.

## 364 API Reference: VcCalendarCollection

	Data Type	Explanation
Return value	System.Boolean	Calendar deleted (True)/not deleted (False)

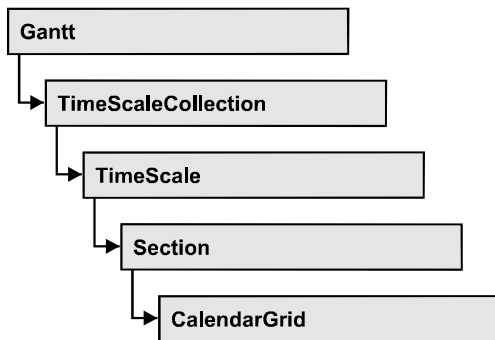
## Update

### Method of VcCalendarCollection

This method lets you update a calendar collection after having modified it.

	Data Type	Explanation
Return value	System.Boolean	update successful (True)/ not successful (False)

## 6.11 VcCalendarGrid



An object of the type **VcCalendarGrid** is a grid of vertical lines to highlight workfree periods by colored vertical areas.

### Properties

- BackgroundColor
- BackgroundColorDataFieldIndex
- BackgroundColorMapName
- CalendarName
- LineColor
- LineColorDataFieldIndex
- LineColorMapName
- LineType
- Name
- Pattern
- PatternColor
- PatternColorDataFieldIndex
- PatternColorMapName
- PatternDataFieldIndex
- PatternMapName
- Priority
- UseGraphicalAttributesOfIntervals
- Visible

## Properties

### BackgroundColor

Property of VcCalendarGrid

This property lets you specify or retrieve the color of the vertical areas 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.

Also see [set/getPatternColor](#) and [set/getPattern](#).

	Data Type	Explanation
Property value	System.Drawing.Color	RGB color values <b>Default value:</b> 14 211 288. Visual Basic: RGB (216, 216, 216)

#### Example Code VB.NET

```
Dim section As VcSection
Dim calendarGrid As VcCalendarGrid

section = VcGantt1.TimeScaleCollection.Active.Section(0)
calendarGrid = section.CalendarGrid(0)
calendarGrid.Color = Color.Blue
```

#### Example Code C#

```
VcSection section = vcGantt1.TimeScaleCollection.Active.get_Section(0);
VcCalendarGrid calendarGrid = section.get_CalendarGrid(0);
calendarGrid.Color = Color.LightSteelBlue;
```

### BackgroundColorDataFieldIndex

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	System.Int16	Data field index

## BackgroundColorMapName

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	System.String	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	System.String	Character string that passes the calendar name

## LineColor

Read Only Property of VcCalendarGrid

This property lets you specify/enquire 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	System.Drawing.Color	RGB color values ({0...255},{0...255},{0...255})





.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

## Name

### Read Only Property of VcCalendarGrid


This property lets you specify/enquire the name of a calendar grid.








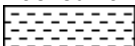


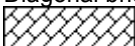
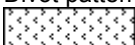

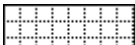


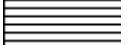
	Data Type	Explanation
Property value	System.String	Name of the calendar grid




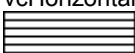







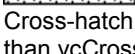
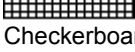


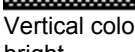


## Pattern

### Property of VcCalendarGrid

This property lets you set or retrieve the pattern of the calendar grid.

	Data Type	Explanation
Property value	VcFillPattern  <b>Possible Values:</b> .vc05PercentPattern... vc90PercentPattern 01 - 11	Pattern type  Dots in foreground color on background color, the density of the foreground color increasing with the percentage 

.vcAeroGlassPattern 44	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 twice the line width
	
.vcDashedHorizontalPattern 2026	Dashed horizontal lines
	
.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
	
.vcHorizontalPattern 3	Horizontal lines
	

.vcLargeCheckerboardPattern 2044	Checkerboard pattern showing squares of twice the size of vcSmallCheckerBoardPattern 
.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 
.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
.vcVerticalPattern 2	Vertical lines
.vcVerticalTopLightedConvexPattern 42	Vertical color gradient from bright to dark
.vcWavePattern 2031	Horizontal waves pattern
.vcWeavePattern 2034	Interwoven stripes 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

## PatternColor

**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.

Also see **set/getBackgroundColor** and **set/getPattern**.

	Data Type	Explanation
<b>Property value</b>	System.Drawing.Color	RGB color values {0...255},{0...255},{0...255}

## PatternColorDataFieldIndex

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	System.Int16	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	System.String	Name of the color map

## PatternDataFieldIndex

Property of VcCalendarGrid

This property lets you set or retrieve the data field index to be used with the property **PatternMapName**. If you set this property to **-1**, no map will be used.

	Data Type	Explanation
Property value	System.Int16	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 **PatternDataFieldIndex**, 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	System.String	Name of the pattern 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 the 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	System.Int16	Rank of Priority {-100 ... 100} <b>Default value:</b> -20

### Example Code VB.NET

```
Dim section As VcSection
Dim calendarGrid As VcCalendarGrid

section = VcGantt1.TimeScaleCollection.Active.Section(0)
calendarGrid = section.CalendarGrid(0)
calendarGrid.Priority = 3
```

### Example Code C#

```
VcSection section = vcGantt1.TimeScaleCollection.Active.get_Section(0);
VcCalendarGrid calendarGrid = section.get_CalendarGrid(0);
calendarGrid.Priority = 3;
```

## UseGraphicalAttributesOfIntervals

**Read Only 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
<b>Property value</b>	System.Boolean	Calendar grid visible/invisible <b>Default value:</b> True

### Example Code VB.NET

```
Dim section As VcSection
Dim calendarGrid As VcCalendarGrid

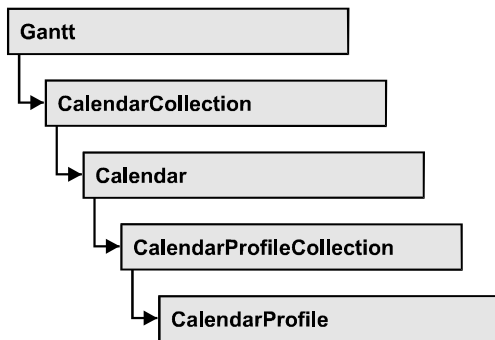
section = VcGantt1.TimeScaleCollection.Active.Section(0)
calendarGrid = section.CalendarGrid(0)
calendarGrid.Visible = False
```

### Example Code C#

```
VcSection section = vcGantt1.TimeScaleCollection.Active.get_Section(0);
VcCalendarGrid calendarGrid = section.get_CalendarGrid(0);
calendarGrid.Visible = true;
```



## 6.12 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	System.String	Name of the calendar profile

## 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	System.String	Specification of the calendar profile

## Type

### Read Only 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	VcCalendarProfileType  <b>Possible Values:</b> .vcDayProfile 4 .vcShiftProfile 5 .vcWeekProfile 3 .vcYearProfile 2	Type of the calendar profile

---

## Methods

### PutInOrderAfter

#### Method of VcCalendarProfile

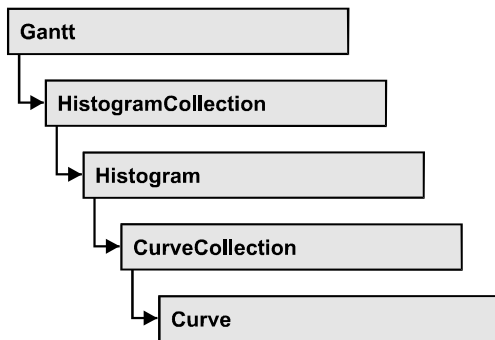
This method lets you set the calendar profile behind the calendar profile specified by name, within the CalendarProfileCollection. If you set the name

## 378 API Reference: VcCalendarProfile

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
<b>Parameter:</b> refNameParam	String	Name of the calendar profile behind which the current calendar profile is to be set

## 6.13 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
- FillReference1BackgroundColor
- FillReference1Name
- FillReference1Pattern
- FillReference1PatternColor
- FillReference2Color
- FillReference2Name
- FillReference2Pattern
- FillReference2PatternColor
- FilterName
- Histogram
- LayerName
- LineColor
- LineThickness
- LineType
- Marked
- Name
- PointsEquidistant
- Source
- Specification

- StackReferenceName
- TimeUnit
- Type
- UnitsPerStep
- ValencyDataFieldIndex
- Visible

### Methods

- Clear
- DeletePoint
- GetFirstOverload
- GetFirstOverloadEx
- GetNextOverload
- GetNextOverloadEx
- GetValues
- GetValuesEx
- SetValues

---

## Properties

### Addend

Property of VcCurve

This property lets you add the value passed to all y values of a histogram curve generated by API commands.

	Data Type	Explanation
Property value	System.Int32	Value that is added to the y values of the histogram curve

### Example Code VB.NET

```
Dim histogram As VcHistogram
Dim fixCurve As VcCurve

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
fixCurve = histogram.CurveCollection.CurveByName("Availability")

fixCurve.Addend = 1
```

**Example Code C#**

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HiSTOGRAM_1");
VcCurve fixCurve = histogram.CurveCollection.CurveByName("Availability");

fixCurve.Addend = 1;
```

**FillReference1BackgroundColor****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
<b>Property value</b>	System.Drawing.Color	RGB color values <b>Default value:</b> As defined in the <b>Edit histogram</b> dialog

**Example Code VB.NET**

```
Dim histogram As VcHistogram
Dim curve As VcCurve

histogram = vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("Curve1")

curve.FillReference1BackgroundColor = Color.Blue
```

**Example Code C#**

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HiSTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("Curve1");

curve.FillReference1BackgroundColor = Color.LightSteelBlue;
```

**FillReference1Name****Property of VcCurve**

This property lets you retrieve the name of the fill reference (for example a different curve or the x axis) of a histogram curve. The fill reference limits an area to be filled by colors and/or patterns. This property can also be set in the **Edit Histogram** dialog.

**Note:** The name of the x axis as fill reference has to be "VC\_AXIS".

## 382 API Reference: VcCurve

	Data Type	Explanation
Property value	System.String	Name of the reference curve

### Example Code VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("Curve1")
curve.FillReference1Name = "VC_AXIS"
```

### Example Code C#




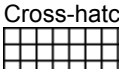
```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("Curve1");





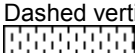
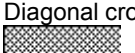
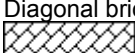
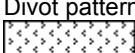

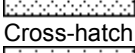
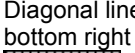
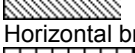
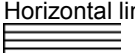
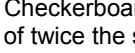

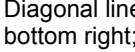

curve.FillReference1Name = "VC_AXIS";
```

## FillReference1Pattern






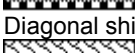




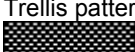


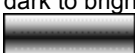





Property of VcCurve

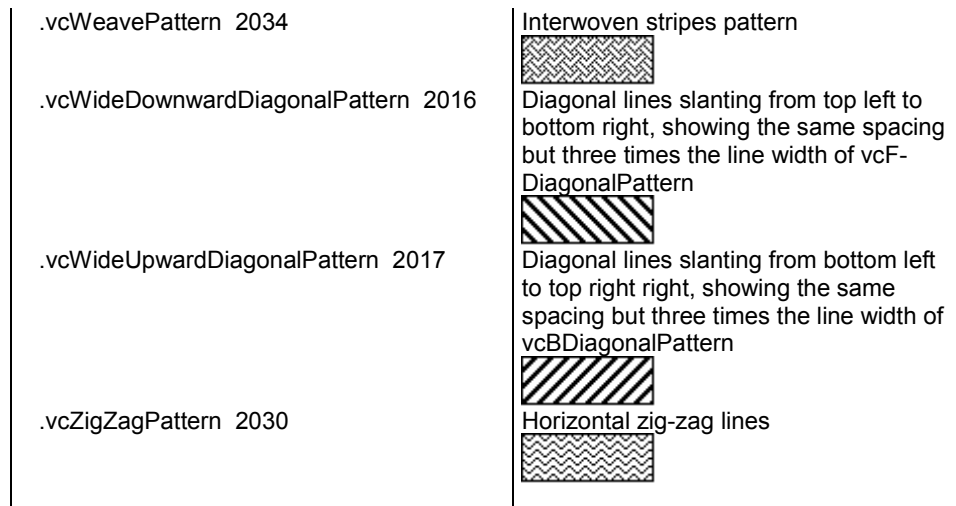
This property lets you set or retrieve the fill pattern of the area between a histogram curve and the fill reference object set. You can also set this property in the **Edit Histogram** dialog.

	Data Type	Explanation
Property value	VcFillPattern	<p>Pattern type</p> <p><b>Default value:</b> As defined in the <b>Edit histogram</b> dialog</p> <p><b>Possible Values:</b></p> <p>.vc05PercentPattern... vc90PercentPattern 01 - 11</p> <p>.vcAeroGlassPattern 44</p> <p>.vcBDiagonalPattern 5</p> <p>.vcCrossPattern 6</p>
		<p>Dots in foreground color on background color, the density of the foreground color increasing with the percentage</p>  <p>Vertical color gradient in the color of the fill pattern</p>  <p>Diagonal lines slanting from bottom left to top right</p>  <p>Cross-hatch pattern</p> 

.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 twice the line width 
.vcDashedHorizontalPattern 2026	Dashed horizontal lines 
.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 
.vcHorizontalPattern 3	Horizontal lines 
.vcLargeCheckerboardPattern 2044	Checkerboard pattern showing squares of twice the size of vcSmallCheckerBoardPattern 
.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 
.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 
.vcVerticalPattern 2	Vertical lines 
.vcVerticalTopLightedConvexPattern 42	Vertical color gradient from bright to dark 
.vcWavePattern 2031	Horizontal waves pattern 

**Example Code VB.NET**

```
Dim histogram As VcHistogram
Dim curve As VcCurve

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("Curve1")

curve.FillReference1Pattern = VcFillPattern.vcCrossPattern
```

**Example Code C#**

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HiSTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("Curve1");

curve.FillReference1Pattern = VcFillPattern.vcDiagCrossPattern;
```

**FillReference1PatternColor****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
<b>Property value</b>	System.Drawing.Color	RGB color values {0...255},{0...255},{0...255} <b>Default value:</b> As defined in the Edit histogram dialog

### Example Code VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("Curve1")

curve.FillReference1PatternColor = Color.Blue
```

### Example Code C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HiSTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("Curve1");

curve.FillReference1PatternColor = Color.LightSteelBlue;
```

## FillReference2Color

### 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	System.Drawing.Color	RGB color values <b>Default value:</b> As defined in the Edit histogram dialog

### Example Code VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("Curve1")

curve.FillReference2BackgroundColor = Color.Blue
```

### Example Code C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HiSTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("Curve1");

curve.FillReference2BackgroundColor = Color.LightSteelBlue;
```

## FillReference2Name

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	System.String	Name of the 2nd reference curve

### Example Code VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve
Dim fillRef As Object

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("Curve1")
fillRef = histogram.CurveCollection.CurveByName(curve.FillReference2Name)
```

### Example Code C#

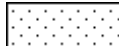
```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HiSTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("Curve1");








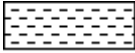


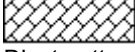
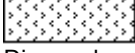

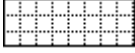



object fillRef =
histogram.CurveCollection.CurveByName(curve.FillReference2Name);
```




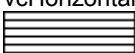








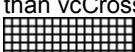





## FillReference2Pattern

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	VcFillPattern  <b>Possible Values:</b> .vc05PercentPattern... vc90PercentPattern 01 - 11	Pattern type <b>Default value:</b> As defined in the Edit histogram dialog  Dots in foreground color on background color, the density of the foreground color increasing with the percentage 

.vcAeroGlassPattern 44	<p>Vertical color gradient in the color of the fill pattern</p> 
.vcBDiagonalPattern 5	<p>Diagonal lines slanting from bottom left to top right</p> 
.vcCrossPattern 6	<p>Cross-hatch pattern</p> 
.vcDarkDownwardDiagonalPattern 2014	<p>Diagonal lines slanting from top left to bottom right; spaced 50% closer than vcFDiagonalPattern and of twice the line width</p> 
.vcDarkHorizontalPattern 2023	<p>Horizontal lines spaced 50% closer than vcHorizontalPattern and of twice the line width</p> 
.vcDarkUpwardDiagonalPattern 2015	<p>Diagonal lines slanting from bottom left to top right, spaced 50% closer than vcBDiagonalPattern and of twice the line width</p> 
.vcDarkVerticalPattern 2022	<p>Vertical lines spaced 50% closer than vcVerticalPattern and of twice the line width</p> 
.vcDashedHorizontalPattern 2026	<p>Dashed horizontal lines</p> 
.vcDashedVerticalPattern 2027	<p>Dashed vertical lines</p> 
.vcDiagCrossPattern 7	<p>Diagonal cross-hatch pattern, small</p> 
.vcDiagonalBrickPattern 2032	<p>Diagonal brick pattern</p> 
.vcDivotPattern 2036	<p>Divot pattern</p> 
.vcDottedDiamondPattern 2038	<p>Diagonal cross-hatch pattern of dotted lines</p> 
.vcDottedGridPattern 2037	<p>Cross-hatch pattern of dotted lines</p> 
.vcFDiagonalPattern 4	<p>Diagonal lines slanting from top left to bottom right</p> 
.vcHorizontalBrickPattern 2033	<p>Horizontal brick pattern</p> 
.vcHorizontalPattern 3	<p>Horizontal lines</p> 

.vcLargeCheckerboardPattern 2044	Checkerboard pattern showing squares of twice the size of vcSmallCheckerBoardPattern 
.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 
.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
.vcVerticalPattern 2	Vertical lines
.vcVerticalTopLightedConvexPattern 42	Vertical color gradient from bright to dark
.vcWavePattern 2031	Horizontal waves pattern
.vcWeavePattern 2034	Interwoven stripes 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, showing the same spacing but three times the line width of vcBDiagonalPattern
.vcZigZagPattern 2030	Horizontal zig-zag lines

### Example Code VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("Curve1")

curve.FillReference2Pattern = VcFillPattern.vcCrossPattern
```

### Example Code C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HiSTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("Curve1");

curve.FillReference2Pattern = VcFillPattern.vcDiagCrossPattern;
```

## FillReference2PatternColor

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
<b>Property value</b>	System.Drawing.Color	RGB color values ({0...255},{0...255},{0...255}) <b>Default value:</b> As defined in the Edit histogram dialog

#### Example Code VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("Curve1")

curve.FillReference2PatternColor = Color.Blue
```

#### Example Code C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HiSTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("Curve1");

curve.FillReference2PatternColor = Color.LightSteelBlue;
```

## FilterName

Property of VcCurve

This property lets you assign a filter to the curve or retrieve an existing one.

	Data Type	Explanation
<b>Property value</b>	System.String	Filter name

#### Example Code VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("Curve1")

curve.FilterName = "Critical"
```

#### Example Code C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HiSTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("Curve1");

curve.FilterName = "Critical";
```



## Histogram

Read Only Property of VcCurve

This property lets you retrieve the histogram, that the curve belongs to.

	Data Type	Explanation
Property value	VcHistogram	Histogram object

### Example Code VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve

curve =
VcGantt1.HistogramCollection.FirstHistogram.CurveCollection.CurveByName("Curve1"
)
histogram = curve.Histogram
```

### Example Code C#

```
VcCurve curve =
vcGantt1.HistogramCollection.FirstHistogram().CurveCollection.CurveByName("Curve
1");
VcHistogram histogram = curve.Histogram;
```

## LayerName

Property of VcCurve

This property lets you assign a layer to the curve or retrieve the existing one.

	Data Type	Explanation
Property value	System.String	Name of the layer

### Example Code VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("Curve1")

curve.LayerName = "Start-Ende"
```

### Example Code C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HiSTOGRAM_1");
VcCurve 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	System.Drawing.Color	RGB color values <b>Default value:</b> As defined in the <b>Edit histogram</b> dialog

### Example Code VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("Curve1")

curve.LineColor = Color.Blue
```

### Example Code C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HiSTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("Curve1");

curve.LineColor = Color.LightSteelBlue;
```

## 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	System.Int16	Line thickness LineType {1...4}: line thickness in pixels LineType {5...1000}: line thickness in 1/100 mm <b>Default value:</b> As defined in the <b>Edit histogram</b> dialog

### Example Code VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("Curve1")

curve.LineType = VcLineType.vcSolid
curve.LineThickness = 3

'or
curve.LineType = VcLineType.vcLineType5
curve.LineThickness = 20
```

### Example Code C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HiSTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("Curve1");

curve.LineType = VcLineType.vcSolid;
curve.LineThickness = 3;

//or:
curve.LineType = VcLineType.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.



## Marked

Property of VcCurve

This property lets you set or retrieve the marking status of an histogram curve set by the API.

	Data Type	Explanation
Property value	System.Boolean	Curve marked/not marked

### Example Code VB.NET

```
Dim histogram As VcHistogram
Dim fixCurve As VcCurve

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
fixCurve = histogram.CurveCollection.CurveByName("Availability")

fixCurve.Marked = True
```

### Example Code C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HiSTOGRAM_1");
VcCurve fixCurve = histogram.CurveCollection.CurveByName("Availability");

fixCurve.Marked = true;
```

## Name

Read Only Property of VcCurve

This property lets you retrieve the name of a histogram curve.

	Data Type	Explanation
Property value	System.String	Curve name

### Example Code VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve
Dim curveName As String

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("Curve1")

curveName = curve.Name
```

### Example Code C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HiSTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("Curve1");

string curveName = curve.Name;
```

## 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	System.Boolean	Curve points equidistant (True)/not equidistant (False)

## Source

Property of VcCurve

This property lets you set or retrieve 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	VcCurveSource	Calculation from field data, from dc data, from layer data, curve set
	<b>Possible Values:</b> .vcCalculateFromLayer 1 .vcSetCurve 3	Curve values calculated from layer Curve values are set manually

### Example Code VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve
Dim source As VcSource

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("Curve1")

source = curve.Source
```

### Example Code C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("Curve1");
VcSource source = curve.Source;
```

## 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 map by the method **VcCurveCollection.AddBySpecification**.

	Data Type	Explanation
Property value	System.String	Specification of the curve

## StackReferenceName

Property of VcCurve

This property lets you set or retrieve the name of the stack reference curve of a histogram curve. The stack reference name has to be specified if curves are to be stacked. It specifies the curve onto which a different curve is to be stacked. You can also set this property in the **Edit Histogram** dialog.

	Data Type	Explanation
Property value	System.String	Name of the stack curve

### Example Code VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve
Dim referenceCurve As Object

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("Curve1")

referenceCurve = histogram.CurveCollection.CurveByName(curve.StackReferenceName)
```

### Example Code C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HiSTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("Curve1");

object 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 equidistant 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
<b>Property value</b>	VcTimeUnit  <b>Possible Values:</b> .vcDay 5 .vcHour 6 .vcMinute 7 .vcSecond 8	Time unit <b>Default value:</b> -1  Time unit <b>day</b> Time unit <b>hour</b> Time unit <b>minute</b> Time unit <b>second</b>

### Example Code VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve
Dim timeUnit As VcTimeUnit

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("Curve1")

timeUnit = curve.TimeUnit
```

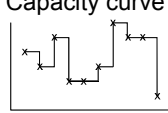
### Example Code C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("Curve1");
VcTimeUnit timeUnit = curve.TimeUnit;
```

## Type

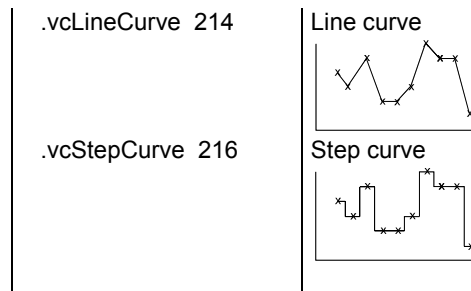
Read Only Property of VcCurve

This property lets you enquire the type of histogram curve.

	Data Type	Explanation
<b>Property value</b>	VcCurveType  <b>Possible Values:</b> .vcCapacityCurve 215	Capacity curve <b>Default value:</b> vcCapacityCurve  Capacity curve 



## 400 API Reference: VcCurve



### Example Code VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve
Dim type As VcType

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("Curve1")
type = curve.Type
```

### Example Code C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("Curve1");
VcType type = curve.Type;
```

## UnitsPerStep

Read Only Property of VcCurve

This property lets you retrieve the number of units per step of a histogram curve. The property can be applied to equidistant curves that were generated by the API only. The number can be set on the property page **General**.

	Data Type	Explanation
Property value	System.Int16	Number of units <b>Default value: -1</b>

### Example Code VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve
Dim unitsPerStep As Integer

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("Curve1")

unitsPerStep = curve.UnitsPerStep
```

### Example Code C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HiSTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("Curve1");

int unitsPerStep = curve.UnitsPerStep;
```

## 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	System.Int16	Index of the valency field

## Visible

Property of VcCurve

This property lets you set or retrieve whether a curve is visible. You can also set this property on the **Administrative Histograms** dialog.

	Data Type	Explanation
Property value	System.Boolean	Curve visible/invisible <b>Default value:</b> True

### Example Code VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("Curve1")

curve.Visible = True
```

### Example Code C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HiSTOGRAM_1");
VcCurve 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
<b>Return value</b>	Void	

#### Example Code VB.NET

```
Dim histogram As VcHistogram
Dim fixCurve As VcCurve

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
fixCurve = histogram.CurveCollection.CurveByName("Availability")

fixCurve.Clear()
```

#### Example Code C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HiSTOGRAM_1");
VcCurve fixCurve = histogram.CurveCollection.CurveByName("Availability");

fixCurve.Clear();
```

### DeletePoint

Method of VcCurve

This method lets you remove the curve point nearest to the x-coordinate.

	Data Type	Explanation
<b>Parameter:</b>		
⇒ x	System.Int32	X value of the curve point to be deleted
⇒ y	System.Int32	Y value of the curve point to be deleted
⇐ pointDate	System.DateTime	Date of the curve point which was deleted
<b>Return value</b>	System.Boolean	Curve point was/was not deleted successfully

#### Example Code VB.NET

```
Dim pointDate As Date

curve.DeletePoint(x, y, pointDate)
```

**Example Code C#**

```
DateTime pointDate = new DateTime();
curve.DeletePoint(x, y, ref pointDate);
```

**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:** For floating point numbers in the parameters **fromValue** and **toValue** please use the method **GetFirstOverloadEx**.

	Data Type	Explanation
<b>Parameter:</b>		
↔ fromDate	System.DateTime	Start date of the overload area
↔ fromValue	System.Int32	Y-value of the start date of the overload area
↔ toDate	System.DateTime	Final date of the overload area
↔ toValue	System.Int32	Y-value of the final date of the overload area
<b>Return value</b>	System.Boolean	Overload was/was not retrieved successfully

**Example Code VB.NET**

```
Dim histogram As VcHistogram
Dim curve As VcCurve
Dim fixCurve As VcCurve
Dim fromDate As Date
Dim toDate As Date
Dim fromValue As Integer
Dim toValue As Integer
Dim yValues As String
Dim bOk As Boolean

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("LayerCurve")
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 = curve.GetFirstOverload(fromDate, fromValue, toDate, toValue)
```

**Example Code C#**

```

VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("LayerCurve");
VcCurve fixCurve = histogram.CurveCollection.CurveByName("FixCurve");
DateTime fromDate = new DateTime();
DateTime toDate = new DateTime();
int fromValue = 0;
int toValue = 0;
string 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(Convert.ToDateTime("31.08.14"),yValues);
bool bOk = curve.GetFirstOverload(ref fromDate, ref fromValue, ref toDate, ref
toValue);

```

**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 **GetNextOverloadEx** for the overloads following.

**Please note:** Compared to the method **GetFirstOverload** this method allows for floating point numbers in the parameters **fromValue** and **toValue**.

	<b>Data Type</b>	<b>Explanation</b>
<b>Parameter:</b>		
↔ fromDate	System.DateTime	Start date of the overload area
↔ fromValue	System.Double	Y-value of the start date of the overload area
↔ toDate	System.DateTime	Final date of the overload area
↔ toValue	System.Double	Y-value of the final date of the overload area
<b>Return value</b>	System.Boolean	Overload was/was not retrieved successfully

**Example Code VB.NET**

```

Dim histogram As VcHistogram
Dim curve As VcCurve
Dim fixCurve As VcCurve
Dim fromDate As Date
Dim toDate As Date
Dim fromValue As Integer
Dim toValue As Integer
Dim yValues As String
Dim bOk As Boolean

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("LayerCurve")
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;0;0;6;6;6;6;6"
fixCurve.SetValues("31.08.14", yValues)
bOk = curve.GetFirstOverload(fromDate, fromValue, toDate, toValue)

```

**Example Code C#**

```

VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("LayerCurve");
VcCurve fixCurve = histogram.CurveCollection.CurveByName("FixCurve");
DateTime fromDate = new DateTime();
DateTime toDate = new DateTime();
int fromValue = 0;
int toValue = 0;
string 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;0;0;6;6;6;6;6";
fixCurve.SetValues(Convert.ToDateTime("31.08.14"), yValues);
bool bOk = curve.GetFirstOverload(ref fromDate, ref fromValue, ref toDate, ref
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:** For floating point numbers in the parameters **fromValue** and **toValue** please use the method **GetNextOverloadEx**.

	Data Type	Explanation
<b>Parameter:</b>		
↔ fromDate	System.DateTime	Start date of the overload area
↔ fromValue	System.Int32	Y-value of the start date of the overload area
↔ toDate	System.DateTime	Final date of the overload area

## 406 API Reference: VcCurve

↩ toValue	System.Int32	Y-value of the final date of the overload area
<b>Return value</b>	System.Boolean	Overload was/was not retrieved successfully.

### Example Code VB.NET

```
...
Dim bOk As Boolean

bOk = curve.GetFirstOverload(fromDate, fromValue, toDate, toValue)
MsgBox(fromDate.ToString() + " ( " + fromValues.ToString() + " ) - " +
toDate.ToString() + " ( " + toValues.ToString() + " ) ")
While bOk
    bOk = curve.GetNextOverload(fromDate, fromValue, toDate, tovalue)
    If bOk Then
        MsgBox(fromDate.ToString() + " ( " + fromValues.ToString() + " ) - " +
- " + toDate.ToString() + " ( " + toValues.ToString() + " ) ")
    End If
End While
```

### Example Code C#

```
...
bool bOk = curve.GetFirstOverload(ref fromDate, ref fromValue, ref toDate, ref
toValue);
MessageBox.Show(fromDate.ToString() + " ( " + fromValue.ToString() + " ) - " +
toDate.ToString() + " ( " + toValue.ToString() + " )");
while (bOk == true)
{
    bOk = curve.GetNextOverload(ref fromDate, ref fromValue, ref toDate, ref
toValue);
    if (bOk == true)
        MessageBox.Show(fromDate.ToString() + " ( " + fromValue.ToString() + " ) -
" + toDate.ToString() + " ( " + toValue.ToString() + " )");
}
```

## 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:** Compared to the method **GetNextOverload** this method allows for floating point numbers in the parameters **fromValue** and **toValue**.

	Data Type	Explanation
<b>Parameter:</b> ↩ fromDate	System.DateTime	Start date of the overload area

⇨ fromValue	System.Double	Y-value of the start date of the overload area
⇨ toDate	System.DateTime	Final date of the overload area
⇨ toValue	System.Double	Y-value of the final date of the overload area
<b>Return value</b>	System.Boolean	Overload was/was not retrieved successfully.

### Example Code VB.NET

```

...
Dim bOk As Boolean

bOk = curve.GetFirstOverload(fromDate, fromValue, toDate, toValue)
MsgBox(fromDate.ToString() + " ( " + fromValues.ToString() + " ) - " +
toDate.ToString() + " ( " + toValues.ToString() + " ) ")
While bOk
    bOk = curve.GetNextOverload(fromDate, fromValue, toDate, tovalue)
    If bOk Then
        MsgBox(fromDate.ToString() + " ( " + fromValues.ToString() + " )
- " + toDate.ToString() + " ( " + toValues.ToString() + " ) ")
    End If
End While

```

### Example Code C#

```

...
bool bOk = curve.GetFirstOverload(ref fromDate, ref fromValue, ref toDate, ref
toValue);
MessageBox.Show(fromDate.ToString() + " ( " + fromValue.ToString() + " ) - " +
toDate.ToString() + " ( " + toValue.ToString() + " )");
while (bOk == true)
{
    bOk = curve.GetNextOverload(ref fromDate, ref fromValue, ref toDate, ref
toValue);
    if (bOk == true)
        MessageBox.Show(fromDate.ToString() + " ( " + fromValue.ToString() + " ) -
" + toDate.ToString() + " ( " + toValue.ToString() + " )");
}

```

## 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.

	Data Type	Explanation
<b>Parameter:</b> ⇨ inputDate	System.DateTime	Date that the value of the histogram curve is to be retrieved



↩ leftDate	System.DateTime	Date of the last defined point of the curve before the specified date
↩ leftValue	System.Int32	Value of the last defined point of the curve before the specified date
↩ rightDate	System.DateTime	Date of the next defined point of the curve after the specified date
↩ rightValue	System.Int32	Value of the next defined point of the curve after the specified date
<b>Return value</b>	void	

**Example Code VB.NET**

```
Dim histogram As VcHistogram
Dim curve As VcCurve
Dim inputDate As String
Dim leftDate As Date
Dim rightDate As Date
Dim leftValues As Integer
Dim rightValues As Integer

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("LayerCurve")
inputDate = InputBox("Date: ")
curve.GetValues(inputDate, leftDate, leftValues, rightDate, rightValues)
MsgBox(leftDate.ToString() + " ( " + leftValues.ToString() + " ) " +
rightDate.ToString() + " ( " + rightValues.ToString() + " ) ")
```

**Example Code C#**

```
DateTime leftDate = new DateTime();
DateTime rightDate = new DateTime();
int leftValue = 0;
int rightValue = 0;

VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("LayerCurve");
curve.GetValues(Convert.ToDateTime("01.05.2014"), ref leftDate, ref leftValues,
ref rightDate, ref rightValues);
MessageBox.Show(leftDate.ToString() + " ( " + leftValues.ToString() + " ) " +
rightDate.ToString() + " ( " + rightValues.ToString() + " ) ");
```

**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.

	Data Type	Explanation
<b>Parameter:</b>		
⇒ inputDate	System.DateTime	Date that the value of the histogram curve is to be retrieved
⇐ leftDate	System.DateTime	Date of the last defined point of the curve before the specified date
⇐ leftValue	System.Double	Value of the last defined point of the curve before the specified date
⇐ rightDate	System.DateTime	Date of the next defined point of the curve after the specified date
⇐ rightValue	System.Double	Value of the next defined point of the curve after the specified date
<b>Return value</b>	void	

### Example Code VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve
Dim inputDate As String
Dim leftDate As Date
Dim rightDate As Date
Dim leftValues As Double
Dim rightValues As Double

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("LayerCurve")
inputDate = InputBox("Date: ")
curve.GetValuesEx(inputDate, leftDate, leftValues, rightDate, rightValues)
MsgBox(leftDate.ToString() + " ( " + leftValues.ToString() + " ) " +
rightDate.ToString() + " ( " + rightValues.ToString() + " ) ")
```

### Example Code C#

```
DateTime leftDate = new DateTime();
DateTime rightDate = new DateTime();
double leftValue = 0;
double rightValue = 0;

VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("LayerCurve");
curve.GetValuesEx(Convert.ToDateTime("01.05.2009"), ref leftDate, ref
leftValues, ref rightDate, ref rightValues);
MessageBox.Show(leftDate.ToString() + " ( " + leftValues.ToString() + " ) " +
rightDate.ToString() + " ( " + rightValues.ToString() + " ) ");
```

## 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 capacity curve to display engine resources or 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
<b>Parameter:</b>		
⇒ startDate	System.DateTime	Start date
⇒ values	System.String	Y values as a string
<b>Return value</b>	System.Boolean	Values were/were not set successfully

### Example Code VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve
Dim yValues As String

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
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("01.05.2014", yValues)

' If the option Curve points equidistant is not checked for the curve:
curve.SetValues("01.05.2014", 5)
curve.SetValues("03.05.2014", 1)
curve.SetValues("07.05.2014", 1)
curve.SetValues("16.05.2014", 2)
```

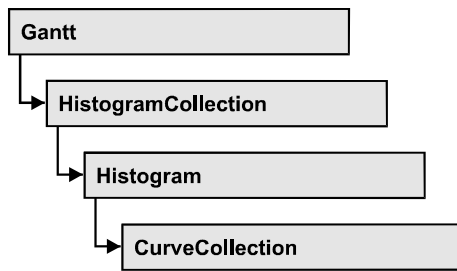
### Example Code C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HiSTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("Curve1");

//If the option Curve points equidistant is checked for the curve:
string yValues = "5;1;1;2;2;2;4;5;5;3;2;1;";
curve.SetValues(Convert.ToDateTime("01.05.2014"), yValues);

//If the option Curve points equidistant is not checked for the curve:
curve.SetValues(Convert.ToDateTime("01.05.2014"), "5");
curve.SetValues(Convert.ToDateTime("03.05.2014"), "1");
curve.SetValues(Convert.ToDateTime("07.05.2014"), "1");
curve.SetValues(Convert.ToDateTime("16.05.2014"), "2");
```

## 6.14 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

- Count

### Methods

- Add
- AddBySpecification
- Copy
- CurveByIndex
- CurveByName
- FirstCurve
- GetEnumerator
- NextCurve
- Remove

---

## Properties

### Count

**Read Only Property of VcCurveCollection**

This property lets you retrieve the number of curves in the `CurveCollection`.

## 412 API Reference: VcCurveCollection

	Data Type	Explanation
Property value	System.Int32	Number of curves

### Example Code VB.NET

```
Dim histogram As VcHistogram
Dim curveCltn As VcCurveCollection
Dim numberOfCurves As Integer

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curveCltn = histogram.CurveCollection

numberOfCurves = curveCltn.Count
```

### Example Code C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1");
VcCurveCollection curveCltn = histogram.CurveCollection;

int 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
<b>Parameter:</b> ⇒ curveName	System.String	Curve name
<b>Return value</b>	VcCurve	New curve object

### Example Code VB.NET

```
newCurve =
VcGantt1.HistogramCollection.HistogramByName("a").CurveCollection.Add("test1")
```

### Example Code C#

```
newCurve =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1").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
<b>Parameter:</b>		
⇒ specification	System.String	Curve specification
<b>Return value</b>	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
<b>Parameter:</b>		
⇒ curveName	System.String	Name of the curve to be copied
⇒ newCurveName	System.String	Name of the new curve
<b>Return value</b>	VcCurve	Curve object

### Example Code VB.NET

```
Dim histogram As VcHistogram
Dim curveCltn As VcCurveCollection

histogram = VcGantt1.HistogramCollection.FirstHistogram
curveCltn = histogram.CurveCollection
curveCltn.Copy("CurrentCurve", "NewCurve")
```

### Example Code C#

```
VcHistogram histogram = vcGantt1.HistogramCollection.FirstHistogram();
VcCurveCollection curveCltn = histogram.CurveCollection;
curveCltn.Copy("CurrentCurve", "NewCurve");
```

## CurveByIndex

Method of VcCurveCollection

This method lets you access a curve by its index. If a curve does not exist at the index specified, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	System.Int16	Index of the curve

### Example Code VB.NET

```
Dim histogram As VcHistogram
Dim curveCltn As VcCurveCollection
Dim curve As VcCurve

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curveCltn = histogram.CurveCollection

curve = curveCltn.CurveByIndex(2)
```

### Example Code C#

```
VcHistogram histogram =
VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1");
VcCurveCollection curveCltn = histogram.CurveCollection;
VcCurve 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
<b>Parameter:</b> ⇒ curveName	System.String	Name of the curve
<b>Return value</b>	VcCurve	Curve

### Example Code VB.NET

```
Dim histogram As VcHistogram
Dim curveCltn As VcCurveCollection
Dim curve As VcCurve

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curveCltn = histogram.CurveCollection

curve = curveCltn.CurveByName("Curve1")
```

**Example Code C#**

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1");
VcCurveCollection curveCltn = histogram.CurveCollection;

VcCurve 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 CurveCollection, and to continue in a forward iteration loop by the method **NextCurve** for the curves following. If there is no curve in the CurveCollection, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcCurve	First curve

**Example Code VB.NET**

```
Dim histogram As VcHistogram
Dim curveCltn As VcCurveCollection
Dim curve As VcCurve

histogram = vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curveCltn = histogram.CurveCollection

curve = curveCltn.FirstCurve
```

**Example Code C#**

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1");
VcCurveCollection curveCltn = histogram.CurveCollection;
VcCurve curve = curveCltn.FirstCurve;
```

**GetEnumerator****Method of VcCurveCollection**

This method returns an Enumerator object which supports the iteration by language specific elements. It is implied in the For...Each construct of Visual Basic and C#. This object allows to iterate over the curve objects included.

	Data Type	Explanation
Return value	VcObject	Reference object



### Example Code VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve

histogram = VcGantt1.HistogramCollection.FirstHistogram
For Each curve In histogram.CurveCollection
    ListBox1.Items.Add(curve.Name)
Next
```

### Example Code C#

```
VcHistogram histogram = vcGantt1.HistogramCollection.FirstHistogram();
foreach (VcCurve curve in histogram.CurveCollection)
    listBox1.Items.Add(curve.Name);
```

## 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	Succeeding Curve

### Example Code VB.NET

```
Dim histogram As VcHistogram
Dim curveCltn As VcCurveCollection
Dim curve As VcCurve

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curveCltn = histogram.CurveCollection

curve = curveCltn.FirstCurve

While Not newCurve Is Nothing
    newCurve = curveCltn.NextCurve
End While
```

### Example Code C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1");
VcCurveCollection curveCltn = histogram.CurveCollection;
VcCurve curve = curveCltn.FirstCurve();

while (curve == null)
{
    curve = curveCltn.NextCurve();
}
```

## 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
<b>Parameter:</b> ⇒ curveName	System.String	Curve name
<b>Return value</b>	System.Boolean	Curve deleted (True)/not deleted (False)

### Example Code VB.NET

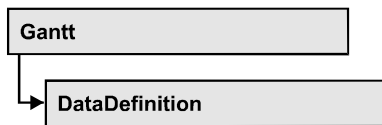
```
Dim histogram As VcHistogram
Dim curveCltn As VcCurveCollection

histogram = VcGantt1.HistogramCollection.FirstHistogram
curveCltn = histogram.CurveCollection
curveCltn.Remove("CurrentCurve")
```

### Example Code C#

```
VcHistogram histogram = vcGantt1.HistogramCollection.FirstHistogram();
VcCurveCollection curveCltn = histogram.CurveCollection;
curveCltn.Remove("CurrentCurve");
```

## 6.15 VcDataDefinition



The data of nodes and links can be defined in the dialog **Administratate 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

- DataDefinitionTable

---

## Properties

### DataDefinitionTable

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

The property DefinitionTable is an Indexed Property, which in C# can be addressed by the method `get_DataDefinitionTable(tableType)`.

	Data Type	Explanation
<b>Parameter:</b> ⇨ tableType	VcDataTableType  <b>Possible Values:</b> .vcMaindata 0 .vcRelations 1	Type of data definition table  table type <b>vcMaindata</b> (for nodes) table type <b>vcRelations</b> (for links)
<b>Property value</b>	VcDataDefinitionTable	Data definition table

**Example Code VB.NET**

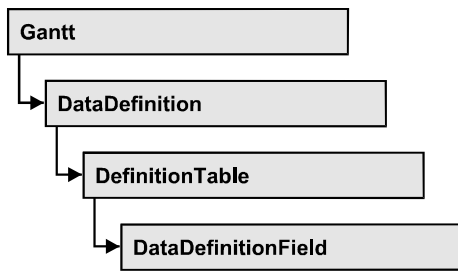
```
Dim dataDefinition As VcDataDefinition
Dim dataDefinitionTable As VcDataDefinitionTable

dataDefinition = VcGantt1.DataDefinition
dataDefinitionTable = dataDefinition.DefinitionTable(VcDataTableType.vcMaindata)
```

**Example Code C#**

```
VcDataDefinition dataDefinition = VcGantt1.DataDefinition;
VcDataDefinitionTable dataDefinitionTable =
dataDefinition.get_DefinitionTable(VcDataTableType.vcMaindata);
```

## 6.16 VcDataDefinitionField



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
- Index
- Name
- Type

---

## Properties

### DateFormat

#### Property of VcDataDefinitionField

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 **InsertLinkRecord**. The format of the date output in the chart is controlled by the property **DateOutputFormat**.

**Note:** You should set the property Type first before setting the property DateFormat.

	Data Type	Explanation
<b>Property value</b>	System.String	Date format {DMYhms;./} <b>Default value:</b> bei vcDefFieldDateTime DD.MM.YYYY hh:mm:ss

**Example Code VB.NET**

```
Dim dataDefTable As VcDataDefinitionTable
Dim dataDefField As VcDataDefinitionField

dataDefTable =
VcGanttASP1.DataDefinition.DataDefinitionTable (VcDataTableType.vcMaindata)
dataDefField = dataDefTable.DataDefinitionFieldByName ("Start")
dataDefField.Type = VcDataDefinitionFieldType.vcDefFieldDateTimeType
'DateFormat = "01.12.2014"
dataDefField.DateFormat = "DD.MM.YYYY"
VcGanttASP1.DataTableCollection.Update ()
```

**Example Code C#**

```
VcDataDefinitionTable dataDefTable =
vcGanttASP1.DataDefinition.get_DataDefinitionTable (VcDataTableType.vcMaindata);
VcDataDefinitionField dataDefField =
dataDefTable.DataDefinitionFieldByName ("Start");
dataDefField.Type = VcDataDefinitionFieldType.vcDefFieldDateTimeType;
//DateFormat = "01.12.2014"
dataDefField.DateFormat = "DD.MM.YYYY";
vcGanttASP1.DataTableCollection.Update ();
```

**Editable****Property of VcDataDefinitionField**

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
<b>Property value</b>	System.Boolean	Definition field editable/not editable <b>Default value:</b> True

**Example Code VB.NET**

```
Dim dataDefTable As VcDataDefinitionTable
Dim dataDefField As VcDataDefinitionField

dataDefTable =
VcGanttASP1.DataDefinition.DataDefinitionTable (VcDataTableType.vcMaindata)
dataDefField = dataDefTable.DataDefinitionFieldByName ("Start")
dataDefField.Editable = False
VcGanttASP1.DataTableCollection.Update ()
```

## 422 API Reference: VcDataDefinitionField

### Example Code C#

```
VcDataDefinitionTable dataDefTable =  
vcGanttASP1.DataDefinition.get_DataDefinitionTable(VcDataTableType.vcMaindata);  
VcDataDefinitionField dataDefField =  
dataDefTable.DataDefinitionFieldByName("Start");  
dataDefField.Editable = false;  
vcGanttASP1.DataTableCollection.Update();
```

## Hidden

### Property of VcDataDefinitionField

This property lets you require/set whether a data field is hidden at run time.

	Data Type	Explanation
Property value	System.Boolean	Definition field hidden/not hidden <b>Default value:</b> False

### Example Code VB.NET

```
Dim dataDefTable As VcDataDefinitionTable  
Dim dataDefField As VcDataDefinitionField  
  
dataDefTable =  
VcGanttASP1.DataDefinition.DataDefinitionTable(VcDataTableType.vcMaindata)  
dataDefField = dataDefTable.DataDefinitionFieldByName("Start")  
dataDefField.Hidden = True  
VcGanttASP1.DataTableCollection.Update()
```

### Example Code C#

```
VcDataDefinitionTable dataDefTable =  
vcGanttASP1.DataDefinition.get_DataDefinitionTable(VcDataTableType.vcMaindata);  
VcDataDefinitionField dataDefField =  
dataDefTable.DataDefinitionFieldByName("Start");  
dataDefField.Hidden = true;  
vcGanttASP1.DataTableCollection.Update();
```

## Index

### Read Only Property of VcDataDefinitionField

This property lets you retrieve the index of the field of a data definition table.

	Data Type	Explanation
Property value	System.Int16	Index of the definition field

**Example Code VB.NET**

```
Dim dataDefTable As VcDataDefinitionTable
Dim dataDefField As VcDataDefinitionField

dataDefTable =
VcGanttASP1.DataDefinition.DataDefinitionTable (VcDataTableType.vcMaindata)
dataDefField = dataDefTable.DataDefinitionFieldByName ("Start")
MsgBox (dataDefField.Index.ToString ())
```

**Example Code C#**

```
VcDataDefinitionTable dataDefTable =
vcGanttASP1.DataDefinition.get_DataDefinitionTable (VcDataTableType.vcMaindata);
VcDataDefinitionField dataDefField =
dataDefTable.DataDefinitionFieldByName ("Start");
MessageBox.Show (dataDefField.Index.ToString ());
```

## Name

**Property of VcDataDefinitionField**

This property lets you set or retrieve the name of the field of a data definition table.

	Data Type	Explanation
Property value	System.String	Name of the definition field

**Example Code VB.NET**

```
Dim dataDefTable As VcDataDefinitionTable
Dim dataDefField As VcDataDefinitionField

dataDefTable =
VcGanttASP1.DataDefinition.DataDefinitionTable (VcDataTableType.vcMaindata)
dataDefField = dataDefTable.CreateDataDefinitionField ("Start")
VcGanttASP1.DataTableCollection.Update ()
```

**Example Code C#**

```
VcDataDefinitionTable dataDefTable =
vcGanttASP1.DataDefinition.get_DataDefinitionTable (VcDataTableType.vcMaindata);
VcDataDefinitionField dataDefField =
dataDefTable.CreateDataDefinitionField ("Start");
vcGanttASP1.DataTableCollection.Update ();
```

## Type

**Property of VcDataDefinitionField**

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!



## 424 API Reference: VcDataDefinitionField

vcDefFieldAlphanumericType: DateFormat = ""

vcDefFieldDateTimeType: DateFormat = "DD.MM.YYYY hh:mm:ss"

vcDefFieldIntegerType: DateFormat = ""

	Data Type	Explanation
<b>Property value</b>	VcDataDefinitionFieldType  <b>Possible Values:</b> .vcDefFieldAlphanumericType 1 .vcDefFieldDateTimeType 4 .vcDefFieldIntegerType 2	Type of the definition field <b>Default value:</b> vcDefFieldIntegerType  Data type <b>alphanumeric</b> Data type <b>date</b> Data type <b>integer</b> (32 bits)

### Example Code VB.NET

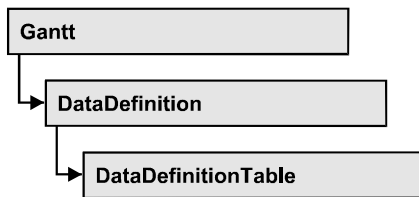
```
Dim dataDefTable As VcDataDefinitionTable
Dim dataDefField As VcDataDefinitionField

dataDefTable =
VcGanttASP1.DataDefinition.DataDefinitionTable(VcDataTableType.vcMaindata)
dataDefField = dataDefTable.DataDefinitionFieldByName("Start")
dataDefField.Type = VcDataDefinitionFieldType.vcDefFieldDateTimeType
VcGanttASP1.DataTableCollection.Update()
```

### Example Code C#

```
VcDataDefinitionTable dataDefTable =
vcGanttASP1.DataDefinition.get_DataDefinitionTable(VcDataTableType.vcMaindata);
VcDataDefinitionField dataDefField =
dataDefTable.DataDefinitionFieldByName("Start");
dataDefField.Type = VcDataDefinitionFieldType.vcDefFieldDateTimeType;
vcGanttASP1.DataTableCollection.Update();
```

## 6.17 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 **DataDefinitionFieldByIndex** or **DataDefinitionFieldByName** or retrieve them in an iterative loop by the methods **FirstDataDefinitionField** and **NextDataDefinitionField**. 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

- Count

### Methods

- CreateDataDefinitionField
- DataDefinitionFieldByIndex
- DataDefinitionFieldByName
- FirstDataDefinitionField
- GetEnumerator
- NextDataDefinitionField

---

## Properties

### Count

**Read Only Property of VcDataDefinitionTable**

This property lets you retrieve the number of fields in the data definition table. You can add fields by the **Administrate Data Tables** dialog or at run time by the method **CreateDataDefinitionField**.

## 426 API Reference: VcDataDefinitionTable

	Data Type	Explanation
Property value	System.Int32	Number of fields

### Example Code VB.NET

```
Dim dataDefinition As VcDataDefinition
Dim dataDefinitionTable As VcDataDefinitionTable
Dim numberOfFields As Integer

dataDefinition = VcGantt1.DataDefinition
dataDefinitionTable = dataDefinition.DefinitionTable(VcDataTableType.vcMaindata)

numberOfFields = dataDefinitionTable.Count
```

### Example Code C#

```
VcDataDefinition dataDefinition = vcGantt1.DataDefinition;
VcDataDefinitionTable dataDefinitionTable =
dataDefinition.get_DefinitionTable(VcDataTableType.vcMaindata);

int numberOfFields = dataDefinitionTable.Count;
```

---

## Methods

### CreateDataDefinitionField

Method of VcDataDefinitionTable

This method lets you add a new data field to the end of the data definition table at run time. The data field of the new data field is Integer. You can change the data type by the property **Type** of **VcDataDefinitionField**.

	Data Type	Explanation
<b>Parameter:</b> ⇒ newfieldName	System.String	Name of the new field
<b>Return value</b>	VcDataDefinitionField	Data definition field

### Example Code VB.NET

```
Dim dataDefinition As VcDataDefinition
Dim dataDefinitionTable As VcDataDefinitionTable

dataDefinition = VcGantt1.DataDefinition
dataDefinitionTable = dataDefinition.DefinitionTable(VcDataTableType.vcMaindata)
dataDefinitionTable.CreateDataDefinitionField("New data field 1")
VcGantt1.DataTableCollection.Update()
```

**Example Code C#**

```
VcDataDefinition dataDefinition = VcGantt1.DataDefinition;
VcDataDefinitionTable dataDefinitionTable =
dataDefinition.get_DefinitionTable(VcDataTableType.vcMaindata);
dataDefinitionTable.CreateDataDefinitionField("New data field 1");
vcGantt1.DataTableCollection.Update();
```

**DataDefinitionFieldByIndex****Method of VcDataDefinitionTable**

By this method you can access a field of the data definition table by its index. A field can be referred to by its name or by its index. You can edit data definitions in the **Administrate Data Tables** dialog.

	Data Type	Explanation
<b>Parameter:</b> ⇒ fieldIndex	System.Int16	Field index
<b>Return value</b>	VcDataDefinitionField	Data definition field

**Example Code VB.NET**

```
Dim dataDefinitionTable As VcDataDefinitionTable
Dim definitionField As VcDataDefinitionField

dataDefinitionTable =
VcGantt1.DataDefinition.DefinitionTable(VcDataTableType.vcMaindata)
definitionField = dataDefinitionTable.DataDefinitionFieldByIndex(2)
```

**Example Code C#**

```
VcDataDefinitionTable dataDefinitionTable =
VcGantt1.DataDefinition.get_DefinitionTable(VcDataTableType.vcMaindata);VcDataDe
finitionField dataDefinitionField =
dataDefinitionTable.DataDefinitionFieldByIndex(2);
```

**DataDefinitionFieldByName****Method of VcDataDefinitionTable**

By this method you can access a field of the data definition 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 edit data definitions in the **Administrate Data Tables** dialog.

	Data Type	Explanation
<b>Parameter:</b> ⇒ fieldName	System.String	Field name
<b>Return value</b>	VcDataDefinitionField	Data definition field

### Example Code VB.NET

```
Dim dataDefinitionTable As VcDataDefinitionTable
Dim definitionField As VcDataDefinitionField

dataDefinitionTable =
VcGantt1.DataDefinition.DefinitionTable(VcDataTableType.vcMaindata)
definitionField = dataDefinitionTable.DataDefinitionFieldByName("Start")
```

### Example Code C#

```
VcDataDefinitionTable dataDefinitionTable =
VcGantt1.DataDefinition.get_DefinitionTable(VcDataTableType.vcMaindata);VcDataDe
finitionField dataDefinitionField =
dataDefinitionTable.DataDefinitionFieldByName("Start");
```

## FirstDataDefinitionField

### Method of VcDataDefinitionTable

This method can be used to access the first field of a data definition table and to continue in a forward iteration loop by the method **NextDataDefinitionField** for the fields following. If there is no field in the data definition table, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcDataDefinitionField	First Data definition field

### Example Code VB.NET

```
Dim dataDefinition As VcDataDefinition
Dim dataDefinitionTable As VcDataDefinitionTable
Dim dataDefinitionField As VcDataDefinitionField

Set dataDefinition = VcGanttASP1.DataDefinition
Set dataDefinitionTable = dataDefinition.DefinitionTable(vcMaindata)

Set dataDefinitionField = dataDefinitionTable.FirstDataDefinitionField
```

### Example Code C#

```
VcDataDefinition dataDefinition = vcGantt1.DataDefinition;
VcDataDefinitionTable dataDefinitionTable =
vcGantt1.DataDefinition.get_DefinitionTable(VcDataTableType.vcMaindata);
VcDataDefinitionField dataDefinitionField =
dataDefinitionTable.FirstDataDefinitionField();
```

## GetEnumerator

**Method of VcDataDefinitionTable**

This method returns an Enumerator object which supports the iteration by language specific elements. It is implied in the For...Each construct of Visual Basic and C#. This object allows to iterate over the fields of a dataDefinitionTable.

	Data Type	Explanation
<b>Return value</b>	VcObject	Enumerator object

### Example Code VB.NET

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable

dataTableCltn = VcGanttASP1.DataTableCollection
For Each dataTable In dataTableCltn
    ListBox1.Items.Add(dataTable.Name)
Next
```

### Example Code C#

```
VcDataTableCollection dataTableCltn = vcGanttASP1.DataTableCollection;
foreach (VcDataTable dataTable in dataTableCltn)
    listBox1.Items.Add(dataTable.Name);
```

## NextDataDefinitionField

**Method of VcDataDefinitionTable**

This method can be used in a forward iteration loop to retrieve subsequent fields from a data definition table after initializing the loop by the method **FirstDataDefinitionField**. If there is no field left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
<b>Return value</b>	VcDataDefinitionField	Subsequent data definition field

## 430 API Reference: VcDataDefinitionTable

### Example Code VB.NET

```
Dim dataDefinition As VcDataDefinition
Dim dataDefinitionTable As VcDataDefinitionTable
Dim definitionField As VcDataDefinitionField

dataDefinition = VcGantt1.DataDefinition
dataDefinitionTable =
VcGantt1.DataDefinition.DefinitionTable(VcDataTableType.vcMaindata)
definitionField = dataDefinitionTable.FirstDataDefinitionField

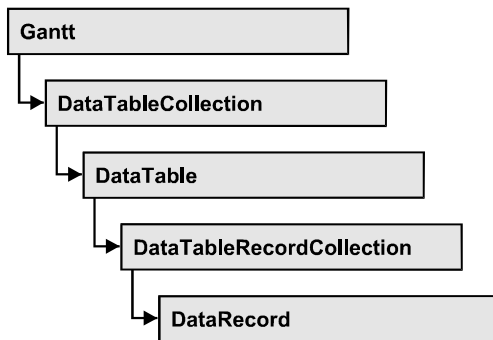
While Not definitionField Is Nothing
    ListBox1.Items.Add(definitionField.Name)
    definitionField = dataDefinitionTable.NextField
End While
```

### Example Code C#

```
VcDataDefinition dataDefinition = vcGantt1.DataDefinition;
VcDataDefinitionTable dataDefinitionTable =
vcGantt1.DataDefinition.get_DefinitionTable(VcDataTableType.vcMaindata);
VcDataDefinitionField dataDefinitionField =
dataDefinitionTable.FirstDataDefinitionField();

while (dataDefinitionField != null)
{
    ListBox.Items.Add(dataDefinitionField.Name);
    dataDefinitionField = dataDefinitionTable.NextDataDefinitionField;
}
```

## 6.18 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

- Delete
- RelatedDataRecord
- Update

---

## 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 "object" are allowed, that contains all data fields of the record in an array. When retrieving the property, a string will be returned.



## 432 API Reference: VcDataRecord

	Data Type	Explanation
<b>Property value</b>	System.Object	All data of the data record

### Example Code VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecCltn As VcDataRecordCollection
Dim dataRecValue() As Object
Dim dataRecord As VcDataRecord

dataTable = VcGanttASP1.DataTableCollection.DataTableByName("Maindata1")
dataRecCltn = dataTable.DataRecordCollection
ReDim dataRecValue(dataTable.DataTableFieldCollection.Count)
dataRecValue(0) = 1
dataRecValue(1) = "Node One"

'Object
dataRecord = dataRecCltn.Add(dataRecValue)
'CSV
dataRecord.AllData = "1;Node One;"

dataRecord.Update()
```

### Example Code C#

```
VcDataTable dataTable =
vcGanttASP1.DataTableCollection.DataTableByName("Maindata");
VcDataRecordCollection dataRecordCltn = dataTable.DataRecordCollection;
Object [] dataRecVal = new object[dataTable.DataTableFieldCollection.Count];

dataRecVal[0] = 1;
dataRecVal[1] = "Node One";

//Object
VcDataRecord dataRecord = dataRecordCltn.Add(dataRecVal);
//CSV
dataRecord.AllData = "1;Node One;";

dataRecord.Update();
```

## 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.

The property DataField is an Indexed Property, which in C# is addressed by the methods set\_DataField (index, pvn) and get\_DataField (index).

	Data Type	Explanation
<b>Parameter:</b> ⇒ index	System.Int16	Index of data field

<b>Property value</b>	System.Object	Content of the data field
-----------------------	---------------	---------------------------

**Example Code VB.NET**

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

dataTable = VcGanttASP1.DataTableCollection.FirstDataTable
dataRecordCltn = dataTable.DataRecordCollection
dataRecord = dataRecordCltn.DataRecordByID(1)

dataRecord.DataField(1) = "Node Two"
dataRecord.Update()
```

**Example Code C#**

```
VcDataTable dataTable = vcGanttASP1.DataTableCollection.FirstDataTable();
VcDataRecordCollection dataRecordCltn = dataTable.DataRecordCollection;
VcDataRecord dataRecord = dataRecordCltn.DataRecordByID(1);
dataRecord.set_DataField(1, "Node Two");
dataRecord.Update();
```

**DataTableName****Read Only Property of VcDataRecord**

This property lets you retrieve the name of the data table that this data record belongs to.

	<b>Data Type</b>	<b>Explanation</b>
<b>Property value</b>	System.String	Name of the associated table

**Example Code VB.NET**

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

dataTable = VcGanttASP1.DataTableCollection.FirstDataTable
dataRecordCltn = dataTable.DataRecordCollection
dataRecord = dataRecordCltn.DataRecordByID(1)

MsgBox(dataRecord.DataTableName)
```

**Example Code C#**

```
VcDataTable dataTable = vcGanttASP1.DataTableCollection.FirstDataTable();
VcDataRecordCollection dataRecordCltn = dataTable.DataRecordCollection;
VcDataRecord dataRecord = dataRecordCltn.DataRecordByID(1);

MessageBox.Show(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	System.String	Data record ID

### Example Code VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord
dataTable = VcGanttASP1.DataTableCollection.FirstDataTable
dataRecordCltn = dataTable.DataRecordCollection
dataRecord = dataRecordCltn.DataRecordByID(1)
MsgBox(dataRecord.ID)
```

### Example Code C#

```
VcDataTable dataTable = vcGanttASP1.DataTableCollection.FirstDataTable();
VcDataRecordCollection dataRecordCltn = dataTable.DataRecordCollection;
VcDataRecord dataRecord = dataRecordCltn.DataRecordByID(1);
MessageBox.Show(dataRecord.ID);
```

## Methods

### Delete

Method of VcDataRecord

This method lets you delete a data record.

	Data Type	Explanation
Return value	System.Boolean	Data record was (true) / was not (false) deleted successfully

### Example Code VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

dataTable = VcGanttASP1.DataTableCollection.FirstDataTable
dataRecordCltn = dataTable.DataRecordCollection
dataRecord = dataRecordCltn.DataRecordByID(1)

dataRecord.Delete()
```

**Example Code C#**

```
VcDataTable dataTable = vcGanttASP1.DataTableCollection.FirstDataTable();
VcDataRecordCollection dataRecordCltn = dataTable.DataRecordCollection;
VcDataRecord dataRecord = dataRecordCltn.DataRecordByID(1);

dataRecord.Delete();
```

**RelatedDataRecord****Method of VcDataRecord**

This property lets you relate a data record to a different one or retrieve a related data set. 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
<b>Parameter:</b>		
⇒ index	System.Int16	Index of data field
<b>Return value</b>	VcDataRecord	Related data record

**Example Code VB.NET**

```
Private Sub VcGanttASP1_VcNodeLeftClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcNodeClickingEventArgs) Handles VcGanttASP1.VcNodeLeftClicking
    Dim dataTable As VcDataTable
    Dim dataRecordCltn As VcDataRecordCollection
    Dim firstDataRecord As VcDataRecord
    Dim secondDataRecord As VcDataRecord

    dataTable = VcGanttASP1.DataTableCollection.DataTableByIndex(0)
    dataRecordCltn = dataTable.DataRecordCollection

    firstDataRecord = dataRecordCltn.DataRecordByID(e.Node.DataField(0))
    secondDataRecord = firstDataRecord.RelatedDataRecord(2)

    MsgBox(secondDataRecord.AllData)
End Sub
```

**Example Code C#**

```
private void vcGanttASP1_VcNodeLeftClicking(object sender,
NETRONIC.XGantt.VcNodeClickingEventArgs e)
{
    VcDataTable dataTable = vcGanttASP1.DataTableCollection.DataTableByIndex(0);
    VcDataRecordCollection dataRecordCltn = dataTable.DataRecordCollection;
    VcDataRecord firstDataRecord =
dataRecordCltn.DataRecordByID(e.Node.get_DataField(0));
    VcDataRecord secondDataRecord = firstDataRecord.RelatedDataRecord(2);

    MessageBox.Show(secondDataRecord.AllData.ToString());
}
```

## Update

### 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
<b>Return value</b>	System.Boolean	Data record was (true) / was not (false) updated successfully

### Example Code VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

dataTable = VcGanttASP1.DataTableCollection.FirstDataTable
dataRecordCltn = dataTable.DataRecordCollection
dataRecord = dataRecordCltn.DataRecordByID(1)

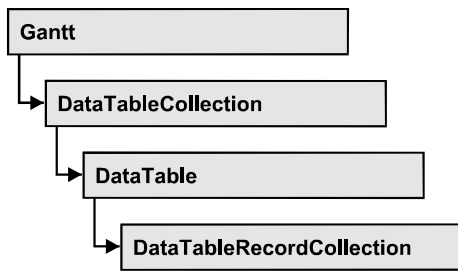
dataRecord.DataField(1) = "Node Two"
dataRecord.Update()
```

### Example Code C#

```
VcDataTable dataTable = vcGanttASP1.DataTableCollection.FirstDataTable();
VcDataRecordCollection dataRecordCltn = dataTable.DataRecordCollection;
VcDataRecord dataRecord = dataRecordCltn.DataRecordByID(1);

dataRecord.set_DataField(1, "Node Two");
dataRecord.Update();
```

## 6.19 VcDataRecordCollection



An object of the type `VcDataRecordCollection` contains the 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

- Count

### Methods

- Add
- DataRecordByID
- FirstDataRecord
- GetEnumerator
- GetNewUniqueID
- NextDataRecord
- Remove
- Update

---

## Properties

### Count

**Read Only Property of VcDataRecordCollection**

This property lets you retrieve the number of data records in the `DataRecordCollection` object.

## 438 API Reference: VcDataRecordCollection

	Data Type	Explanation
Property value	System.Int32	Number of data records in the collection object

### Example Code VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection

dataTable = VcGanttASP1.DataTableCollection.DataTableByName("Maindata")
dataRecordCltn = dataTable.DataRecordCollection
MsgBox("Number of DataRecords: " & dataRecordCltn.Count)
```

### Example Code C#

```
VcDataTable dataTable =
vcGanttASP1.DataTableCollection.DataTableByName("Maindata");
VcDataRecordCollection dataRecordCltn = dataTable.DataRecordCollection;
MessageBox.Show("Number of DataRecords: " + dataRecordCltn.Count);
```

---

## Methods

### Add

#### Method of VcDataRecordCollection

By this method you can create a data record as a member of the DataRecordCollection. If the ID was not used before, the new data record will be returned; otherwise a **VcPrimaryKeyNotUniqueException** will be thrown.

	Data Type	Explanation
<b>Parameter:</b> ⇒ dataRecordContent	VcObject	Content of the data record (as an array or a string)
<b>Return value</b>	VcDataRecord	Data record created

**Example Code VB.NET**

```

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 Object

dataTable = VcGanttASP1.DataTableCollection.DataTableByName("Maindata")
dataRecCltn = dataTable.DataRecordCollection

Dim dataRec1 As VcDataRecord
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
dataRec1 = dataRecCltn.Add(dataRecVal)

' equivalent
' dataRec1 = dataRecCltn.Add("1;Node 1;01.08.14;;8")

```

**Example Code C#**

```

const int Main_ID = 0;
const int Main_Name = 1;
const int Main_Start = 2;
const int Main_Duration = 4;

//...

VcDataTable dataTable =
vcGanttASP1.DataTableCollection.DataTableByName("Maindata");

VcDataRecordCollection dataRecCltn = dataTable.DataRecordCollection;

Object [] dataRecVal = new object[dataTable.DataTableFieldCollection.Count];

VcDataRecord dataRec1;

dataRecVal[Main_ID] = "1";
dataRecVal[Main_Name] = "Node 1";
dataRecVal[Main_Start] = "08.01.2014";
dataRecVal[Main_Duration] = 8;
dataRec1 = dataRecCltn.Add(dataRecVal);

// equivalent
// dataRec2 = 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
<b>Parameter:</b> ⇒ dataRecordID	System.String	ID of the data record
<b>Return value</b>	VcDataRecord	Data record object

### Example Code VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

dataTable = VcGanttASP1.DataTableCollection.DataTableByName("Maindata")
dataRecordCltn = dataTable.DataRecordCollection
dataRecord = dataRecordCltn.DataRecordByID(0)
```

### Example Code C#

```
VcDataTable dataTable =
vcGanttASP1.DataTableCollection.DataTableByName("Maindata");
VcDataRecordCollection dataRecordCltn = dataTable.DataRecordCollection;
VcDataRecord 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
<b>Return value</b>	VcDataRecord	First data record

### Example Code VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

dataTable = VcGanttASP1.DataTableCollection.DataTableByName("Maindata")
dataRecordCltn = dataTable.DataRecordCollection
dataRecord = dataRecordCltn.FirstDataRecord
```

**Example Code C#**

```
VcDataTable dataTable =
vcGanttASP1.DataTableCollection.DataTableByName("Maindata");
VcDataRecordCollection dataRecordCltn = dataTable.DataRecordCollection;
VcDataRecord dataRecord = dataRecordCltn.FirstDataRecord();
```

**GetEnumerator****Method of VcDataRecordCollection**

This method returns an Enumerator object which supports the iteration by language specific elements. It is implied in the For...Each construct of Visual Basic and C#. This object allows to iterate over the data records included.

	Data Type	Explanation
<b>Return value</b>	VcObject	Enumerator object

**Example Code VB.NET**

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

dataTable = vcGanttASP1.DataTableCollection.DataTableByName("Maindata")
dataRecordCltn = dataTable.DataRecordCollection

vcGanttASP1.SuspendUpdate(True)

dataRecord = dataRecordCltn.FirstDataRecord
While Not dataRecord Is Nothing
    dataRecord.DataField(4) = "10"
    dataRecord.Update()
    dataRecord = dataRecordCltn.NextDataRecord
End While

vcGanttASP1.SuspendUpdate(False)
```

**Example Code C#**

```
VcDataTable dataTable =
vcGanttASP1.DataTableCollection.DataTableByName("Maindata");
VcDataRecordCollection dataRecordCltn = dataTable.DataRecordCollection;

vcGanttASP1.SuspendUpdate(true);

foreach (VcDataRecord dataRecord in dataRecordCltn)
{
    dataRecord.set_DataField(4, "10");
    dataRecord.Update();
    dataRecordCltn.NextDataRecord();
}

vcGanttASP1.SuspendUpdate(false);
```

## 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	System.Int32	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	Succeeding data record

### Example Code VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

dataTable = VcGanttASP1.DataTableCollection.DataTableByName("Maindata")
dataRecordCltn = dataTable.DataRecordCollection

VcGanttASP1.SuspendUpdate(True)

dataRecord = dataRecordCltn.FirstDataRecord
While Not dataRecord Is Nothing
    dataRecord.DataField(4) = "10"
    dataRecord.Update()
    dataRecord = dataRecordCltn.NextDataRecord
End While

VcGanttASP1.SuspendUpdate(False)
```

**Example Code C#**

```

VcDataTable dataTable =
vcGanttASP1.DataTableCollection.DataTableByName("Maindata");
VcDataRecordCollection dataRecordCltn = dataTable.DataRecordCollection;

vcGanttASP1.SuspendUpdate(true);

foreach (VcDataRecord dataRecord in dataRecordCltn)
{
    dataRecord.set_DataField(4, "10");
    dataRecord.Update();
    dataRecordCltn.NextDataRecord();
}

vcGanttASP1.SuspendUpdate(false);

```

**Remove****Method of VcDataRecordCollection**

This method lets you delete a data record. The method always returns **true**. The content of the data record is used to identify the object by its identification.

	Data Type	Explanation
<b>Parameter:</b>		
⇒ dataRecordContent	VcObject	Content of the data record (as an array or a string)
<b>Return value</b>	System.Boolean	true

**Example Code VB.NET**

```

Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

dataTable = vcGanttASP1.DataTableCollection.DataTableByName("Maindata")
dataRecordCltn = dataTable.DataRecordCollection
dataRecordCltn.Remove("1;1Activity; Y;Z;18.01.14;;5")

' equivalent
' dataRecord = dataRecordCltn.DataRecordByID(1)
' dataRecord.Delete()
' dataRecord.Update()

```

**Example Code C#**

```

VcDataTable dataTable =
vcGanttASP1.DataTableCollection.DataTableByName("Maindata");
VcDataRecordCollection dataRecCltn = dataTable.DataRecordCollection;

dataRecCltn.Remove("1;1Activity Y;Z;18.01.14;;5");

// equivalent
// VcDataRecord dataRecord = dataRecordCltn.DataRecordByID(1);
// dataRecord.Delete();
// dataRecord.Update();

```

## 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()**.

	Data Type	Explanation
<b>Parameter:</b>		
⇒ dataRecordContent	VcObject	Content of the data record (as an array or a string)
<b>Return value</b>	System.Boolean	Update successful (true) / not successful (false)

### Example Code VB.NET

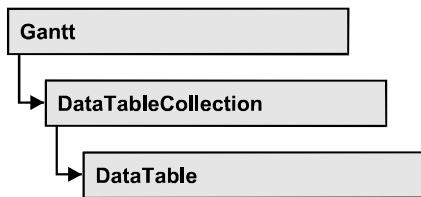
```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

dataTable = VcGanttASP1.DataTableCollection.DataTableByName("Maindata")
dataRecordCltn = dataTable.DataRecordCollection
dataRecordCltn.Update("1;1.8.2012;;8")
```

### Example Code C#

```
VcDataTable dataTable =
vcGanttASP1.DataTableCollection.DataTableByName("Maindata");
VcDataRecordCollection dataRecCltn = dataTable.DataRecordCollection;
dataRecCltn.Update("1;1.8.2012;;8")
```

## 6.20 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 VB.NET

```

Dim dataTable As VcDataTable

dataTable = VcGanttASP1.DataTableCollection.FirstDataTable()
MsgBox(dataTable.DataRecordCollection.Count)
  
```

### Example Code C#

```
VcDataTable dataTable = vcGanttASP1.DataTableCollection.FirstDataTable();  
MessageBox.Show(dataTable.DataRecordCollection.Count.ToString());
```

## 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	VcDataTableFieldCollection	DataTableFieldCollection object

### Example Code VB.NET

```
Dim dataTable As VcDataTable  
  
dataTable = VcGanttASP1.DataTableCollection.DataTableByIndex(0)  
MsgBox(dataTable.DataTableFieldCollection.Count)
```

### Example Code C#

```
VcDataTable dataTable = vcGanttASP1.DataTableCollection.DataTableByIndex(0);  
MessageBox.Show(dataTable.DataTableFieldCollection.Count.ToString());
```

## 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	System.String	Description of the data table <b>Default value:</b> Empty string

**Example Code VB.NET**

```
Dim dataTable As VcDataTable

dataTable = VcGanttASP1.DataTableCollection.DataTableByName("Maindata")
dataTable.Description = "This table contains data for nodes"
```

**Example Code C#**

```
VcDataTable dataTable =
vcGanttASP1.DataTableCollection.DataTableByName("Maindata");
dataTable.Description = "This table contains data for nodes";
```

**MultiplePrimaryKeysAllowed****Property of VcDataTable**

With this property you can set or retrieve whether the use of composite primary keys is possible.

	Data Type	Explanation
<b>Property value</b>	System.Boolean	Use of composite primary keys allowed (true)/not allowed (false) <b>Default value:</b> False

**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 **DataTableCollection** you can retrieve a reference to the data table object.

	Data Type	Explanation
<b>Property value</b>	System.String	Name of the data table <b>Default value:</b> Empty string

**Example Code VB.NET**

```
Dim dataTable As VcDataTable

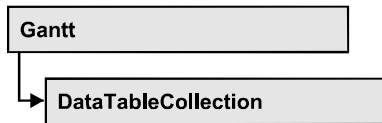
dataTable = VcGanttASP1.DataTableCollection.DataTableByIndex(0)
MsgBox(dataTable.Name)
```

**Example Code C#**

```
VcDataTable dataTable = vcGanttASP1.DataTableCollection.DataTableByIndex(0);
MessageBox.Show(dataTable.Name);
```



## 6.21 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.

### Properties

- Count

### Methods

- Add
- Copy
- DataTableByIndex
- DataTableByName
- FirstDataTable
- GetEnumerator
- NextDataTable
- Update

---

## Properties

### Count

Read Only Property of VcDataTableCollection

This property lets you retrieve the number of data tables in the DataTableCollection object.

	Data Type	Explanation
Property value	System.Int32	Number of data tables in the collection object

**Example Code VB.NET**

```
Dim dataTableCltn As VcDataTableCollection

dataTableCltn = VcGanttASP1.DataTableCollection
MsgBox(dataTableCltn.Count.ToString())
```

**Example Code C#**

```
VcDataTableCollection dataTableCltn = vcGanttASP1.DataTableCollection;
MessageBox.Show(dataTableCltn.Count.ToString());
```

---

## Methods

### Add

**Method of VcDataTableCollection**

By this method you can create a data table as a member of the DataTableCollection. If the name was not used before, an object of the type **VcDataTable** will be returned; otherwise "Nothing" (in Visual Basic) or "0" (in other languages) will be returned. Only if the property **ExtendedDataTables** is set to **True**, tables can be added.

	Data Type	Explanation
<b>Parameter:</b>		
⇒ dataTableName	System.String	Name of the new data table
<b>Return value</b>	VcDataTable	Data table generated

**Example Code VB.NET**

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable

dataTableCltn = VcGanttASP1.DataTableCollection
dataTable = dataTableCltn.Add("Resources")
dataTableCltn.Update()
```

**Example Code C#**

```
VcDataTableCollection dataTableCltn = vcGanttASP1.DataTableCollection;
VcDataTable 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 property **ExtendedDataTables**

## 450 API Reference: VcDataTableCollection

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
<b>Parameter:</b>		
⇒ dataTableName	System.String	Name of the data table to be copied (source table)
⇒ newDataTableName	System.String	Name of the data table to be generated (target table)
<b>Return value</b>	VcDataTable	Data table object generated

### Example Code VB.NET

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable

dataTableCltn = VcGanttASp1.DataTableCollection
dataTable = dataTableCltn.Copy("Resources", "NewResources")
dataTableCltn.Update()
```

### Example Code C#

```
VcDataTableCollection dataTableCltn = vcGanttASp1.DataTableCollection;
VcDataTable 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
<b>Parameter:</b>		
⇒ index	System.Int16	Index of the data table
<b>Return value</b>	VcDataTable	Data table object returned

### Example Code VB.NET

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable

dataTableCltn = VcGanttASp1.DataTableCollection
dataTable = dataTableCltn.DataTableByIndex(2)
MsgBox(dataTable.Name)
```

**Example Code C#**

```
VcDataTableCollection dataTableCltn = vcGanttASP1.DataTableCollection;
VcDataTable dataTable = dataTableCltn.DataTableByIndex(2);
MessageBox.Show(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
<b>Parameter:</b>		
⇒ dataTableName	System.String	Name of the data table
<b>Return value</b>	VcDataTable	Data table object returned

**Example Code VB.NET**

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable

dataTableCltn = VcGanttASP1.DataTableCollection
dataTable = dataTableCltn.DataTableByName("Resources")
MsgBox(dataTable.Description)
```

**Example Code C#**

```
VcDataTableCollection dataTableCltn = vcGanttASP1.DataTableCollection;
VcDataTable dataTable = dataTableCltn.DataTableByName("Resources");
MessageBox.Show(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
<b>Return value</b>	VcDataTable	First data table

## 452 API Reference: VcDataTableCollection

### Example Code VB.NET

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable

dataTableCltn = VcGanttASP1.DataTableCollection
dataTable = dataTableCltn.FirstDataTable
```

### Example Code C#

```
VcDataTableCollection dataTableCltn = vcGanttASP1.DataTableCollection;
VcDataTable dataTable= dataTableCltn.FirstDataTable();
```

## GetEnumerator

### Method of VcDataTableCollection

This method returns an Enumerator object which supports the iteration by language specific elements. It is implied in the For...Each construct of Visual Basic and C#. This object allows to iterate over the data tables included.

	Data Type	Explanation
Return value	VcObject	Enumerator object

### Example Code VB.NET

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable

dataTableCltn = VcGanttASP1.DataTableCollection
For Each dataTable In dataTableCltn
    ListBox1.Items.Add(dataTable.Name)
Next
```

### Example Code C#

```
VcDataTableCollection dataTableCltn = vcGanttASP1.DataTableCollection;
foreach (VcDataTable dataTable in dataTableCltn)
    listBox1.Items.Add(dataTable.Name);
```

## 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	Succeeding data table

**Example Code VB.NET**

```

Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable
Dim i As Integer

dataTableCltn = VcGanttASP1.DataTableCollection
dataTable = dataTableCltn.FirstDataTable
For i = 1 To dataTableCltn.Count
    ListBox1.Items.Add(dataTable.Name)
    dataTable = dataTableCltn.NextDataTable
Next

```

**Example Code C#**

```

VcDataTableCollection dataTableCltn = vcGanttASP1.DataTableCollection;
VcDataTable dataTable = dataTableCltn.FirstDataTable();
for (int i=0; i<dataTableCltn.Count; i++)
{
    listBox1.Items.Add(dataTable.Name);
    dataTable = dataTableCltn.NextDataTable();
}

```

## 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
<b>Return value</b>	System.Boolean	Update successful (true) / not successful (false)

**Example Code VB.NET**

```

Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable

dataTableCltn = VcGanttASP1.DataTableCollection
dataTable = dataTableCltn.Add("Resources")
dataTable.DataTableFieldCollection.Add("Id")
dataTableCltn.Update()

```

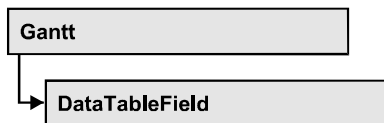
**Example Code C#**

```

VcDataTableCollection dataTableCltn = vcGanttASP1.DataTableCollection;
VcDataTable dataTable = dataTableCltn.Add("Resources");
dataTable.DataTableFieldCollection.Add("Id");
dataTableCltn.Update();

```

## 6.22 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	System.String	Name of the data table

**Example Code VB.NET**

```
Dim dataTable As VcDataTable

dataTable = VcGanttASP1.DataTableCollection.FirstDataTable
MsgBox(dataTable.DataTableFieldCollection.FirstDataTableField.DataTableName)
```

**Example Code C#**

```
VcDataTable dataTable = vcGanttASP1.DataTableCollection.FirstDataTable();
MessageBox.Show(dataTable.DataTableFieldCollection.FirstDataTableField().DataTab
leName);
```

## DateFormat

**Read Only 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
<b>Property value</b>	System.String	Date format {DMYhms:;./}

**Example Code VB.NET**

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

dataTable = VcGanttASP1.DataTableCollection.DataTableByName("Operation")
dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Start")
dataTableField.Type = VcDataTableFieldType.VcDataTableFieldDateTimeType
'DateFormat = "01.12.2014"
dataTableField.DateFormat = "DD.MM.YYYY"
VcGanttASP1.DataTableCollection.Update()
```

**Example Code C#**

```
VcDataTable dataTable =
vcGanttASP1.DataTableCollection.DataTableByName("Operation");
VcDataTableField dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Start");
dataTableField.Type = VcDataTableFieldType.VcDataTableFieldDateTimeType;
//DateFormat = "01.12.2014"
dataTableField.DateFormat = "DD.MM.YYYY";
vcGanttASP1.DataTableCollection.Update();
```



## 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
<b>Property value</b>	System.Boolean	Field editable (True) / not editable (False) <b>Default value:</b> True

### Example Code VB.NET

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

dataTable = VcGanttASP1.DataTableCollection.DataTableByName("Operation")
dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Start")
dataTableField.Editable = False
VcGanttASP1.DataTableCollection.Update()
```

### Example Code C#

```
VcDataTable dataTable =
vcGanttASP1.DataTableCollection.DataTableByName("Operation");
VcDataTableField dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Start");
dataTableField.Editable = false;
VcGanttASP1.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
<b>Property value</b>	System.Boolean	Field hidden (True) / not hidden (False) <b>Default value:</b> False

### Example Code VB.NET

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

dataTable = VcGanttASP1.DataTableCollection.DataTableByName("Operation")
dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Start")
dataTableField.Hidden = True
VcGanttASP1.DataTableCollection.Update()
```

**Example Code C#**

```
VcDataTable dataTable =
vcGanttASP1.DataTableCollection.DataTableByName("Operation");
VcDataTableField dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Start");
dataTableField.Hidden = true;
vcGanttASP1.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
<b>Property value</b>	System.Int16	Index of the data table field

**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
<b>Property value</b>	System.String	Name of the field <b>Default value:</b> Empty string

**Example Code VB.NET**

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

dataTable = vcGanttASP1.DataTableCollection.DataTableByName("Operation")
dataTableField = dataTable.DataTableFieldCollection.Add("Start")
vcGanttASP1.DataTableCollection.Update()
```

**Example Code C#**

```
VcDataTable dataTable =
vcGanttASP1.DataTableCollection.DataTableByName("Operation");
VcDataTableField dataTableField =
dataTable.DataTableFieldCollection.Add("Start");
vcGanttASP1.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
<b>Property value</b>	System.Boolean	The field serves (True) / does not serve (False) as a primary key. <b>Default value:</b> False

### Example Code VB.NET

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField
Dim isPrimaryKey As Boolean

dataTable = VcGanttASP1.DataTableCollection.DataTableByName("Operation")
dataTableField = dataTable.DataTableFieldCollection.DataTableFieldByName("Id")
dataTableField.PrimaryKey = True
VcGanttASP1.DataTableCollection.Update()
```

### Example Code C#

```
VcDataTable dataTable =
vcGanttASP1.DataTableCollection.DataTableByName("Operation");
VcDataTableField dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Id");
dataTableField.PrimaryKey = true;
vcGanttASP1.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
<b>Property value</b>	System.Int32	Index of the record field to which the data definition of the data table field refers. <b>Default value:</b> -1

**Example Code VB.NET**

```

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 = VcGanttASP1.DataTableCollection.Add("Task")
dataTaskFieldId = dataTableTask.DataTableFieldCollection.Add("Id")
dataTaskFieldId.PrimaryKey = True
dataTaskFieldName = dataTableTask.DataTableFieldCollection.Add("Name")
dataTaskFieldName.Type = VcDataTableFieldType.vcDataTableFieldStringType

'Create table Operation
dataTableOperation = VcGanttASP1.DataTableCollection.Add("Operation")
dataOperationFieldId = dataTableOperation.DataTableFieldCollection.Add("Id")
dataOperationFieldId.PrimaryKey = True
dataOperationFieldName = dataTableOperation.DataTableFieldCollection.Add("Name")
dataOperationFieldName.Type = VcDataTableFieldType.vcDataTableFieldStringType
dataOperationFieldTaskId =
dataTableOperation.DataTableFieldCollection.Add("TaskId")
dataOperationFieldTaskId.Type = VcDataTableFieldType.vcDataTableFieldIntegerType

'Node tables Task and Operations
dataOperationFieldTaskId.RelationshipFieldIndex =
VcGanttASP1.DetectFieldIndex("Task", "Id")
VcGanttASP1.DataTableCollection.Update()

```

**Example Code C#**

```

//Create table Task
VcDataTable dataTableTask = vcGanttASP1.DataTableCollection.Add("Task");
VcDataTableField dataTaskFieldId =
dataTableTask.DataTableFieldCollection.Add("Id");
dataTaskFieldId.PrimaryKey = true;
VcDataTableField dataTaskFieldName =
dataTableTask.DataTableFieldCollection.Add("Name");
dataTaskFieldName.Type = VcDataDefinitionFieldType.vcDefFieldStringType;

//Create table Operation
VcDataTable dataTableOperation =
vcGanttASP1.DataTableCollection.Add("Operation");
VcDataTableField dataOperationFieldId =
dataTableOperation.DataTableFieldCollection.Add("Id");
dataOperationFieldId.PrimaryKey = true;
VcDataTableField dataOperationFieldName =
dataTableOperation.DataTableFieldCollection.Add("Name");
dataOperationFieldName.Type = VcDataDefinitionFieldType.vcDefFieldStringType;
VcDataTableField dataOperationFieldTaskId =
dataTableOperation.DataTableFieldCollection.Add("TaskId");
dataOperationFieldTaskId.Type = VcDataDefinitionFieldType.vcDefFieldIntegerType;

//Node tables Task and Operation
dataOperationFieldTaskId.RelationshipFieldIndex =
vcGanttASP1.DetectFieldIndex("Task", "Id");
vcGanttASP1.DataTableCollection.Update();

```

## 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 **vcDataTableFieldInteger** the date format probably set will change to "".

	Data Type	Explanation
<b>Property value</b>	VcDataTableFieldType	Data type of the field, can contain 512 characters maximum <b>Default value:</b> vcDataTableFieldIntegerType
	<b>Possible Values:</b>	
	.vcDataFieldAlphanumericType 1	Data type <b>alphanumeric</b>
	.vcDataFieldDateTimeType 3	Data type <b>date</b>
	.vcDataTableFieldIntegerType 2	Data type <b>integer</b> (32 bits)

### Example Code VB.NET

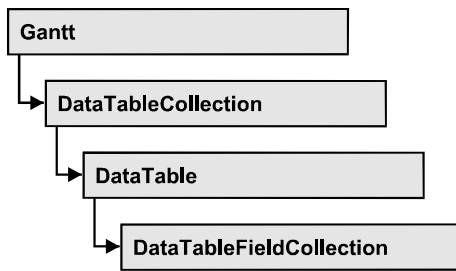
```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

VcGanttASP1.DataTableCollection.DataTableByName("Operation")
dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Start")
dataTableField.Type = VcDataTableFieldType.vcDataTableFieldDateTimeType
VcGanttASP1.DataTableCollection.Update()
```

### Example Code C#

```
VcDataTable dataTable =
vcGanttASP1.DataTableCollection.DataTableByName("Operation");
VcDataTableField dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Start");
dataTableField.Type = VcDataTableFieldType.vcDataTableFieldDateTimeType;
vcGanttASP1.DataTableCollection.Update();
```

## 6.23 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 **FirstDataTableField** and **NextDataTableField** allow to access data fields by iteration while by **DataTableFieldByName** and **DataTableFieldByIndex** single data fields can be accessed. **Add** and **Copy** represent basic administering methods.

### Properties

- Count

### Methods

- Add
- Copy
- DataTableFieldByIndex
- DataTableFieldByName
- FirstDataTableField
- GetEnumerator
- NextDataTableField

---

## Properties

### 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	System.Int32	Number of data table fields in the collection object

### Example Code VB.NET

```
Dim dataTable As VcDataTable

dataTable = VcGanttASP1.DataTableCollection.FirstDataTable()
MsgBox(dataTable.DataTableFieldCollection.Count.ToString())
```

### Example Code C#

```
VcDataTable dataTable = vcGanttASP1.DataTableCollection.FirstDataTable();
MessageBox.Show(dataTable.DataTableFieldCollection.Count.ToString());
```

---

## 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.

	Data Type	Explanation
<b>Parameter:</b>		
⇒ dataTableFieldName	System.String	Name of the data table field to be generated
<b>Return value</b>	VcDataTableField	Data table field generated

### Example Code VB.NET

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

dataTable = VcGanttASP1.DataTableCollection.FirstDataTable()
dataTableField = dataTable.DataTableFieldCollection.Add("Priority")
VcGanttASP1.DataTableCollection.Update()
```

### Example Code C#

```
VcDataTable dataTable = vcGanttASP1.DataTableCollection.FirstDataTable();
VcDataTableField dataTableField =
dataTable.DataTableFieldCollection.Add("Priority");
vcGanttASP1.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
<b>Parameter:</b>		
⇒ dataTableFieldName	System.String	Name of the data table field to be copied (source field)
⇒ newDataTableFieldName	System.String	Name of the data table field to be generated (target field)
<b>Return value</b>	VcDataTableField	Data table field generated

**Example Code VB.NET**

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

dataTable = VcGanttASP1.DataTableCollection.FirstDataTable()
dataTableField = dataTable.DataTableFieldCollection.Copy("Name", "NewName")
VcGanttASP1.DataTableCollection.Update()
```

**Example Code C#**

```
VcDataTable dataTable = vcGanttASP1.DataTableCollection.FirstDataTable();
VcDataTableField dataTableField =
dataTable.DataTableFieldCollection.Copy("Name", "NewName");
vcGanttASP1.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
<b>Parameter:</b>		
⇒ index	System.Int16	Index of the data table field
<b>Return value</b>	VcDataTableField	Data table field returned

**Example Code VB.NET**

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

dataTable = VcGanttASP1.DataTableCollection.FirstDataTable()
dataTableField = dataTable.DataTableFieldCollection.DataTableFieldByIndex(1)
MsgBox(dataTableField.Name)
```

**Example Code C#**

```
VcDataTable dataTable = vcGanttASP1.DataTableCollection.FirstDataTable();
VcDataTableField dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByIndex(1);
MessageBox.Show(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
<b>Parameter:</b> ⇒ dataTableFieldName	System.String	Name of the data table field
<b>Return value</b>	VcDataTableField	Data table field returned

### Example Code VB.NET

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

dataTable = VcGanttASP1.DataTableCollection.FirstDataTable()
dataTableField = dataTable.DataTableFieldCollection.DataTableFieldByName("Name")
dataTableField.Editable = False
VcGanttASP1.DataTableCollection.Update()
```

### Example Code C#

```
VcDataTable dataTable = vcGanttASP1.DataTableCollection.FirstDataTable();
VcDataTableField dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Name");
dataTableField.Editable = false;
vcGanttASP1.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
<b>Return value</b>	VcDataTableField	First data table field

### Example Code VB.NET

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

dataTable = VcGanttASP1.DataTableCollection.FirstDataTable()
dataTableField = dataTable.DataTableFieldCollection.FirstDataTableField()
```

**Example Code C#**

```
VcDataTable dataTable = vcGanttASP1.DataTableCollection.FirstDataTable();
VcDataTableField dataTableField =
dataTable.DataTableFieldCollection.FirstDataTableField();
```

**GetEnumerator****Method of VcDataTableFieldCollection**

This method returns an Enumerator object which supports the iteration by language specific elements. It is implied in the For...Each construct of Visual Basic and C#. This object allows to iterate over the data table fields included.

	Data Type	Explanation
<b>Return value</b>	VcObject	Enumerator object

**Example Code VB.NET**

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

dataTable = VcGanttASP1.DataTableCollection.FirstDataTable()
For Each dataTableField In dataTable.DataTableFieldCollection
    ListBox1.Items.Add(dataTableField.Name)
Next
```

**Example Code C#**

```
VcDataTable dataTable = vcGanttASP1.DataTableCollection.FirstDataTable();
foreach (VcDataTableField dataTableField in dataTable.DataTableFieldCollection)
    listBox1.Items.Add(dataTableField.Name);
```

**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
<b>Return value</b>	VcDataTable	Succeeding data table field

## 466 API Reference: VcDataTableFieldCollection

### Example Code VB.NET

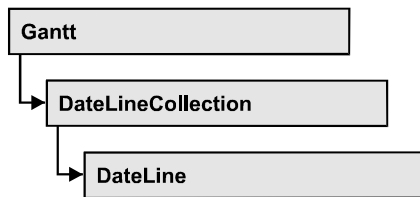
```
Dim dataTable As VcDataTable
Dim dataTableFieldCltn As VcDataTableFieldCollection
Dim dataTableField As VcDataTableField
Dim i As Integer

dataTable = VcGanttASP1.DataTableCollection.FirstDataTable()
dataTableFieldCltn = dataTable.DataTableFieldCollection
dataTableField = dataTableFieldCltn.FirstDataTableField
For i = 1 To dataTableFieldCltn.Count
    ListBox1.Items.Add(dataTableField.Name)
    dataTableField = dataTableFieldCltn.NextDataTableField()
Next
```

### Example Code C#

```
VcDataTable dataTable = vcGanttASP1.DataTableCollection.FirstDataTable();
VcDataTableFieldCollection dataTableFieldCltn =
dataTable.DataTableFieldCollection;
VcDataTableField dataTableField = dataTableFieldCltn.FirstDataTableField();
for (int i=0; i<dataTableFieldCltn.Count; i++)
{
    listBox1.Items.Add(dataTableField.Name);
    dataTableField = dataTableFieldCltn.NextDataTableField();
}
```

## 6.24 VcDateLine



An object of the type VcDateLine is a time-orientated vertical line in a Gantt diagram that marks a date.

### Properties

- AlwaysCurrentDate
- Date
- LineColor
- LineThickness
- LineType
- Name
- Priority
- Specification
- Text
- Visible

### 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 control. This property can be set in the **Specify Date Lines** dialog.

	Data Type	Explanation
Property value	System.Boolean	Property active/not active <b>Default value:</b> False

## 468 API Reference: VcDateLine

### Example Code VB.NET

```
Dim dateLine As VcDateLine
Dim dateLineTimer As Timer

dateLine = VcGantt1.DateLineCollection.DateLineByName("DateLine1")
If dateLine.AlwaysAtCurrentDate = True Then
    dateLineTimer.Enabled = True
End If
```

### Example Code C#

```
VcDateLine dateLine = vcGantt1.DateLineCollection.DateLineByName("DateLine1");
Timer dateLineTimer;

if (dateLine.AlwaysAtCurrentDate)
    dateLineTimer.Enabled = true;
```

## Date

### Property of VcDateLine

This property lets you set or retrieve 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	System.DateTime	Date {1.1.1970...31.12.2035} <b>Default value:</b> none or current date

### Example Code VB.NET

```
Dim dateLine As VcDateLine

dateLine = VcGantt1.DateLineCollection.DateLineByName("DateLine1")
dateLine.Date = "30.09.14 12:00:00"
```

### Example Code C#

```
VcDateLine dateLine = vcGantt1.DateLineCollection.DateLineByName("DateLine1");
dateLine.Date = Convert.ToDateTime("30.09.14 12:00:00");
```

## LineColor

### Property of VcDateLine

This property lets you set or retrieve the line color of a date line. This property also can be set in the **Edit Date Line** dialog.

	Data Type	Explanation
Property value	System.Drawing.Color	RGB color values {0...255},{0...255},{0...255} <b>Default value:</b> 255. Visual Basic: RGB (255, 0, 0)

**Example Code VB.NET**

```
Dim dateLine As VcDateLine

dateLine = VcGantt1.DateLineCollection.DateLineByName("DateLine1")
dateLine.LineColor = Color.Blue
```

**Example Code C#**

```
VcDateLine dateLine = vcGantt1.DateLineCollection.DateLineByName("DateLine1");
dateLine.LineColor = Color.LightSteelBlue;
```

## 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.

This property also can be set in the **Edit Date Line** dialog.

## 470 API Reference: VcDateLine

	Data Type	Explanation
<b>Property value</b>	System.Int16	Line thickness  LineType {1...4}: line thickness in pixels  LineType {5...1000}: line thickness in 1/100 mm <b>Default value:</b> As defined in the dialog

### Example Code VB.NET

```
Dim dateLine As VcDateLine

dateLine = VcGantt1.DateLineCollection.DateLineByName("DateLine1")
dateLine.LineType = VcLineType.vcSolid
dateLine.LineThickness = 3
```

### Example Code C#

```
VcDateLine dateLine = vcGantt1.DateLineCollection.DateLineByName("DateLine1");
dateLine.LineType = VcLineType.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
<b>Property value</b>	VcLineType	Line type <b>Default value:</b> vcSolid
	<b>Possible Values:</b>	
	.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 -----

**Example Code VB.NET**

```
Dim dateLine As VcDateLine

dateLine = VcGantt1.DateLineCollection.DateLineByName("DateLine1")
dateLine.LineType = VcLineType.vcSolid
```

**Example Code C#**

```
VcDateLine dateLine = vcGantt1.DateLineCollection.DateLineByName("DateLine1");
dateLine.LineType = VcLineType.vcSolid;
```

**Name****Read Only Property of VcDateLine**

This property lets you retrieve the name of a date line.

	Data Type	Explanation
Property value	System.String	Name

**Example Code VB.NET**

```
Dim datelineCltn As VcDateLineCollection
Dim dateline As VcDateLine

datelineCltn = VcGantt1.DateLineCollection
For Each dateline In datelineCltn
    ListBox1.Items.Add(dateline.Name)
Next
```

**Example Code C#**

```
VcDateLineCollection datelineCltn = vcGantt1.DateLineCollection;

foreach (VcDateLine dateline in datelineCltn)
{
    ListBox.Items.Add(dateline.Name);
}
```



## 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	System.Int16	Priority value <b>Default value:</b> 0

### Example Code VB.NET

```
Dim dateLine As VcDateLine
```

```
dateLine = VcGantt1.DateLineCollection.DateLineByName("DateLine1")
dateLine.Priority = 10
```

### Example Code C#

```
VcDateLine dateLine = vcGantt1.DateLineCollection.DateLineByName("DateLine1");
dateLine.Priority = 10;
```

## 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	System.String	Specification of the date line

## 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
Property value	System.String	Annotation

#### Example Code VB.NET

```
Dim dateLine As VcDateLine

dateLine = VcGantt1.DateLineCollection.DateLineByName("DateLine1")
dateLine.Text = "Stichtag"
```

#### Example Code C#

```
VcDateLine dateLine = vcGantt1.DateLineCollection.DateLineByName("DateLine1");
dateLine.Text = "Stichtag";
```

## 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	System.Boolean	Date line visible/invisible <b>Default value:</b> True

#### Example Code VB.NET

```
Dim dateLine As VcDateLine

dateLine = VcGantt1.DateLineCollection.DateLineByName("DateLine1")
dateLine.Visible = False
```

#### Example Code C#

```
VcDateLine dateLine = vcGantt1.DateLineCollection.DateLineByName("DateLine1");
dateLine.Visible = false;
```

---

## Methods

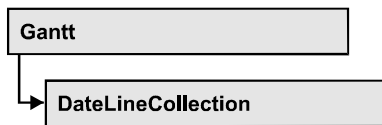
### PutInOrderAfter

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
<b>Parameter:</b> ↔ refName	System.String	Name of the date line, after which the current date line shall be placed.
<b>Return value</b>	Void	

## 6.25 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 **DateLineByName** 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

- Count

### Methods

- DateLineByIndex
- DateLineByName
- FirstDateLine
- GetEnumerator
- NextDateLine

---

## Properties

### 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
<b>Parameter:</b> ⇒ Rückgabewert	System.Int32	Number of data lines
<b>Property value</b>	System.Int32	Number of date lines

### Example Code VB.NET

```
Dim numberOfDateLine As Integer  
  
numberOfDateLine = VcGantt1.DateLineCollection.Count
```

### Example Code C#

```
int numberOfDateLines = vcGantt1.DateLineCollection.Count;
```

---

## Methods

### DateLineByIndex

Method of VcDateLineCollection

This method lets you access a data line by its index. If a date line does not exist at the index specified, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
<b>Parameter:</b> ⇒ index	System.Int16	Index of the date line
<b>Return value</b>	VcDateLine	Date line object returned

### Example Code VB.NET

```
Dim dateLineCltn As VcDateLineCollection  
Dim dateLine As VcDateLine  
  
dateLineCltn = VcGantt1.DateLineCollection  
dateLine = dateLineCltn.DateLineByIndex(0)  
MsgBox(dateLine.Name)
```

### Example Code C#

```
VcDateLineCollection dateLineCltn = vcGantt1.DateLineCollection;  
VcDateLine dateLine = dateLineCltn.DateLineByIndex(0);  
MessageBox.Show(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
<b>Parameter:</b> ⇒ dateLineName	System.String	Name of the date line
<b>Return value</b>	VcDateLine	Date line

**Example Code VB.NET**

```
Dim dateLineCltn As VcDateLineCollection
Dim dateLine As VcDateLine

dateLineCltn = VcGantt1.DateLineCollection
dateLine = dateLineCltn.DateLineByName("DateLineOne")
MsgBox(dateLine.Name)
```

**Example Code C#**

```
VcDateLineCollection dateLineCltn = vcGantt1.DateLineCollection;
VcDateLine dateLine = dateLineCltn.DateLineByName("DateLineOne");
MessageBox.Show(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 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
<b>Return value</b>	VcDateLine	First date line

**Example Code VB.NET**

```
Dim datelineCltn As VcDateLineCollection
Dim dateline As VcDateLine

datelineCltn = VcGantt1.DateLineCollection
dateline = datelineCltn.FirstDateLine

While Not dateline Is Nothing
    ListBox1.Items.Add(dateline.Name)
    dateline = datelineCltn.NextDateLine
End While
```

## 478 API Reference: VcDateLineCollection

### Example Code C#

```
VcDateLineCollection datelineCltn = vcGantt1.DateLineCollection;
VcDateLine dateline = datelineCltn.FirstDateLine();

while (dateline != null)
{
    ListBox.Items.Add(dateline.Name);
    dateline = datelineCltn.NextDateLine();
}
```

## GetEnumerator

### Method of VcDateLineCollection

This method returns an Enumerator object which supports the iteration by language specific elements. It is implied in the For...Each construct of Visual Basic and C#. This object allows to iterate over the date line objects included.

	Data Type	Explanation
Return value	VcObject	Reference object

### Example Code VB.NET

```
Dim dateLineCltn As VcDateLineCollection
Dim dateLine As VcDateLine

dateLineCltn = VcGantt1.DateLineCollection
For Each dateLine In dateLineCltn
    ListBox1.Items.Add(dateLine.Name)
Next
```

### Example Code C#

```
VcDateLineCollection dateLineCltn = vcGantt1.DateLineCollection;
foreach (VcDateLine dateLine in dateLineCltn)
    listBox1.Items.Add(dateLine.Name);
```

## 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 VB.NET**

```
Dim datelineCltn As VcDateLineCollection
Dim dateline As VcDateLine

datelineCltn = VcGantt1.DateLineCollection
dateline = datelineCltn.FirstDateLine

While Not dateline Is Nothing
    ListBox1.Items.Add(dateline.Name)
    dateline = datelineCltn.NextDateLine
End While
```

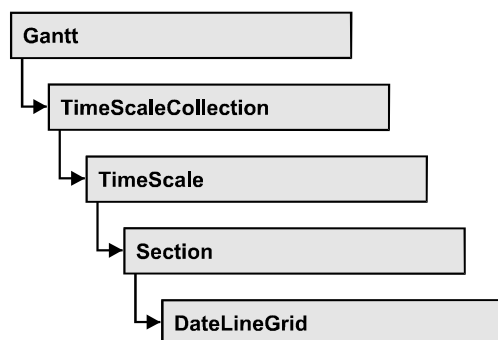
**Example Code C#**

```
VcDateLineCollection datelineCltn = vcGantt1.DateLineCollection;
VcDateLine dateline = datelineCltn.FirstDateLine();

while (dateline != null)
{
    ListBox.Items.Add(dateline.Name);
    dateline = datelineCltn.NextDateLine();
}
```



## 6.26 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
- Period
- Priority
- TurningAnnotationEnabled
- Unit
- UseReferenceDate
- Visible

## 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**.

	Data Type	Explanation
Property value	System.Boolean	Date line grid positioned (False) / not positioned on reference date (False) <b>Default value:</b> False

### 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	System.Boolean	Date line grid annotations positioned at bottom (True) / not at bottom (False) <b>Default value:</b> False

### 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	System.Boolean	Date line grid annotations positioned in the center (True) / not in the center (False) <b>Default value:</b> False

## 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	System.Boolean	Date line grid annotations positioned at top (True) / not at top (False) <b>Default value:</b> False

## FormatName

Property of VcDateLineGrid

This property lets you set or retrieve the name of the line format of this date line grid.

	Data Type	Explanation
Property value	System.String	Name of the line format

## HorAlignment

Property of VcDateLineGrid

This property lets you set or retrieve the horizontal alignment of the line annotations.

	Data Type	Explanation
Property value	VcHorizontalAlignment	Horizontal alignment
	<b>Possible Values:</b> .vcHorCenterAligned - 1	horizontally centered
	.vcLeftAligned -3	left aligned

.vcRightAligned -2	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	System.Drawing.Color	RGB color values {0...255},{0...255},{0...255} <b>Default value:</b> 255. Visual Basic: RGB (255, 0, 0)

### Example Code VB.NET

```
Dim dateLineGrid As VcDateLineGrid

dateLineGrid = VcGantt1.TimeScaleCollection.Active.Section(0).DateLineGrid(0)
dateLineGrid.LineColor = Color.Blue
```

### Example Code C#

```
VcDateLineGrid dateLineGrid =
vcGantt1.TimeScaleCollection.Active.get_Section(0).get_DateLineGrid(0);
dateLineGrid.LineColor = Color.LightSteelBlue;
```

## LineColorDataFieldIndex

Read Only 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	System.Int32	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
Property value	System.String	Name of the color map

## 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** dialog.

	Data Type	Explanation
Property value	System.Int16	Line thickness LineType {1...4}: line thickness in pixels LineType {5...1000}: line thickness in 1/100 mm <b>Default value:</b> As defined in the dialog

### Example Code VB.NET

```
Dim dateLineGrid As VcDateLineGrid

dateLineGrid = VcGantt1.TimeScaleCollection.Active.Section(0).DateLineGrid(0)
dateLineGrid.LineThickness = 2
```



### Example Code VB.NET

```
Dim dateLineGrid As VcDateLineGrid

dateLineGrid = VcGantt1.TimeScaleCollection.Active.Section(0).DateLineGrid(0)
dateLineGrid.LineType = VcLineType.vcSolid
```

### Example Code C#

```
VcDateLineGrid dateLineGrid =
vcGantt1.TimeScaleCollection.Active.get_Section(0).get_DateLineGrid(0);
dateLineGrid.LineType = VcLineType.vcSolid;
```

## 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	System.Int32	Period value <b>Default value: 1</b>

### Example Code VB.NET

```
Dim dateLineGrid As VcDateLineGrid

dateLineGrid = VcGantt1.TimeScaleCollection.Active.Section(0).DateLineGrid(0)
dateLineGrid.Unit = VcGridUnit.vcGridUnitDay
dateLineGrid.Period = 1
```

### Example Code C#

```
VcDateLineGrid dateLineGrid =
vcGantt1.TimeScaleCollection.Active.get_Section(0).get_DateLineGrid(0);
dateLineGrid.Unit = VcGridUnit.vcGridUnitDay;
dateLineGrid.Period = 1;
```

## Priority

### Property of VcDateLineGrid

This property lets you set or retrieve 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	System.Int32	Priority value  {-1000...+1000} <b>Default value:</b> -20

**Example Code VB.NET**

```
Dim dateLineGrid As VcDateLineGrid
```

```
dateLineGrid = VcGantt1.TimeScaleCollection.Active.Section(0).DateLineGrid(0)
dateLineGrid.Priority = 10
```

**Example Code C#**

```
VcDateLineGrid dateLineGrid =
vcGantt1.TimeScaleCollection.Active.get_Section(0).get_DateLineGrid(0);
dateLineGrid.Period = 10;
```

## 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	System.Boolean	The annotations can be turned (True) / were already turned (False) <b>Default value:</b> True

## 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	VcGridUnit	Time unit <b>Default value:</b> vcGridUnitWeek
	<b>Possible Values:</b> .vcGridUnitDay 5 .vcGridUnitHour 6 .vcGridUnitMinute 7 .vcGridUnitMonth 3 .vcGridUnitQuarter 2 .vcGridUnitSecond 8	Grid unit <b>day</b> Grid unit <b>hour</b> Grid unit <b>minute</b> Grid unit <b>month</b> Grid unit <b>quarter</b> Grid unit <b>second</b>



## 488 API Reference: VcDateLineGrid

.vcGridUnitWeek 4	Grid unit <b>week</b>
.vcGridUnitYear 1	Grid unit <b>year</b>

### Example Code VB.NET

```
Dim dateLineGrid As VcDateLineGrid

dateLineGrid = VcGantt1.TimeScaleCollection.Active.Section(0).DateLineGrid(0)
dateLineGrid.Unit = VcGridUnit.vcGridUnitDay
```

### Example Code C#

```
VcDateLineGrid dateLineGrid =
vcGantt1.TimeScaleCollection.Active.get_Section(0).get_DateLineGrid(0);
dateLineGrid.Period = 1;
dateLineGrid.Unit = VcGridUnit.vcGridUnitDay;
```

## UseReferenceDate

### Read Only Property of VcDateLineGrid

This property lets you set or retrieve whether the date line grid uses a reference date.

	Data Type	Explanation
Property value	System.Boolean	Date line grid uses (True)/does not use (False) reference date

## Visible

### Property of VcDateLineGrid

This property lets you set or retrieve whether a date line grid is visible.

	Data Type	Explanation
Property value	System.Boolean	Date line grid visible/invisible <b>Default value:</b> True

### Example Code VB.NET

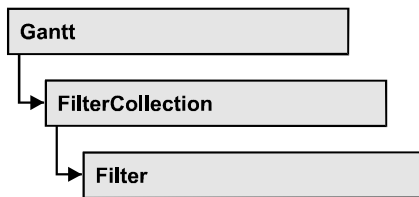
```
Dim dateLineGrid As VcDateLineGrid

dateLineGrid = VcGantt1.TimeScaleCollection.Active.Section(0).DateLineGrid(0)
dateLineGrid.Visible = True
```

### Example Code C#

```
VcDateLineGrid dateLineGrid =
vcGantt1.TimeScaleCollection.Active.get_Section(0).get_DateLineGrid(0);
dateLineGrid.Visible = true;
```

## 6.27 VcFilter



An object of the type VcFilter contains subconditions (VcFilterSubCondition), p.e. 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

- DataDefinitionTable
- DatesWithHourAndMinute
- Name
- Specification
- StringsCaseSensitive
- SubCondition
- SubConditionCount

### Methods

- AddSubCondition
- CopySubCondition
- GetEnumerator
- IsValid
- RemoveSubCondition

## Properties

### 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 conditions.

	Data Type	Explanation
Property value	VcDataTableType  <b>Possible Values:</b> .vcMaindata 0 .vcRelations 1	Type of data definition table  table type <b>vcMaindata</b> (for nodes) table type <b>vcRelations</b> (for links)

### DatesWithHourAndMinute

Property of VcFilter

This property lets you set or retrieve whether the comparison of conditions that contain dates takes into account hours and minutes. This setting can only be modified if there is at least one subcondition that compares dates. Otherwise the property value is always False.

	Data Type	Explanation
Property value	System.Boolean	Hours and minutes are compared (True)/ not compared (False)

### Name

Property of VcFilter

This property lets you set or retrieve the name of the filter.

	Data Type	Explanation
Property value	System.String	Name of the filter

**Example Code VB.NET**

```
Dim filterCltn As VcFilterCollection
Dim filter As VcFilter

filterCltn = VcGanttASP1.FilterCollection

For Each filter In filterCltn
    ListBox1.Items.Add(filter.Name)
Next
```

**Example Code C#**

```
VcFilterCollection filterCltn = vcGanttASP1.FilterCollection;

foreach (VcFilter filter in filterCltn)
{
    ListBox.Items.Add(filter.Name);
}
```

**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 databases. This allows for persistency. A specification can be used to create a filter by the method **VcFilterCollection.AddBySpecification**.

	Data Type	Explanation
Property value	System.String	Specification of the filter

**StringsCaseSensitive****Property of VcFilter**

This property lets you enquire/set whether subconditions that contain strings are case-sensitive.

	Data Type	Explanation
Property value	System.Boolean	Case-sensitive (True)/not case-sensitive (False)

**SubCondition****Read Only Property of VcFilter**

This property lets you access a VcFilterSubCondition object by its index.

The property SubCondition is an Indexed Property, which in C# is addressed by the method get\_SubCondition (index).

	Data Type	Explanation
<b>Parameter:</b> ⇒ index	System.Int16	Index of the filter subcondition  {0 ... VcFilter.SubConditionCount-1}
<b>Property value</b>	VcFilterSubCondition	Filter subcondition object

## SubConditionCount

**Read Only Property of VcFilter**

This property lets you enquire the number of filter subconditions.

	Data Type	Explanation
<b>Property value</b>	System.Int16	Number of filter subconditions

---

## 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
<b>Parameter:</b> ⇨ atIndex	System.Int16	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)}
<b>Return value</b>	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
<b>Parameter:</b> ⇨ fromIndex  ⇨ atIndex	System.Int16  System.Int16	Index of the filter subcondition to be copied  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)}
<b>Return value</b>	VcFilterSubCondition	Filter subcondition object

## GetEnumerator

**Method of VcFilter**

This method returns an Enumerator object which supports the iteration by language specific elements. It is implied in the For...Each construct of Visual Basic and C#. This object allows to iterate over the condition objects included.

	Data Type	Explanation
<b>Return value</b>	VcObject	Reference object

## 494 API Reference: VcFilter

### Example Code VB.NET

```
Dim filter As VcFilter
Dim filterCond As VcFilterSubCondition

filter = VcGanttASPl.FilterCollection.FirstFilter

For Each filterCond In filter
    Debug.Write(filterCond.Index)
Next
```

### Example Code C#

```
VcFilter filter = vcGanttASPl.FilterCollection.FirstFilter();
foreach(VcFilterSubCondition filterCond in filter)
{
    Console.Write(filterCond.Index);
}
```

## 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 subconditions will remain valid.

	Data Type	Explanation
<b>Return value</b>	System.Boolean	Filter subconditions correct (True)/ not correct (False)

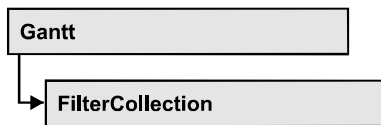
## RemoveSubCondition

### Method of VcFilter

This method lets you delete a filter subcondition by its index.

	Data Type	Explanation
<b>Parameter:</b> ⇒ index	System.Int16	Index of the filter subcondition to be removed

## 6.28 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

- Count
- MarkedNodesFilter

### Methods

- Add
- AddBySpecification
- Copy
- FilterByIndex
- FilterByName
- FirstFilter
- GetEnumerator
- NextFilter
- Remove

---

## Properties

### Count

**Read Only Property of VcFilterCollection**

This property lets you retrieve the number of filters in the filter collection.

	Data Type	Explanation
Property value	System.Int32	Number of filters



### Example Code VB.NET

```
Dim filterCltn As VcFilterCollection
Dim numberOfFilters As Integer

filterCltn = VcGanttASP1.FilterCollection
numberOfFilters = filterCltn.Count
```

### Example Code C#

```
VcFilterCollection filterCltn = vcGanttASP1.FilterCollection;
int 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 VB.NET

```
VcGanttASP1.ActiveNodeFilter = VcGanttASP1.FilterCollection.MarkedNodesFilter
```

### Example Code C#

```
vcGanttASP1.ActiveNodeFilter = vcGanttASP1.FilterCollection.MarkedNodesFilter;
```

---

## Methods

### Add

#### Method of VcFilterCollection

By this method you can create a filter as a member of the FilterCollection. If the name was not 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
<b>Parameter:</b> ⇒ newName	System.String	Filter name
<b>Return value</b>	VcFilter	New filter object

**Example Code VB.NET**

```
newFilter = VcGanttASP1.FilterCollection.Add("foo")
```

**Example Code C#**

```
newFilter = vcGanttASP1.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
<b>Parameter:</b> ⇒ filterSpecification	System.String	Filter specification
<b>Return value</b>	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
<b>Parameter:</b> ⇒ fromName	System.String	Name of the filter to be copied
⇒ newName	System.String	Name of the new filter
<b>Return value</b>	VcFilter	Filter object

## FilterByIndex

Method of VcFilterCollection

This method lets you access a filter by its index. If a filter does not exist at the index specified, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
<b>Parameter:</b> ⇒ index	System.Int16	Index of the filter
<b>Return value</b>	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
<b>Parameter:</b> ⇒ filterName	System.String	Filter name
<b>Return value</b>	VcFilter	Filter

### Example Code VB.NET

```
Dim filterCltn As VcFilterCollection
Dim filter As VcFilter

filterCltn = VcGanttASP1.FilterCollection
filter = filterCltn.FilterByName("Department A")
```

### Example Code C#

```
VcFilterCollection filterCltn = vcGanttASP1.FilterCollection;
VcFilter 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 VB.NET**

```
Dim filterCltn As VcFilterCollection
Dim filter As VcFilter

filterCltn = VcGanttASP1.FilterCollection
filter = filtercltn.FirstFilter
```

**Example Code C#**

```
VcFilterCollection filterCltn = vcGanttASP1.FilterCollection;
VcFilter filter = filterCltn.FirstFilter();
```

## GetEnumerator

**Method of VcFilterCollection**

This method returns an Enumerator object which supports the iteration by language specific elements. It is implied in the For...Each construct of Visual Basic and C#. This object allows to iterate over the filter objects included.

	Data Type	Explanation
Return value	VcObject	Reference object

**Example Code VB.NET**

```
Dim filter As VcFilter
Dim filterCond As VcFilterSubCondition

filter = VcGanttASP1.FilterCollection.FirstFilter

For Each filterCond In filter
    Debug.Write(filterCond.FilterName)
Next
```

**Example Code C#**

```
VcFilter filter = vcGanttASP1.FilterCollection.FirstFilter();
foreach(VcFilterSubCondition filterCond in filter)
{
    Console.Write(filterCond.FilterName);
}
```

## 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
<b>Return value</b>	VcFilter	Next filter

### Example Code VB.NET

```
Dim filterCltn As VcFilterCollection
Dim filter As VcFilter

filterCltn = VcGanttASP1.FilterCollection
filter = filterCltn.FirstFilter

While Not filter Is Nothing
    ListBox1.Items.Add(filter.Name)
    filter = filterCltn.NextFilter
End While
```

### Example Code C#

```
VcFilterCollection filterCltn = vcGanttASP1.FilterCollection;
VcFilter filter = filterCltn.FirstFilter();

while (filter != null)
{
    ListBox.Items.Add(filter.Name);
    filter = filterCltn.NextFilter();
}
```

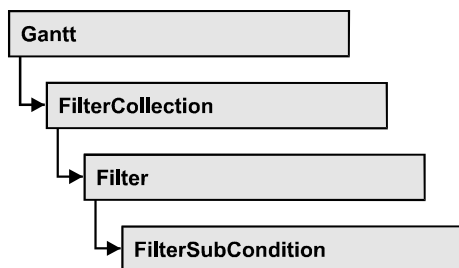
## Remove

### 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
<b>Parameter:</b> ⇒ name	System.String	Filter name
<b>Return value</b>	System.Boolean	Filter deleted (True)/not deleted (False)

## 6.29 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

- `GetEnumerator`
- `IsValid`

---

## Properties

### ComparisonValueAsString

Property of `VcFilterSubCondition`

This property lets you set or retrieve the comparison value. This string must have the below format:

- String: needs to be included by double quotation marks. Example in VB: `""Berlin""`; Example in C/C++: `"Berlin"`

## 502 API Reference: VcFilterSubCondition

- Date: included by # signs. Example: "#21/03/2005 12:00#". 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 mentioned 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	System.String	Comparison value

## ConnectionOperator

### Property of VcFilterSubCondition

This property lets you set or retrieve the operator that connects the subsequent subcondition. Among the operators **vcAnd** is stronger than **vcOr**.

	Data Type	Explanation
Property value	VcConnectionOperator  <b>Possible Values:</b> .vcAnd 1 .vcInvalidConnOp 0 .vcOr 2	Operator for the connection holding the below subcondition  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 be set in the **Edit filter** dialog.

	Data Type	Explanation
Property value	System.Int32	Index of the data field to be compared

**FilterName****Read Only Property of VcFilterSubCondition**

This property lets you retrieve the name of the filter to which this subcondition belongs.

	Data Type	Explanation
Property value	System.String	Name of the filter

**Index****Read Only Property of VcFilterSubCondition**

This property lets you retrieve the index of this subcondition in the corresponding filter.



## 504 API Reference: VcFilterSubCondition

	Data Type	Explanation
Property value	System.Int16	Index of the subcondition in the filter

## Operator

### Property of VcFilterSubCondition

This property lets you set or retrieve 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	VcOperator	Comparison operator
	<b>Possible Values:</b>	
	.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

### GetEnumerator

Method of VcFilterSubCondition

This method returns an Enumerator object which supports the iteration by language specific elements. It is implied in the For...Each construct of Visual Basic and C#. This object allows to iterate over the fields included in the filter subcondition object.

	Data Type	Explanation
Return value	VcObject	Reference object

### IsValid

Method of VcFilterSubCondition

This property checks whether the filter subcondition is correct.

	Data Type	Explanation
Return value	System.Boolean	Filter subcondition correct (True)/ not correct (False)

## 6.30 VcGantt

A VcGantt object is the VARCHART XGantt server control. It can be customized by a number of properties and methods to meet your demands.

### Properties

- ActiveNodeFilter
- AjaxExtensionsEnabled
- Arrangement
- BorderArea
- BoxCollection
- BoxFormatCollection
- CalendarCollection
- CalendarGridCollection
- CalendarProfileCollection
- DataDefinition
- DataTableCollection
- DateLineCollection
- DateOutputFormat
- DiagramAlternatingRowBackgroundColor
- DiagramBackgroundColor
- DiagramHistogramHeightRatio
- Enabled
- EndDateForAutomaticScheduling
- ExtendedDataTablesEnabled
- FilePath
- FilterCollection
- FontAntiAliasingEnabled
- GroupCollection
- GroupingDataFieldIndex
- GroupingModificationsAllowed
- GroupOptimizationOnInteractionsEnabled
- GroupSortingDataFieldIndex
- GroupSortingOrder
- HierarchyDataFieldIndex
- HistogramCollection
- HistogramSeparationLineColor
- InitialRowCount
- KeepingNodesTogetherDataFieldIndex

- LayerCollection
- LayersWithNonWorkInterval
- LeftTable
- LeftTableDiagramWidthRatio
- LineFormatCollection
- LinkAppearanceCollection
- LinkCollection
- LinkPredecessorDataFieldIndex
- LinksDataTableName
- LinkSuccessorDataFieldIndex
- LinkTypeDataFieldIndex
- MapCollection
- MinimumRowHeight
- MovingLayersAsNodeWithShiftKeyAllowed
- NodeCalendarNameDataFieldIndex
- NodeCollection
- NodeDurationDataFieldIndex
- NodeEndDateDataFieldIndex
- NodeRowNumberDataFieldIndex
- NodesDataTableName
- NodeSortingDataFieldIndex
- NodeSortingOrder
- NodeStartDateDataFieldIndex
- NodesUseCalendars
- NumericScaleCollection
- OverlapLayerEnabled
- OverlapLayerName
- Printer
- RequestImage
- ResourceScheduler2
- RoundedLinkSlantsEnabled
- RowHeightReductionEnabled
- RowMargins
- Scheduler
- ScrollEventsEnabled
- StartDateForAutomaticScheduling
- SubRowMargins
- SummaryBarsVisible
- TextEntrySupplyingEventEnabled
- TimeScaleCollection

- TimeScaleEnd
- TimeScaleStart
- TimeUnit
- TimeUnitsPerStep
- ToolTipChangeDuration
- ToolTipDuration
- ToolTipPointerDuration
- ToolTipShowAfterClick
- ToolTipTextSupplyingEventEnabled
- VerticalNodeMovementAllowed
- VerticalNodeMovementViaTableAllowed
- ZoomFactor
- ZoomingPerMouseWheelAllowed

### Methods

- ConvertDistance
- DeleteLinkRecord
- DeleteNodeRecord
- DetectDataTableFieldName
- DetectDataTableName
- DetectFieldIndex
- EndLoading
- ExportGraphicsToFileEx
- FitChartIntoView
- FitHistogramsIntoView
- FitRangeIntoView
- GetCurrentViewDates
- GetDate
- GetLicenseInformation
- GetLinkByID
- GetLinkByNodeIDs
- GetNodeByID
- GetViewComponentSize
- GroupNodes
- IdentifyField
- IdentifyLayerAt
- IdentifyObjectAt
- ImportConfiguration
- InsertLinkRecord
- InsertNodeRecord

- Load
- OptimizeTimeScaleStartEnd
- PrintEx
- PrintToFile
- RecalculateAllStructureCodes
- Reset
- SaveAsEx
- ScrollToDate
- ScrollToGroupLine
- ScrollToNode
- ScrollToNodeLine
- SortGroups
- SortNodes
- SuspendUpdate
- UpdateLinkRecord
- UpdateNodeRecord
- UpdateRowNumberFields
- Zoom

## **Events**

- VcBoxLeftClicking
- VcComponentScrolled
- VcComponentScrolling
- VcCurveLeftClicking
- VcDataRecordModified
- VcDataRecordModifying
- VcDataRecordNotFound
- VcDiagramHorizontalScrolled
- VcDiagramHorizontalScrolling
- VcDiagramLeftClicking
- VcErrorOccurring
- VcFieldSelecting
- VcGroupLeftClicking
- VcGroupModified
- VcGroupModifying
- VcGroupsMarked
- VcGroupsMarking
- VcHistogramLeftClicking
- VcHistogramsHeightChanged
- VcHistogramsHeightChanging

- VcLinksLeftClicking
- VcNodeLeftClicking
- VcNodeModified
- VcNodeModifiedEx
- VcNodeModifying
- VcNodesMarked
- VcNodesMarking
- VcNumericScaleLeftClicking
- VcObjectDrawing
- VcObjectDrawn
- VcResourceSchedulingProgressing
- VcResourceSchedulingWarning
- VcTableCaptionLeftClicking
- VcTableColumnWidthChanging
- VcTableWidthChanging
- VcTextEntrySupplying
- VcTimeScaleLeftClicking
- VcTimeScaleSectionRescaled
- VcToolTipTextSupplying

---

## 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 <b>Default value:</b> Nothing

#### Example Code VB.NET

```
VcGantt1.ActiveNodeFilter = VcGantt1.FilterCollection.FilterByName("Milestone")
```

#### Example Code C#

```
vcGantt1.ActiveNodeFilter = vcGantt1.FilterCollection.FilterByName("Milestone");
```

## AjaxExtensionsEnabled

Property of VcGantt

This property lets you enable or disable the AJAX extensions. This property can also be set on the **General** property page.

	Data Type	Explanation
<b>Property value</b>	System.Boolean	Ajax extensions enabled (True) or disabled (False) <b>Default value:</b> True

### Example Code VB.NET

```
VcGantt1.AjaxExtensionsEnabled = True
```

### Example Code C#

```
vcGantt1.AjaxExtensionsEnabled = true;
```

## 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
<b>Property value</b>	VcArrangementType	Arrangement of activities groupwise or hierarchical <b>Default value:</b> vcArrangementTypeGroupwise
	<b>Possible Values:</b> .vcArrangementTypeGroupwise 1 .vcArrangementTypeHierarchical 2	Groupwise Arrangement of activities Hierarchical Arrangement of activities

### Example Code VB.NET

```
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)
```



## 512 API Reference: VcGantt

### Example Code C#

```
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);
```

## 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 VB.NET

```
Dim borderArea As VcBorderArea
borderArea = VcGantt1.BorderArea
```

### Example Code C#

```
VcBorderArea 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 VB.NET

```
Dim boxCltn As VcBoxCollection
boxCltn = VcGantt1.BoxCollection
```

### Example Code C#

```
VcBoxCollection boxCltn = vcGantt1.BoxCollection;
```

## BoxFormatCollection

Read Only Property of VcGantt

This property gives access to the BoxFormatCollection object that contains all box formats available to the table.

	Data Type	Explanation
Property value	VcBoxFormatCollection	BoxFormatCollection object

### Example Code VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormat As VcBoxFormat

boxFormatCltn = VcGantt1.BoxFormatCollection
For Each boxFormat In boxFormatCltn
    ListBox1.Items.Add(boxFormat.Name)
Next
```

### Example Code C#

```
VcBoxFormatCollection boxFormatCltn = vcGantt1.BoxFormatCollection;
foreach (VcBoxFormat boxFormat in boxFormatCltn)
{
    listBox1.Items.Add(boxFormat.Name);
}
```

## CalendarCollection

Read Only Property of VcGantt

This property gives access to the calendar collection object and thus to the calendars used.

	Data Type	Explanation
Property value	VcCalendarCollection	CalendarCollection object

### Example Code VB.NET

```
Dim calendarCltn As VcCalendarCollection
calendarcltn = VcGantt1.CalendarCollection
```

### Example Code C#

```
VcCalendarCollection calendarCltn = vcGantt1.CalendarCollection;
```

## CalendarGridCollection

Read Only Property of VcGantt

This property gives access to the calendar grid collection object and thus to the calendar grids used.

## 514 API Reference: VcGantt

	Data Type	Explanation
Property value	VcCalendarGridCollection	CalendarGridCollection object

### Example Code VB.NET

```
Dim calendarGridCltn As VcCalendarGridCollection  
calendarGridCltn = VcGantt1.CalendarGridCollection
```

### Example Code C#

```
VcCalendarGridCollection calendarGridCltn = vcGantt1.CalendarGridCollection;
```

## 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

## 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 VB.NET

```
Dim dataDefinition As VcDataDefinition  
dataDefinition = VcGantt1.DataDefinition
```

### Example Code C#

```
VcDataDefinition dataDefinition = vcGantt1.DataDefinition;
```

## DataTableCollection

Property of VcGantt

This property gives access to the data table collection that contains the existing data tables.

	Data Type	Explanation
Property value	VcDataTableCollection	Data table collection object returned

### Example Code VB.NET

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable

dataTableCltn = VcGanttASP1.DataTableCollection
For Each dataTable In dataTableCltn
    ListBox1.Items.Add(dataTable.Name)
Next
```

### Example Code C#

```
VcDataTableCollection dataTableCltn = vcGanttASP1.DataTableCollection;
foreach(VcDataTable dataTable in dataTableCltn)
    listBox1.Items.Add(dataTable.Name);
```

## 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 VB.NET

```
Dim dateLineCltn As VcDateLineCollection
dateLineCltn = VcGantt1.DateLineCollection
```

### Example Code C#

```
VcDateLineCollection dateLineCltn = vcGantt1.DateLineCollection;
```

## DateOutputFormat

Property of VcGantt

This property lets you set or retrieve 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 **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.

This property can also be set on the **General** property page.

	Data Type	Explanation
Property value	System.String	Date  {DMYhms:;}

#### Example Code VB.NET

```
VcGantt1.DateOutputFormat = "DD.MM.YY"
```

#### Example Code C#

```
vcGantt1.DateOutputFormat = "DD.MM.YY";
```

## DiagramAlternatingRowBackgroundColor

Read Only 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 **DiagramBackgroundColor**. This property also can be set on the **Layout** property page.

	Data Type	Explanation
Property value	System.Drawing.Color	RGB color values  {0...255},{0...255},{0...255} <b>Default value:</b> SystemDrawing.Color.White

#### Example Code VB.NET

```
VcGantt1.DiagramAlternatingRowBackgroundColor = System.Drawing.Color.Blue
```

#### Example Code C#

```
vcGantt1.DiagramAlternatingRowBackgroundColor = System.Drawing.Color.Blue;
```

## DiagramBackgroundColor

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
<b>Property value</b>	System.Drawing.Color	RGB color values {0...255},{0...255},{0...255}) <b>Default value:</b> SystemDrawing.Color.White

### Example Code VB.NET

```
VcGantt1.DiagramBackgroundColor = System.Drawing.Color.Blue
```

### Example Code C#

```
vcGantt1.DiagramBackgroundColor = System.Drawing.Color.Blue;
```

## 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
<b>Property value</b>	Integer {-1, 0, 1, ..., 1000}	Ratio between diagram height and histogram height

### Example Code VB.NET

```
Dim ratio As Integer
ratio = VcGantt1.DiagramHistogramHeightRatio
```

### Example Code C#

```
int ratio = vcGantt1.DiagramHistogramHeightRatio;
```

## Enabled

Property of VcGantt

This property lets you disable the VARCHART XGantt control so that it will not react to mouse commands.

	Data Type	Explanation
<b>Property value</b>	System.Boolean	VARCHART control enabled/disabled

**Example Code VB.NET**

```
VcGantt1.Enabled = False
```

**Example Code C#**

```
vcGantt1.Enabled = false;
```

## EndDateForAutomaticScheduling

**Property of VcGantt**

This property lets you set or retrieve the end value for autoscheduling of the current project (**Schedule** property page).

	Data Type	Explanation
<b>Property value</b>	System.DateTime	End date

## ExtendedDataTablesEnabled

**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
<b>Property value</b>	System.Boolean	<b>true:</b> only two data tables (Maindata and Relations) <b>false:</b> up to 99 data tables <b>Default value:</b> false

**Example Code VB.NET**

```
VcGantt1.ExtendedDataTablesEnabled = True
```

**Example Code C#**

```
vcGantt1.ExtendedDataTablesEnabled = 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 XGantt control.

This property should be set when the application is started during the initializing procedure of the VARCHART XGantt 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	System.String	File path Default value: ""

### Example Code VB.NET

```
Dim exeName As String
Dim exeDir As String

exeName = System.Environment.GetCommandLineArgs(0)
exeDir = System.IO.Path.GetDirectoryName(exeName)
VcGantt1.FilePath = exeDir + "\Bitmaps"
```

### Example Code C#

```
String exeName = Environment.GetCommandLineArgs()[0];
vcGantt1.FilePath = System.IO.Path.GetDirectoryName(exeName) + @"..\Bitmaps";
```

## 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 VB.NET

```
Dim filterCltn As VcFilterCollection
filterCltn = VcGantt1.FilterCollection
```

### Example Code C#

```
VcFilterCollection filterCltn = vcGantt1.FilterCollection;
```

## FontAntiAliasingEnabled

**Read Only Property of VcGantt**

This property lets you set or retrieve whether or not font characters should be anti-aliased. Some fonts, especially non-Latin ones may lose clarity, so the property should be set to **False** in those cases.

This property also can be set on the **General** property page.

	Data Type	Explanation
Property value	System.Boolean	Characters will/will not be anti-aliased

## GroupCollection

**Read Only Property of VcGantt**

If a grouping is specified, this property gives access to the GroupCollection object that contains all groups.

	Data Type	Explanation
Property value	VcGroupCollection	GroupCollection object

### Example Code VB.NET

```
Dim groupCltn As VcGroupCollection
groupCltn = VcGantt1.GroupCollection
```

### Example Code C#

```
VcGroupCollection groupCltn = vcGantt1.GroupCollection;
```

## GroupingDataFieldIndex

**Property of VcGantt**

This property lets you set or retrieve the field in the data definition table that is to be used as criterion for the grouping on a certain level. The groups by default will be sorted in the order of reading the first activity of the group. The sorting order can be modified by the property **GroupSortingDataFieldIndex**.

This property also can be set in the **Grouping** property page.

The property **GroupingDataFieldIndex** is an Indexed Property, which in C# is addressed by the methods `set_GroupingDataFieldIndex (groupingLevel, pvn)` and `get_GroupingDataFieldIndex (groupingLevel)`.

	Data Type	Explanation
<b>Parameter:</b> ⇒ groupingLevel	System.Int16	Grouping level (starting by 0)
<b>Property value</b>	System.Int32	Field ID of the data definition table

### Example Code VB.NET

```
Dim definitionTable As VcDataDefinitionTable
definitionTable =
VcGanttASP1.DataDefinition.DefinitionTable(VcDataTableType.vcMaindata)
VcGanttASP1.GroupingDataFieldIndex(0) =
definitionTable.DataDefinitionFieldByName("Code 1").ID
VcGanttASP1.GroupNodes(True)
```

### Example Code C#

```
VcDataDefinitionTable definitionTable =
vcGanttASP1.DataDefinition.get_DefinitionTable(VcDataTableType.vcMaindata);
vcGanttASP1.set_GroupingDataFieldIndex(0, "Code 1");
vcGanttASP1.GroupNodes(true);
```

## GroupingModificationsAllowed

### Property of VcGantt

This property lets you specify whether 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 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.

The property **GroupingModificationsAllowed** is an Indexed Property, which in C# is addressed by the methods `set_GroupingModificationsAllowed (groupingLevel, pvn)` and `get_GroupingModificationsAllowed (groupingLevel)`.

	Data Type	Explanation
<b>Parameter:</b> ⇒ groupingLevel	System.Int16	Grouping level
<b>Property value</b>	System.Boolean	Modifications allowed (True)/ not allowed (False)

### Example Code VB.NET

```
VcGantt1.GroupingModificationsAllowed(0) = False
```

**Example Code C#**

```
vcGantt1.set_GroupingModificationsAllowed(0, false);
```

**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
<b>Property value</b>	System.Boolean	The <b>Optimized</b> re-arrangement of nodes will (True) / will not (False) be performed on interaction

**GroupSortingDataFieldIndex**

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 **GroupSortingDataFieldIndex**, the groups will be sorted in ascending or descending alphabetical order by this field. This property also can be set in the **Grouping** dialog.

The property GroupSortingDataFieldIndex is an Indexed Property, which in C# is addressed by the methods set\_GroupSortingDataFieldIndex (groupingLevel, pvn) and get\_GroupSortingDataFieldIndex (groupingLevel).

	Data Type	Explanation
<b>Parameter:</b> ⇒ groupingLevel	System.Int16	Grouping level
<b>Property value</b>	System.Int32	Field index of the data definition table

### Example Code VB.NET

```
VcGanttASP1.GroupSortingDataFieldIndex(0) = 12
VcGanttASP1.GroupSortingOrder(0) = VcNodeSortingOrder.vcDescending
VcGanttASP1.SortGroups()
```

### Example Code C#

```
vcGantt1.set_GroupSortingDataFieldIndex(0, 12);
vcGantt1.set_GroupSortingOrder(0, VcNodeSortingOrder.vcDescending);
vcGantt1.SortGroups();
```

## GroupSortingOrder

### Property of VcGantt

This property lets you specify the sorting order of groups (ascending or descending). By the property **GroupSortingDataFieldIndex** you can specify the field by that the groups are sorted. This property also can be set in the **Grouping** dialog.

	Data Type	Explanation
<b>Property value</b>	VcNodeSortingOrder	Ascending or descending order <b>Default value:</b> vcAscending
	<b>Possible Values:</b> .vcAscending 1 .vcDescending 2	ascending order Descending order

### Example Code VB.NET

```
VcGantt1.GroupSortingDataFieldIndex(0) = 12
VcGantt1.GroupSortingOrder(0) = VcNodeSortingOrder.vcAscending
VcGantt1.SortGroups()
```

### Example Code C#

```
vcGantt1.set_GroupSortingDataFieldIndex(0, 12);
vcGantt1.set_GroupSortingOrder(0, VcNodeSortingOrder.vcAscending);
vcGantt1.SortGroups();
```

## 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	System.Int32	Index of the data field which defines the hierarchical order of activities

**Example Code VB.NET**

```
VcGantt1.HierarchyDataFieldIndex = VcGantt1.DetectFieldIndex("Maindata",
"Hierarchy")
VcGantt1.Arrangement = VcArrangementType.vcArrangementTypeHierarchical
VcGantt1.GroupNodes(True)
```

**Example Code C#**

```
vcGantt1.HierarchyDataFieldIndex = vcGantt1.DetectFieldIndex("Maindata",
"Hierarchy");
vcGantt1.Arrangement = VcArrangementType.vcArrangementTypeHierarchical;
vcGantt1.GroupNodes(true);
```

## HistogramCollection

**Read Only Property of VcGantt**

This property gives access to the HistogramCollection object and thus to the histograms used.

	Data Type	Explanation
Property value	VcHistogramCollection	HistogramCollection object

**Example Code VB.NET**

```
Dim histogramCltn As VcHistogramCollection
histogramCltn = VcGantt1.HistogramCollection
```

**Example Code C#**

```
VcHistogramCollection histogramCltn = vcGantt1.HistogramCollection;
```

## 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	System.Drawing.Color	Color value ({0...255},{0...255},{0...255})

## InitialRowCount

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 **Layout** property page.

	Data Type	Explanation
Property value	System.Int32	Number of node rows at the program start

## KeepingNodesTogetherDataFieldIndex

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 and the grouping options **In one line** and **Nodes optimized** have been activated (**Grouping** dialog). Then you can select a data field that should be used for the separation. Then all nodes of a group with the same value in this data field will be displayed in one line, even if they will 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	System.Int16	Number of the field that should be used for the separation of nodes in groups

### Example Code VB.NET

```
VcGantt1.KeepingNodesTogetherDataFieldIndex = 3
```

### Example Code C#

```
vcGantt1.KeepingNodesTogetherDataFieldIndex = 3;
```

## LayerCollection

Read Only Property of VcGantt

This property gives access to the LayerCollection object that contains all defined layers.

	Data Type	Explanation
Property value	VcLayerCollection	LayerCollection object

**Example Code VB.NET**

```
Dim layerCltn As VcLayerCollection
layerCltn = VcGantt1.LayerCollection
```

**Example Code C#**

```
VcLayerCollection layerCltn = vcGantt1.LayerCollection;
```

## LayersWithNonWorkInterval

**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:** **NodesUseCalendars** has to be set to True.

	Data Type	Explanation
Property value	System.Boolean	Show workfree intervals (true)/do not show workfree intervals (false).

**Example Code VB.NET**

```
VcGantt1.LayersWithNonWorkInterval = True
```

**Example Code C#**

```
vcGantt1.LayersWithNonWorkInterval = true;
```

## LeftTable

**Read Only Property of VcGantt**

This property gives access to the Table object in order to access the formats used in order to modify its table columns and their headings.

	Data Type	Explanation
Property value	VcTable	Table

**Example Code VB.NET**

```
Dim table As VcTable
table = VcGantt1.LeftTable
```



### Example Code C#

```
VcTable table = vcGantt1.LeftTable;
```

## LeftTableDiagramWidthRatio

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	System.Int16	Width ratio {-1, 1...100}

### Example Code VB.NET

```
VcGantt1.LeftTableDiagramWidthRatio = 40
```

### Example Code C#

```
vcGantt1.LeftTableDiagramWidthRatio = 40;
```

## LineFormatCollection

Read Only Property of VcGantt

This property gives access to the LineFormatCollection object that contains all line formats available to the table.

	Data Type	Explanation
Property value	VcLineFormatCollection	LineFormatCollection object

### Example Code VB.NET

```
Dim lineFormatCltn As VcLineFormatCollection  
Dim lineFormat As VcLineFormat  
  
lineFormatCltn = VcGantt1.LineFormatCollection  
For Each lineFormat In lineFormatCltn  
    ListBox1.Items.Add(lineFormat.Name)  
Next
```

### Example Code C#

```
VcLineFormatCollection lineFormatCltn = vcGantt1.LineFormatCollection;  
foreach (VcLineFormat lineFormat in lineFormatCltn)  
{  
    listBox1.Items.Add(lineFormat.Name);  
}
```

## LinkAppearanceCollection

Read Only Property of VcGantt

This property gives access to the LinkAppearanceCollection object that contains all link appearance objects defined.

	Data Type	Explanation
Property value	VcLinkAppearanceCollection	LinkAppearanceCollection object

### Example Code VB.NET

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
linkAppearanceCltn = VcGanttASP1.LinkAppearanceCollection
```

### Example Code C#

```
VcLinkAppearanceCollection linkAppearanceCltn =
vcGanttASP1.LinkAppearanceCollection;
```

## LinkCollection

Read Only Property of VcGantt

This property gives access to the LinkCollection object that contains all links defined.

	Data Type	Explanation
Property value	VcLinkCollection	LinkCollection object

### Example Code VB.NET

```
Dim linkCltn As VcLinkCollection
linkCltn = VcGantt1.LinkCollection
```

### Example Code C#

```
VcLinkCollection linkCltn = vcGantt1.LinkCollection;
```

## LinkPredecessorDataFieldIndex

Property of VcGantt

This property lets you set or retrieve the data field which holds the identification of the predecessor 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.

The property `LinkPredecessorDataFieldIndex` is an Indexed Property, which in C# is addressed by the methods `set_LinkPredecessorDataFieldIndex` (`identifierIndex`, `pvn`) and `get_LinkPredecessorDataFieldIndex` (`identifierIndex`).

	Data Type	Explanation
<b>Parameter:</b> ⇒ <code>identifierIndex</code>	System.Int16	Index of predecessor node {0...2}
<b>Property value</b>	System.Int32	Field index of the data definition table

### Example Code VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecord As VcDataRecord

'create Link DataTable
dataTable = VcGantt1.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
dataTable = VcGantt1.DataTableCollection.DataTableByName("LinkDataTable")
dataRecord = dataTable.DataRecordCollection.Add("1;1;2;")
VcGantt1.EndLoading()
```

### Example Code C#

```
VcDataTable dataTable;
VcDataRecord dataRecord;

//create Link DataTable
dataTable = vcGantt1.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.set_LinkPredecessorDataFieldIndex(0,
vcGantt1.DetectFieldIndex("LinkDataTable", "Id"));
vcGantt1.set_LinkSuccessorDataFieldIndex(0,
vcGantt1.DetectFieldIndex("LinkDataTable", "Id"));

//Load Data
dataTable = vcGantt1.DataTableCollection.DataTableByName("LinkDataTable");
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 provides the data fields for links.

To make links appear on the screen, also the properties **LinkSuccessorDataFieldIndex** and **LinkPredecessor** need to be set.

This property can also be set on the **Links** property page.

	Data Type	Explanation
<b>Property value</b>	System.String	Name of the data table which provides the fields for the links

### Example Code VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecord As VcDataRecord

'create Link DataTable
dataTable = VcGantt1.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
dataTable = VcGantt1.DataTableCollection.DataTableByName("LinkDataTable")
dataRecord = dataTable.DataRecordCollection.Add("1;1;2;")
VcGantt1.EndLoading()
```

**Example Code C#**

```

VcDataTable dataTable;
VcDataRecord dataRecord;

//create Link DataTable
dataTable = vcGantt1.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.set_LinkPredecessorDataFieldIndex(0,
vcGantt1.DetectFieldIndex("LinkDataTable", "Id"));
vcGantt1.set_LinkSuccessorDataFieldIndex(0,
vcGantt1.DetectFieldIndex("LinkDataTable", "Id"));

//Load Data
dataTable = vcGantt1.DataTableCollection.DataTableByName("LinkDataTable");
dataRecord = dataTable.DataRecordCollection.Add("1;1;2;");
vcGantt1.EndLoading();

```

**LinkSuccessorDataFieldIndex****Property of VcGantt**

This property lets you set or retrieve the data field which holds the successor node of a link. This is only possible as long as no data has been loaded.

This property can also be set on the **Links** property page.

The property LinkSuccessorDataFieldIndex is an Indexed Property, which in C# is addressed by the methods set\_LinkSuccessorDataFieldIndex (identifierIndex, pvn) and get\_LinkSuccessorDataFieldIndex (identifierIndex).

	Data Type	Explanation
<b>Parameter:</b>		
⇒ identifierIndex	System.Int16	Index of successor node {0..2}
<b>Property value</b>	System.Int32	Fieldindex of the data definition table

**Example Code VB.NET**

```

Dim dataTable As VcDataTable
Dim dataRecord As VcDataRecord

'create Link DataTable
dataTable = VcGantt1.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
dataTable = VcGantt1.DataTableCollection.DataTableByName("LinkDataTable")
dataRecord = dataTable.DataRecordCollection.Add("1;1;2;")
VcGantt1.EndLoading()

```

**Example Code C#**

```

VcDataTable dataTable;
VcDataRecord dataRecord;

//create Link DataTable
dataTable = vcGantt1.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.set_LinkPredecessorDataFieldIndex(0,
vcGantt1.DetectFieldIndex("LinkDataTable", "Id"));
vcGantt1.set_LinkSuccessorDataFieldIndex(0,
vcGantt1.DetectFieldIndex("LinkDataTable", "Id"));

//Load Data
dataTable = vcGantt1.DataTableCollection.DataTableByName("LinkDataTable");
dataRecord = dataTable.DataRecordCollection.Add("1;1;2;");
vcGantt1.EndLoading();

```

**LinkTypeDataFieldIndex****Property of VcGantt**

This property lets you set or retrieve the name of the data field which contains the link type. This is only possible as long as no data was loaded.

This property can also be set on the **Links** property page.

	Data Type	Explanation
<b>Property value</b>	System.Int32	Index of the data field which contains the link type

**Example Code VB.NET**

```
Dim dataTable As VcDataTable

'create Link DataTable
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()
```

**Example Code C#**

```
VcDataTable dataTable;

//create Link DataTable
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 MapCollection object that contains a defined number of maps. The number of maps is defined by the method **VcMapCollection.SelectMaps**.

	Data Type	Explanation
<b>Property value</b>	VcMapCollection	MapCollection object

**Example Code VB.NET**

```
Dim mapCltn As VcMapCollection
mapCltn = VcGantt1.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)
```

**Example Code C#**

```
VcMapCollection mapCltn = vcGantt1.MapCollection;
mapCltn.SelectMaps(VcMapType.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	System.Int32	Minimal row height

#### Example Code VB.NET

```
VcGantt1.MinimumRowHeight = 100
```

#### Example Code C#

```
vcGantt1.MinimumRowHeight = 100;
```

## MovingLayersAsNodeWithShiftKeyAllowed

Read Only Property of VcGantt

This property lets you specify/enquire 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	System.Boolean	Moving of all layers of a node with shift enabled/disabled <b>Default value:</b> true

#### Example Code VB.NET

```
VcGanttASP1.MoveLayersAsNodeWithShiftKey = False
```

## NodeCalendarNameDataFieldIndex

Property of VcGantt

This property lets you set or retrieve the index of the data field which holds the name of the calendar if you wish to use an individual calendar for a node. Setting this property is only possible if no data was loaded yet.

This property also can be set on the **Nodes** property page.



	Data Type	Explanation
Property value	System.Int32	Index of the data field which contains the name of the node calendar

## NodeCollection

Read Only Property of VcGantt

This property gives access to the NodeCollection object, that 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 VB.NET

```
Dim nodeCltn As VcNodeCollection
nodeCltn = VcGantt1.NodeCollection
nodeCltn.SelectNodes(VcSelectionType.vcAll)
```

### Example Code C#

```
VcNodeCollection nodeCltn = vcGantt1.NodeCollection;
nodeCltn.SelectNodes(VcSelectionType.vcAll);
```

## NodeDurationDataFieldIndex

Property of VcGantt

This property lets you set or retrieve 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	System.Int32	Index of the data field which holds the duration of an interactively created node

## NodeEndDateDataFieldIndex

Property of VcGantt

This property lets you set or retrieve the index of the data field which holds 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	System.Int32	Index of the data field which holds the end date of an interactively created node

## NodeRowNumberDataFieldIndex

Property of VcGantt

This property lets you set or retrieve the index of the data field which holds the row number of an activity. Setting this property is only possible if no data was loaded yet. This property also can be set on the **Nodes** property page.

	Data Type	Explanation
Property value	System.Int32	Index of the data field which holds the row number of an activity

### Example Code VB.NET

```
Private Sub Form1_Load(ByVal sender As Object, ByVal e As System.EventArgs)
    Handles MyBase.Load
        VcGantt1.NodeRowNumberDataFieldIndex =
            VcGantt1.DetectFieldIndex("NodeDataTable", "SortNumber")
            'Load data
            LoadData()

            VcGantt1.UpdateRowNumberFields()
            VcGantt1.SaveAsEx("C:\ProjectData.txt", VcEncoding.vcUnicodeEncoding)
End Sub
```

### Example Code C#

```
private void Form1_Load(object sender, System.EventArgs e)
{
    vcGantt1.NodeRowNumberDataFieldIndex =
        VcGantt1.DetectFieldIndex("NodeDataTable", "SortNumber");

    // Load data
    loadData();

    vcGantt1.UpdateRowNumberFields();
    vcGantt1.SaveAsEx(@"C:\ProjectData.txt", VcEncoding.vcUnicodeEncoding);
}
```

## NodesDataTableName

Property of VcGantt

This property lets you set or retrieve the name of the data table which provides the data fields for the nodes. This property also can be set on the **Nodes** property page.

	Data Type	Explanation
Property value	System.String	Name of the data table which provides the fields for the nodes

### Example Code VB.NET

```
Dim dataTable As VcDataTable
    Dim dataRecord As VcDataRecord

    'create Node DataTable
    dataTable = vcGantt1.DataTableCollection.Add("NodeDataTable")
    VcGantt1.NodesDataTableName = dataTable.Name
    dataTable.DataTableFieldCollection.Add("Id").PrimaryKey = True
```

### Example Code C#

```
VcDataTable dataTable;
VcDataRecord dataRecord;

//create Node DataTable
dataTable = vcGantt1.DataTableCollection.Add("NodeDataTable");
vcGantt1.NodesDataTableName = dataTable.Name;
dataTable.DataTableFieldCollection.Add("Id").PrimaryKey = true;
//Load Data
dataTable = vcGantt1.DataTableCollection.DataTableByName("NodeDataTable");
dataRecord = dataTable.DataRecordCollection.Add("1;Node One;");
dataRecord = dataTable.DataRecordCollection.Add("2;Node Two;");
vcGantt1.EndLoading();
```

## NodeSortingDataFieldIndex

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 **NodeSortingOrder** property. Sorting is to be triggered by the method **SortNodes**.

This property also can be set in the **Grouping** dialog.

The property **NodeSortingDataFieldIndex** is an Indexed Property, which in C# can be addressed by the methods `set_NodeSortingDataFieldIndex (sortLevel, pvn)` and `get_NodeSortingDataFieldIndex (sortLevel)`.

	Data Type	Explanation
<b>Parameter:</b> ⇒ sortLevel	System.Int16	Sorting level {0...2}
<b>Property value</b>	System.Int32	Field index of the data definition table

**Example Code VB.NET**

```
VcGantt1.NodeSortingDataFieldIndex(0) = 11
VcGantt1.NodeSortingOrder(0) = VcNodeSortingOrder.vcDescending
VcGantt1.SortNodes()
```

**Example Code C#**

```
vcGantt1.set_NodeSortingDataFieldIndex(0,11);
vcGantt1.set_NodeSortingOrder(0, VcNodeSortingOrder.vcDescending);
vcGantt1.SortNodes();
```

## NodeSortingOrder

**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
<b>Property value</b>	Integer  <b>Possible Values:</b> .vcAscending 1 .vcDescending 2	Ascending or descending order <b>Default value:</b> vcAscending  ascending order Descending order

**Example Code VB.NET**

```
VcGantt1.NodeSortingDataFieldIndex(0) = 11
VcGantt1.NodeSortingOrder(0) = VcNodeSortingOrder.vcDescending
VcGantt1.SortNodes()
```

**Example Code C#**

```
vcGantt1.set_NodeSortingDataFieldIndex(0,11);
vcGantt1.set_NodeSortingOrder(0, VcNodeSortingOrder.vcDescending);
vcGantt1.SortNodes();
```

## NodeStartDateDataFieldIndex

Property of VcGantt

This property lets you set or retrieve the index of the data field which holds the start date of an interactively created activity. Setting this property is only possible if no data was loaded yet. This property also can be set on the **Nodes** property page.

	Data Type	Explanation
Property value	System.Int32	Index of the data field which holds the start date of an interactively created node

## NodesUseCalendars

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	System.Boolean	Property active/not active

### Example Code VB.NET

```
VcGantt1.NodesUseCalendars = False
```

### Example Code C#

```
vcGantt1.NodesUseCalendars = false;
```

## NumericScaleCollection

Read Only Property of VcGantt

This property gives access to the NumericScaleCollection object, that contains all numeric scales available.

	Data Type	Explanation
Property value	VcNumericScaleCollection	NumericScaleCollection object

**Example Code VB.NET**

```
Dim histogramCltn As VcHistogramCollection
Dim numericScaleCltn As VcNumericScaleCollection
histogramCltn = VcGantt1.HistogramCollection
numericScaleCltn = histogram.FirstHistogram.NumericScaleCollection
```

**Example Code C#**

```
VcHistogramCollection histogramCltn = vcGantt1.HistogramCollection;
VcNumericScaleCollection numericScaleCltn =
histogramCltn.FirstHistogram().NumericScaleCollection;
```

## 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
<b>Property value</b>	System.Boolean	Overlap layer on (True) / off (False) <b>Default value:</b> False

## 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
<b>Property value</b>	System.String	Name of the overlap layer <b>Default value:</b> " "

## 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 current printer.

	Data Type	Explanation
Property value	VcPrinter	Printer object

**Example Code VB.NET**

```
Dim printerZoomfactor As Integer
Dim printerCuttingMarks As String
printerZoomfactor = VcGantt1.Printer.ZoomFactor
printerCuttingMarks = VcGantt1.Printer.CuttingMarks
```

**Example Code C#**

```
int printerZoomfactor = vcGantt1.Printer.ZoomFactor;
bool printerCuttingMarks = vcGantt1.Printer.CuttingMarks;
```

## RequestImage

**Property of VcGantt**

This method is static and only for internal use. In the response object of the page passed, it returns the present look of the VARCHART XGantt control as an image stream to the client. The method is invoked by the ASP.NET page **ImageServer.aspx** which by obligation has to be present on a server using VARCHART XGantt ASP.NET. (A copy of the file **ImageServer.aspx** you can find in the installation folder of VARCHART XGantt ASP.NET.)

	Data Type	Explanation
<b>Parameter:</b>		
⇒ page	System.Web.UI.Page	Page in the web where the control resides.
⇒ sessionStateKey	System.String	Name of the session variable, by which the control is kept on the server.
⇒ width	System.String	Width of the control (unit: pixels)
⇒ height	System.String	Height of the control (unit: pixels)
<b>Property value</b>	Void	

## ResourceScheduler2

**Read Only Property of VcGantt**

This property gives access to the ResourceScheduler2 object for resource scheduling.

	Data Type	Explanation
Property value	VcResourceScheduler2	ResourceScheduler2 object

**Example Code VB.NET**

```
VcGantt1.ResourceScheduler2.TaskDataTableName = "Task"
VcGantt1.ResourceScheduler2.TaskDueDateFieldIndex = 1
VcGantt1.ResourceScheduler2.TaskReleaseDateFieldIndex = 2
'...
VcGantt1.ResourceScheduler2.Process()
```

**Example Code C#**

```
vcGantt1.ResourceScheduler2.TaskDataTableName = "Task";
vcGantt1.ResourceScheduler2.TaskDueDateFieldIndex = 1;
vcGantt1.ResourceScheduler2.TaskReleaseDateFieldIndex = 2;
//...
vcGantt1.ResourceScheduler2.Process();
```

## RoundedLinkSlantsEnabled

**Property of VcGantt**

This property lets you set or retrieve whether the slants of links of the routing type **vcLRTOrthogonalDistinguishable** are to be displayed as quarter circles instead of straight 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 <b>Default value:</b> false

**Example Code VB.NET**

```
VcGanttASP1.RoundedLinkSlantsEnabled = True
```

**Example Code C#**

```
vcGanttASP1.RoundedLinkSlants.Enabled = true;
```

## RowHeightReductionEnabled

**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.



## 544 API Reference: VcGantt

If the 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 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 on the **General** property page.

	Data Type	Explanation
Property value	System.Boolean	Reduction of row height allowed (true)/not allowed (false)

### Example Code VB.NET

```
VcGanttASP1.RowHeightReductionEnabled = True
```

### Example Code C#

```
vcGanttASP1.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	System.Int32	Distance (in 1/100 mm) between the upper/ lower node margins and the upper/lower margins of the node rows

### Example Code VB.NET

```
VcGantt1.RowMargins = 100
```

### Example Code C#

```
vcGantt1.RowMargins = 100
```

## 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 **VcComponentScrolled**, **VcComponentScrolling**, **VcDiagramHorizontalScrolled** and **VcDiagramHorizontalScrolling**. This feature can also be set on the **General** property page.

	Data Type	Explanation
Property value	System.Boolean	Scroll events enabled (True) or disabled (False) <b>Default value:</b> False

### Example Code VB.NET

```
VcGantt1.ScrollEventsEnabled = True
```

### Example Code C#

```
vcGantt1.ScrollEventsEnabled = true;
```

## StartDateForAutomaticScheduling

Property of VcGantt

This property lets you set or retrieve the start value for autoscheduling of the current project (**Schedule** property page).

	Data Type	Explanation
Property value	System.DateTime	Start date

## SubRowMargins

Read Only 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	System.Int32	width between sub rows by 1/100 mm  {0...200} <b>Default value:</b> 50

**Example Code VB.NET**

```
VcGantt1.SubRowMargins = 100
```

## SummaryBarsVisible

Property of VcGantt

This property lets you set or retrieve whether summary bars are visible or not.

The property SummaryBarsVisible is an Indexed Property, which in C# can be addressed by the methods set\_SummaryBarsVisible (groupingLevel, pvn) and get\_SummaryBarsVisible (groupingLevel).

	Data Type	Explanation
Parameter: ⇒ groupingLevel	System.Int16	(not for hierarchy) grouping level (GroupingLevel = -1: reading: all levels, writing: at least one level )
Property value	System.Boolean	Summary bars visible (True)/ invisible (False)

**Example Code VB.NET**

```
VcGantt1.SummaryBarsVisible(-1) = True
```

**Example Code C#**

```
vcGantt1.set_SummaryBarsVisible(-1, true);
```

## TextEntrySupplyingEventEnabled

Property of VcGantt

This property lets you activate the **VcTextEntrySupplying** 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	System.Boolean	Property active/not active

**Example Code VB.NET**

```
VcGantt1.TextEntrySupplyingEventEnabled = True
```

**Example Code C#**

```
vcGantt1.TextEntrySupplyingEventEnabled = false;
```

## TimeScaleCollection

**Read Only Property of VcGantt**

This property gives access to the TimescaleCollection object and thus to the time scales available.

	Data Type	Explanation
Property value	VcTimeScaleCollection	TimeScaleCollection object

**Example Code VB.NET**

```
Dim timeScaleCltn As VcTimeScaleCollection
timeScaleCltn = VcGantt1.TimeScaleCollection
```

**Example Code C#**

```
VcTimeScaleCollection timeScaleCltn = vcGantt1.TimeScaleCollection;
```

## TimeScaleEnd

**Property of VcGantt**

This property lets you set or retrieve the end of the timescale. When setting, 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 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.

**Note:** The end date is not included. If you specify **TimeScaleEnd** = "31.12.02" for example, the last day displayed will be the 30.12.02.

	Data Type	Explanation
Property value	System.DateTime	End date of the time scale  {1.1.1980...31.12.2035}

**Example Code VB.NET**

```
'Timescale from 01.10.2014 to 30.11.2014
VcGantt1.TimeScaleEnd = "01.12.2014"
VcGantt1.TimeScaleStart = "01.10.2014"
VcGantt1.TimeScaleEnd = "01.12.2014"
```

**Example Code C#**

```
//Timescale from 01.10.2014 to 30.11.2014
vcGantt1.TimeScaleEnd = Convert.ToDateTime("01.12.14");
vcGantt1.TimeScaleStart = Convert.ToDateTime("01.10.14");
vcGantt1.TimeScaleEnd = Convert.ToDateTime("01.12.14");
```

## TimeScaleStart

**Property of VcGantt**

This property lets you set or retrieve the start of the timescale. 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	System.DateTime	Start date of the time scale  {1.1.1980...31.12.2035}

**Example Code VB.NET**

```
'Timescale from 01.10.2014 to 30.11.2014
VcGantt1.TimeScaleEnd = "01.12.2014"
VcGantt1.TimeScaleStart = "01.10.2014"
VcGantt1.TimeScaleEnd = "01.12.2014"
```

**Example Code C#**

```
//Timescale from 01.10.2014 to 30.11.2014
vcGantt1.TimeScaleEnd = Convert.ToDateTime("01.12.14");
vcGantt1.TimeScaleStart = Convert.ToDateTime("01.10.14");
vcGantt1.TimeScaleEnd = Convert.ToDateTime("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 also set on the **General** property page.

**Note:** If you want to change the time unit, you should do this before reading data because modifications set later will not be effective.

	Data Type	Explanation
<b>Property value</b>	VcTimeUnit	Time unit <b>Default value:</b> vcDay
	<b>Possible Values:</b> .vcDay 5 .vcHour 6 .vcMinute 7 .vcSecond 8	Time unit <b>day</b> Time unit <b>hour</b> Time unit <b>minute</b> Time unit <b>second</b>

### Example Code VB.NET

```
Dim timeUnit As VcTimeUnit
timeUnit = VcGantt1.TimeUnit
```

### Example Code C#

```
VcTimeUnit 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
<b>Property value</b>	System.Int16	Number of time units per step <b>Default value:</b> 1

### Example Code VB.NET

```
VcGantt1.TimeUnitsPerStep = 4
```

## 550 API Reference: VcGantt

### Example Code C#

```
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 (for Windows XP), please set it to -1.

	Data Type	Explanation
Property value	System.Int32	Duration in milliseconds. Maximum value: 32767 msec <b>Default value: -1</b>

### Example Code VB.NET

```
VcGantt1.ToolTipChangeDuration = 1000
```

### Example Code C#

```
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	System.Int32	Duration in milliseconds. Maximum value: 32767 msec <b>Default value: -1</b>

### Example Code VB.NET

```
VcGantt1.ToolTipDuration = 1000
```

### Example Code C#

```
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 default value of 480 msec (for Windows XP), please set it to -1.

	Data Type	Explanation
Property value	System.Int32	Duration in milliseconds <b>Default value:</b> -1

### Example Code VB.NET

```
VcGantt1.ToolTipPointerDuration = 1000
```

### Example Code C#

```
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	System.Boolean	Tool tip window disappears (false) or remains (true) <b>Default value:</b> False

### Example Code VB.NET

```
VcGantt1.ToolTipShowAfterClick = True
```

### Example Code C#

```
vcGantt1.ToolTipShowAfterClick = true;
```

## ToolTipTextSupplyingEventEnabled

Property of VcGantt

This property lets you activate/deactivate the event **VcToolTipTextSupplying**. This property also can be set on the **General** property page. The event **VcToolTipTextSupplying** lets you edit tooltip texts.



## 552 API Reference: VcGantt

	Data Type	Explanation
<b>Parameter:</b> ⇒ Rückgabewert	System.Boolean	Property active/not active
<b>Property value</b>	System.Boolean	Scroll event enabled (True) or disabled (False) <b>Default value:</b> False

### Example Code VB.NET

```
VcGantt1.ToolTipTextSupplyingEventEnabled = True
```

### Example Code C#

```
vcGantt1.ToolTipTextSupplyingEventEnabled = true;
```

## VerticalNodeMovementAllowed

Property of VcGantt

Returns/Sets 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
<b>Property value</b>	System.Boolean	Vertical node movement in diagram enabled/disabled <b>Default value:</b> false

## VerticalNodeMovementViaTableAllowed

Property of VcGantt

Returns/Sets 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
<b>Property value</b>	System.Boolean	Vertical node movement in table enabled/disabled <b>Default value:</b> false

## 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 cause inexactnesses.

Please see also the property **FitChartIntoView** and the method **Zoom()** of VcGantt.

	Data Type	Explanation
Property value	System.Int16	absolute zoom factor (%)

### Example Code VB.NET

```
VcGantt1.ZoomFactor = 150
```

### Example Code C#

```
vcGantt1.ZoomFactor = 150;
```

## ZoomingPerMouseWheelAllowed

Property of VcGantt

This property lets you set or retrieve whether zooming by mouse wheel should be allowed to the user. This property also can be set on the **General** property page.

	Data Type	Explanation
Property value	System.Boolean	Zooming allowed (true)/not allowed (False) <b>Default value:</b> False

### Example Code VB.NET

```
Dim h As VcHistogram  
h.FitRangeIntoView(0,10,1)
```

### Example Code C#

```
VcHistogram h;  
h.FitRangeIntoView(0,10,1);
```

## Methods

### 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
<b>Parameter:</b>		
⇒ conversionType	VcDistanceConversionType	Conversion type
⇒ value	System.Int32	Number of source units (that are to be converted)
<b>Return value</b>	System.Int32	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 **Administrative Data Tables** dialog.

	Data Type	Explanation
<b>Parameter:</b>		
⇒ linkRecordContent	System.Object	Content of the link record
<b>Return value</b>	System.Boolean	Link record was/was not deleted successfully.

#### Example Code VB.NET

```
VcGantt1.DeleteLinkRecord("A100;A105;;")
```

#### Example Code C#

```
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 **Administrative Data Tables** dialog.

	Data Type	Explanation
<b>Parameter:</b> ⇒ nodeRecordContent	System.Object	Content of the node record
<b>Return value</b>	System.Boolean	Node record was/was not deleted successfully.

### Example Code VB.NET

```
VcGantt1.DeleteNodeRecord("A100;;;;;")
```

### Example Code C#

```
vcGantt1.DeleteNodeRecord("A100;;;;;");
```

## DetectDataTableFieldName

**Method of VcGantt**

This property lets you retrieve the name of a data table field by its index.

	Data Type	Explanation
<b>Parameter:</b> ⇒ fieldIndex	System.Int32	Index of the data table field of which the name is to be retrieved
<b>Return value</b>	System.String	Name of the data table field returned

### Example Code VB.NET

```
'Find the name of a DataTableField
Dim fieldName As String

fieldName = VcGanttASP1.DetectDataTableFieldName(0)
```

### Example Code C#

```
//Find the name of a DataTableField
string fieldName = vcGanttASP1.DetectDataTableFieldName(0);
```

## DetectDataTableName

**Method of VcGantt**

This property lets you retrieve the name of a data table by its index.

	Data Type	Explanation
<b>Parameter:</b> ⇒ fieldIndex	System.Int32	Index of the data table of which the name is to be retrieved
<b>Return value</b>	System.String	Name of the data table returned

**Example Code VB.NET**

```
'Find the name of a DataTable
Dim tableName As String

tableName = VcGanttASP1.DetectDataTableName(0)
```

**Example Code C#**

```
//Find the name of a DataTable
string tableName = vcGanttASP1.DetectDataTableName(0);
```

**DetectFieldIndex****Method of VcGantt**

This property lets you retrieve the index of a data table field by its name and the name of the data table.

	Data Type	Explanation
<b>Parameter:</b> ⇒ dataTableFieldName	System.String	Name of the data table field of which the index is to be retrieved
⇒ dataTableName	System.String	Name of the data table that holds the field of which the index is to be retrieved
<b>Return value</b>	System.Int32	Index of the data table field returned

**Example Code VB.NET**

```
'Find the index of a DataTableField
Dim fieldIndex As Integer

fieldIndex = VcGanttASP1.DetectFieldIndex("Maindata", "Name")
```

**Example Code C#**

```
//Find the index of a DataTableField
int fieldIndex = vcGanttASP1.DetectFieldIndex("Maindata", "Name");
```

## 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
<b>Return value</b>	System.Boolean	Loading was successfully finished

### Example Code VB.NET

```
VcGantt1.EndLoading()
```

### Example Code C#

```
vcGantt1.EndLoading();
```

## ExportGraphicsToFileEx

**Method of VcGantt**

This method lets you store a Gantt diagram to a file without generating a **Save as** dialog box. Possible formats for saving:

- \*.BMP (Microsoft Windows Bitmap)
- \*.EMF (Enhanced Metafile)
- \*.EMF, with EMF+ included
- \*.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

## 558 API Reference: VcGantt

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 further details on the different formats please read the chapter "Important Concepts: Graphics Formats".

Further details on the different formats please find in the chapter **Important Concepts: Graphics Formats**.

	Data Type	Explanation
<b>Parameter:</b>		
⇒ fileName	System.String	File name (including a path, if necessary)
⇒ printOutputFormat	PrintOutputFormat	Format of the file to be stored.
	<b>Possible Values:</b> .vcBMP 2 .vcEMF 9 .vcEMFPlus 12  .vcEMFWithEMFPlusIncluded 11  .vcEPS 3 .vcGIF 4 .vcJPG 5 .vcPCX 6 .vcPNG 7 .vcTIF 8 .vcVMF 0 .vcWMF 1 .vcWMFWithEMFIncluded 10	File is put out in the format BMP. File is put out in the format EMF. File is put out as a *.EMF file but is a pure EMF+ format. File is put out as a *.EMF file and in addition includes the EMF+ format. Deprecated File is put out in the format GIF. File is put out in the format JPG. Deprecated File is put out in the format PNG. File is put out in the format TIF. File is put out in the format VMF. File is put out in the format WMF. File is put out as a *.WMF file and in addition includes the EMF format.
⇒ SizeX	System.Int16	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.
⇒ SizeY	System.Int16	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.
<b>Return value</b>	System.Boolean	File was (true) / was not (false) stored successfully.

### Example Code VB.NET

```
VcGanttASP1.ExportGraphicsToFile "C:\Tmp\test1.vmf", vcVMF,0,0
```

**Example Code C#**

```
vcGanttASP1.ExportGraphicsToFile(@"c:\Tmp\test.vmf",
VcPrintOutputFormat.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
<b>Parameter:</b> ⇒ fitMode	VcFitMode  <b>Possible Values:</b> .vcFitHeight 23  .vcFitMaximumOfWidthAnd Height 1051 .vcFitMinimumOfWidthAnd Height 1052 .vcFitWidth 24  .vcUseLargerZoomFactor 1053  .vcUseSmallerZoomFactor 1054	Selection of zoom factor  The diagram is adjusted height-wise to the control size. The largest dimension of the diagram is adjusted to the control size. The smallest dimension of the diagram is adjusted to the control size. The diagram is adjusted width-wise to the control size. The larger of the zoom factors is used. The corresponding dimension of the diagram does not fit into the frame of the control. The smaller of the zoom factors is used and the corresponding dimension of the diagram fits completely into the control.
<b>Return value</b>	System.Int32	Relative zoom factor

**Example Code VB.NET**

```
VcGanttASP1.(FitChartIntoView(VcFitMode.vcFitWidth))
```

**Example Code C#**

```
vcGanttASP1.FitChartIntoView(VcFitMode.vcFitWidth);
```



## 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
<b>Return value</b>	System.Boolean	The histograms had to (True) / did not have to (False) be re-scaled.

### Example Code VB.NET

```
VcGantt1.FitHistogramsIntoView = True
```

### Example Code C#

```
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
<b>Parameter:</b>		
⇒ startDate	System.DateTime	Start date of the area to be matched
⇒ endDate	System.DateTime	End date of the area to be matched
⇒ gapAsNoOfTimeUnits	System.Int32	Number of time units to form the "gap" between startDate/endDate and the beginning of the visible section of the time scale start/end
<b>Return value</b>	System.Boolean	Area could/could not be matched.

### Example Code VB.NET

```
VcGantt1.FitRangeIntoView("14.09.2014", "21.09.2014", 1)
```

**Example Code C#**

```
vcGantt1.FitRangeIntoView(Convert.ToDateTime("14.09.2014"),
Convert.ToDateTime("21.09.2014"),1);
```

**GetCurrentViewDates****Method of VcGantt**

This method lets you retrieve the start and end dates of the visible section of the time scale.

	Data Type	Explanation
<b>Parameter:</b>		
↔ leftDate	System.DateTime	Start date of the visible section of the time scale
↔ rightDate	System.DateTime	End date of the visible section of the time scale
<b>Return value</b>	System.Boolean	Start/end dates of the visible section of the time scale are returned/not returned.

**Example Code VB.NET**

```
Dim bGetCurrentViewDates As Boolean
Dim leftDate As Date
Dim rightDate As Date
GetCurrentViewDates = VcGantt1.GetCurrentViewDates(leftDate, rightDate)
```

**Example Code C#**

```
DateTime leftDate = new DateTime();
DateTime rightDate = new DateTime();
bool bGetCurrentViewDates = vcGantt1.GetCurrentViewDates(ref leftDate, ref
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
<b>Parameter:</b>		
⇒ x	System.Int32	X coordinate in the Gantt diagram, the corresponding date of which is to be retrieved
<b>Return value</b>	System.DateTime	Date retrieved

### Example Code VB.NET

```
Private Sub VcGantt1_VcDiagramLeftClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcDiagramClickingEventArgs) Handles
VcGantt1.VcDiagramLeftClicking
    Label1.Name = VcGantt1.GetDate(e.X)
End Sub
```

### Example Code C#

```
private void vcGantt1_VcDiagramLeftClicking(object sender,
NETRONIC.XGantt.VcDiagramClickingEventArgs e)
{
    label1.Text = vcGantt1.GetDate(e.X).ToString();
}
```

## GetLicenseInformation

**Method of VcGantt**

This method returns an HTML string containing informations about licensing, the position of the control and its loaded dependent DLLs in the file system. The method serves for troubleshooting and hence is only used for debugging an application.

	Data Type	Explanation
<b>Return value</b>	System.String	License information retrieved

## GetLinkByID

**Method of VcGantt**

This method gives access to a link by its identification which was specified on the **Administrative 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
<b>Parameter:</b> ⇒ linkID	System.Object	Link identification
<b>Return value</b>	VcLink	Link

**Example Code VB.NET**

```
Dim link As VcLink
Dim successor As Integer
link = VcGantt1.GetLinkByID(" 1")
successor = link.DataField(2)
```

**Example Code C#**

```
VcLink link = vcGantt1.GetLinkByID(" 1");
int successor = Convert.ToInt32(link.get_DataField(2));
```

## GetLinkByNodeIDs

**Method of VcGantt**

This method lets you access a link by the ID of its predecessor and 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
<b>Parameter:</b>		
⇒ predecessorID	System.String	Identification of the predecessor node
⇒ successorID	System.String	Identification of the successor node
<b>Return value</b>	VcLink	Link

**Example Code VB.NET**

```
Dim link As VcLink
link = VcGantt1.GetLinkByNodeIDs(" 2", " 3")
```

**Example Code C#**

```
VcLink link = vcGantt1.GetLinkByNodeIDs(" 2", " 3");
```

## GetNodeByID

**Method of VcGantt**

This method lets you access a node by its identification which was specified on the **Administrative Data Tables** dialog. If the identification consists of more than one field (composite primary key), the multipart ID needs to be noted as shown below:

**ID=ID1|ID2|ID3**

	Data Type	Explanation
<b>Parameter:</b> ⇒ nodeID	System.Object	Node identification
<b>Return value</b>	VcNode	Node

**Example Code VB.NET**

```
Dim node As VcNode
node = VcGantt1.GetNodeByID("10")
```

**Example Code C#**

```
VcNode node = vcGantt1.GetNodeByID("10");
```

## GetViewComponentSize

**Method of VcGantt**

This method lets you require at run time the size and position of a graphical element of the VARCHART XGantt control (time scale, diagram, histogram, table, table caption etc.) (see event **VcViewComponentsSizeModified**).

**Note:**

1. The position refers to the origin of the graphical element of the VARCHART XGantt control.
2. The values returned are pixel values.

	Data Type	Explanation
<b>Parameter:</b> ⇒ viewComponent	VcComponentType  <b>Possible Values:</b> .vcAdditionalListComponent 1 .vcBottomListTitleComponent 14 .vcBottomRightListTitleComponent 17 .vcBottomTimeScaleComponent 15 .vcDiagramComponent 4 .vcHistogramComponent 8 .vcHistogramVerScaleComponent 7 .vcLegendComponent 10  .vcListComponent 0 .vcListTitleComponent 2 .vcTimeScaleComponent 3 .vcTopTitleComponent 11	Component type  additional table bottom title bar bottom right table bottom time scale diagram histogram numeric scale (vertical histogram scale) legend (currently functionless; return values 00) table table title upper time scale upper title bar
⇒ x	System.Int32	X coordinate of the component
⇒ y	System.Int32	Y coordinate of the component
⇒ width	System.Int32	Component width

⇐ height	System.Int32	Component height
<b>Return value</b>	Void	

**Example Code VB.NET**

```
Private Sub handleHideHistogram()
    Dim x As Integer
    Dim y As Integer
    Dim width As Integer
    Dim height As Integer
    VcGantt1.GetViewComponentSize(VcComponentType.vcHistogramVerScaleComponent,
x, y, width, height)
    ' plus 6 because of the sash
    TextBox1.Top = VcGantt1.Top + y + 6
    TextBox1.Left = VcGantt1.Left + x
    ' minus 25 because of the numeric scale
    TextBox1.Width = width - 25
    ' minus 6 because of the sash
    TextBox1.Height = height - 6
End Sub
```

**Example Code C#**

```
private void handleHideHistogram()
{
    int x;
    int y;
    int width;
    int height;
    vcGantt1.GetViewComponentSize(VcComponentType.vcHistogramVerScaleComponent,
ref x, ref y, ref width, ref height);
    // plus 6 because of the sash
    textBox1.Top = vcGantt1.Top + y + 6;
    textBox1.Left = vcGantt1.Left + x;
    // minus 25 because of the numeric scale
    textBox1.Width = width - 25;
    // minus 6 because of the sash
    textBox1.Height = height - 6;
}
```

## GroupNodes

**Method of VcGantt**

This methods lets you activate/deactivate the grouping. If you have set a grouping field by the **GroupingDataFieldIndex** property or if you have set the grouping order by the **GroupSortingDataFieldIndex** property, you need to activate the grouping by **GroupNodes**.

	Data Type	Explanation
<b>Parameter:</b>		
⇐ onOff	System.Boolean	Grouping on/off
<b>Return value</b>	System.Boolean	Nodes were/were not grouped successfully.

### Example Code VB.NET

```
VcGantt1.GroupingDataFieldIndex(0) = 11  
VcGantt1.GroupSortingDataFieldIndex(0) = 12
```

```
VcGantt1.GroupNodes(True)
```

### Example Code C#

```
vcGantt1.set_GroupingDataFieldIndex(0, 11);  
vcGantt1.set_GroupSortingDataFieldIndex(0,12);
```

```
vcGantt1.GroupNodes(true);
```

## IdentifyField

Method of VcGantt

This method lets you identify the index of a data field the content of which is to be displayed in the table field at the given cursor position.

	Data Type	Explanation
<b>Parameter:</b>		
⇒ x	System.Int32	X coordinate of the cursor
⇒ y	System.Int32	Y coordinate of the cursor
<b>Return value</b>	System.Int32	Data field index identified -1 if there is no table field at the given position

### Example Code VB.NET

```
Dim intField As Integer  
Dim text As String  
  
intField = VcGantt1.IdentifyField(x, y)  
  
If intField > -1 Then  
    text = node.DataField(intField)  
End If
```

### Example Code C#

```
string text;  
  
int i = vcGantt1.IdentifyField(x, y) ;  
  
if (i > -1)  
    text = Convert.ToString(node.get_DataField(i)) ;
```

## IdentifyLayerAt

Method of VcGantt

This method lets you identify a layer. If 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**.

	Data Type	Explanation
<b>Parameter:</b>		
⇒ x	System.Int32	X coordinate of the cursor
⇒ y	System.Int32	Y coordinate of the cursor
⇒ referenceNode	VcNode	Reference node
⇐ identifiedLayer	VcLayer	Layer identified
<b>Return value</b>	System.Boolean	Object identified/no object identified

### Example Code VB.NET

```
Dim identifiedObj As Object
Dim identifiedObjType As VcObjectType
Dim identifiedLayer As VcLayer
Dim node As VcNode

VcGantt1.IdentifyObjectAt(e.X, e.Y, identifiedObj, identifiedObjType)
node = identifiedObj

Select Case identifiedObjType
    Case VcObjectType.vcObjTypeNodeInDiagram
        VcGantt1.IdentifyObjectAt(e.X, e.Y, identifiedObj, identifiedObjType)
        If Not identifiedLayer Is Nothing Then
            MsgBox("The Node " + node.DataField(0) + " , Layer " +
                identifiedLayer.Name + " , was identified in the diagram area.")
        Else
            MsgBox("The Node" + node.DataField(0) + " was identified in diagram
                area; no layer was identified")
        End If
    Case VcObjectType.vcObjTypeNodeInTable
        MsgBox("The Node" + node.DataField(0) + " was identified via table")
    Case Else
        MsgBox("No node was identified")
End Select
```



**Example Code C#**

```

object identifiedObj = null;
VcObjectType identifiedObjType = VcObjectType.vcObjTypeNodeInDiagram;
VcLayer identifiedLayer = null;
VcNode node;

vcGantt1.IdentifyObjectAt(e.X, e.Y, ref identifiedObj, ref identifiedObjType);
node = (VcNode)identifiedObj;

switch (identifiedObjType)
{
    case VcObjectType.vcObjTypeNodeInDiagram:
        vcGantt1.IdentifyObjectAt(e.X, e.Y, ref identifiedObj, ref
identifiedObjType);
        if (identifiedLayer != null)
            MessageBox.Show("The Node " + node.get_DataField(0) + " , Layer " +
identifiedLayer.Name + ", was identified in the diagram area.");
        else
            MessageBox.Show("The Node" + node.get_DataField(0) + " was identified
in diagram area; no layer was identified");
        break;
    case VcObjectType.vcObjTypeNodeInTable:
        MessageBox.Show("The Node" + node.get_DataField(0) + " was identified via
table");
        break;
    default:
        MessageBox.Show("No node was identified");
        break;
}

```

**IdentifyObjectAt****Method of VcGantt**

This method lets you identify any object in VARCHART XGantt. The object type will be returned. If 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**.

	Data Type	Explanation
<b>Parameter:</b>		
⇒ x	System.Int32	X coordinate of the cursor
⇒ y	System.Int32	Y coordinate of the cursor
⇐ identifiedObject	VcObject	Object identified
⇐ identifiedObjectType	VcObjectType	Type of the object identified
	<b>Possible Values:</b>	
	.vcObjTypeBox 15	object type <b>box</b>
	.vcObjTypeCurve 12	object type <b>curve</b>
	.vcObjTypeDateLine 9	object type <b>date line</b>
	.vcObjTypeGroup 7	object type <b>group</b>
	.vcObjTypeGroupInDiagram 11	object type <b>group in diagram area</b>
	.vcObjTypeGroupInTable 7	object type <b>group in table area</b>
	.vcObjTypeHistogram 13	object type <b>histogram</b>
	.vcObjTypeLayer 8	object type <b>layer</b>
	.vcObjTypeLinkCollection 3	object type <b>link collection</b>
	.vcObjTypeNodeInDiagram 2	object type <b>node in diagram area</b>

	.vcObjTypeNodeInLegend 17 .vcObjTypeNodeInTable 1 .vcObjTypeNone 0 .vcObjTypeNumericScale 10 .vcObjTypeSummaryNode 14 .vcObjTypeTable 4 .vcObjTypeTableCaption 5 .vcObjTypeTimeScale 6	object type <b>node in legend area</b> object type <b>node in table area</b> no object object type <b>numeric scale</b> object type <b>summary bar</b> object type <b>table</b> object type <b>table caption</b> object type <b>time scale</b>
<b>Return value</b>	System.Boolean	Object identified/no object identified

**Example Code VB.NET**

```

Private Sub VcGanttASP1_MouseMove(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles VcGanttASP1.MouseMove

    Dim identifiedObject As Object = Nothing
    Dim identifiedObjectType As VcObjectType = VcObjectType.vcObjTypeNone
    Dim node As VcNode = Nothing
    Dim identifiedLayer As VcLayer = Nothing

    VcGanttASP1.IdentifyObjectAt(e.X, e.Y, identifiedObject,
identifiedObjectType)

    Select Case identifiedObjectType
        Case VcObjectType.vcObjTypeNodeInDiagram
            node = identifiedObject

            VcGanttASP1.IdentifyLayerAt(e.X, e.Y, node, identifiedLayer)

            If identifiedLayer IsNot Nothing Then
                Labell1.Text = "X = " & e.X & "    Y = " & e.Y & vbCrLf & _
                    "Node ID = " & node.DataField(0) & vbCrLf & _
                    "Layer Name = " & identifiedLayer.Name
            End If

        Case Else
            Labell1.Text = ""
        End Select
    End Select

End Sub

```

### Example Code C#

```
private void VcGanttASP1_MouseMove(object sender, MouseEventArgs e)
{
    object identifiedObject = null;
    VcObjectType identifiedObjectType = VcObjectType.vcObjTypeNone;
    VcNode node = null;
    VcLayer identifiedLayer = null;

    VcGanttASP1.IdentifyObjectAt(e.X, e.Y, ref identifiedObject, ref
identifiedObjectType);

    switch (identifiedObjectType)
    {
        case VcObjectType.vcObjTypeNodeInDiagram:
            {
                node = (VcNode)identifiedObject;

                VcGanttASP1.IdentifyLayerAt(e.X, e.Y, node, ref
identifiedLayer);

                if (identifiedLayer != null)
                    labell1.Text = "X = " + e.X + "    Y = " + e.Y +
                        "\nNode ID = " + node.get_DataField(0) +
                        "\nLayer Name = " + identifiedLayer.Name;

                break;
            }
        default:
            {
                labell1.Text = "";
                break;
            }
    }
}
```

## ImportConfiguration

**Method of VcGantt**

This method enables a configuration file (\*.ini) to be loaded, which all settings are adopted from, including the corresponding data interface (\*.ifd).

You can specify either a local file including the path or an URL.

**Note:** When loading a new configuration file, the data are lost and have to be imported again if necessary.

	Data Type	Explanation
<b>Parameter:</b> ⇒ fileName	System.String	Name of file to be imported
<b>Return value</b>	Void	

**Example Code VB.NET**

```
VcGantt1.ImportConfiguration ( "c:\VARCHART\XGantt\sample.ini")
'or
VcGantt1.ImportConfiguration
("http://members.tripod.de/netronic_te/xgantt_sample.ini)
```

**Example Code C#**

```
vcGantt1.ImportConfiguration (@"c:\VARCHART\XGantt\sample.ini");
// or
vcGantt1.ImportConfiguration
(@"http://members.tripod.de/netronic_te/xgantt_sample.ini");
```

**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 **Administrative 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
<b>Parameter:</b>		
⇒ linkRecordContent	System.Object[]	Content of the link record
<b>Return value</b>	VcLink	Link

**Example Code VB.NET**

```
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("1;A100;A105;FS;0")
VcGantt1.EndLoading()
```

**Example Code C#**

```
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("1;A100;A105;FS;2");
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 **Administrative 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
<b>Parameter:</b> ⇒ nodeRecordContent	Data field	Content of the node record
<b>Return value</b>	VcNode	Node

**Example Code VB.NET**

```
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 Object = New Object(5) {"A100", "Activity 1", "12.09.14",
"17.09.14", "5", "Planning"}
VcGantt1.InsertNodeRecord(nodeRecord)
VcGantt1.EndLoading()
```

**Example Code C#**

```
string nodeRecord = "A100;Activity 1;12.09.14;17.09.14;5;Planning";
vcGantt1.InsertNodeRecord(nodeRecord);
vcGantt1.EndLoading();
```

**Load****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. The records are allocated to the corresponding data tables by using an appropriate identification line. CSV-Files may be retrieved and written in ANSI as well as in Unicode coding which is automatically recognized when read

```
**** 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
<b>Parameter:</b> ⇒ fileName	System.String	File name
<b>Return value</b>	System.Boolean	File was/was not opened successfully.

**Example Code VB.NET**

```
vcgantt1.Load("c:\Data\project1.bar")
```

**Example Code C#**

```
vcGantt1.Load(@"c:\Data\project1.bar");
```

## OptimizeTimeScaleStartEnd

**Method of VcGantt**

This method lets you define the start and the end date of the timescale 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
<b>Parameter:</b> ⇒ noOfUnits	System.Int16	Number of time units
<b>Return value</b>	System.Boolean	Timescale was/was not optimized successfully.

**Example Code VB.NET**

```
VcGantt1.OptimizeTimeScaleStartEnd(5)
```

**Example Code C#**

```
vcGantt1.OptimizeTimeScaleStartEnd(5);
```

## PrintEx

**Method of VcGantt**

This method lets you print the diagram directly. A dialog box will not be displayed.

## 574 API Reference: VcGantt

	Data Type	Explanation																		
<b>Return value</b>	VcPrintResultStatus	<b>Possible values:</b> <table border="1"><thead><tr><th>Name</th><th>parameter position</th><th>description</th></tr></thead><tbody><tr><td>vcPrintingSucceeded</td><td>0</td><td>Printing was performed successfully.</td></tr><tr><td>vcNoPrinterInstalled</td><td>1</td><td>No printer was found neither the one specified by the call <b>VcPrinter.PrinterName</b> nor the one labeled as default printer by the Windows operating system.</td></tr><tr><td>vcPrintingAbortedByUser</td><td>2</td><td>Printing was aborted by the user.</td></tr><tr><td>vcPrintingAbortedByDriver</td><td>3</td><td>Printing was aborted by the Windows printer driver.</td></tr><tr><td>vcUnprintablePageLayout</td><td>4</td><td>Printing could not be performed since the page layout did not match the printer properties such as paper size or margins.</td></tr></tbody></table>	Name	parameter position	description	vcPrintingSucceeded	0	Printing was performed successfully.	vcNoPrinterInstalled	1	No printer was found neither the one specified by the call <b>VcPrinter.PrinterName</b> 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.
Name	parameter position	description																		
vcPrintingSucceeded	0	Printing was performed successfully.																		
vcNoPrinterInstalled	1	No printer was found neither the one specified by the call <b>VcPrinter.PrinterName</b> 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 VB.NET

```
VcGantt1.PrintEx()
```

### Example Code C#

```
vcGantt1.PrintEx();
```

## PrintToFile

### Method of VcGantt

This method lets you print the diagram directly into a file. Whether this is successful depends on the printer driver because many PDF printer drivers don't accept file names.

	Data Type	Explanation
<b>Parameter:</b> ⇒ fileName	System.String	File name
<b>Return value</b>	Void	

## 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
<b>Return value</b>	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
<b>Parameter:</b> ⇒ resetAction	VcResetAction  <b>Possible Values:</b> .vcEmptyAllDataTables 4 .vcReloadConfiguration 2 .vcRemoveGroups 0 .vcRemoveNodes 1	Objects to be initialized or deleted  The contents of all data tables are deleted but the data tables are kept. Complete reinitialization with the INI-file. All settings and created objects expire. All groups and dependent objects and with that also all nodes and links are deleted. All nodes and dependent objects and, if necessary all links are deleted.
<b>Return value</b>	System.Boolean	The objects in the diagram were deleted successfully.  {True}



### Example Code VB.NET

```
VcGanttASP1.Reset (VcResetAction.vcRemoveNodes)
```

### Example Code C#

```
vcGanttASP1.Reset (VcResetAction.vcRemoveNodes);
```

## 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 (corresponding to the common **Save** function).

	Data Type	Explanation
<b>Parameter:</b>		
⇒ fileName	System.String	Name of the file to be saved
⇒ encoding	VcEncoding	Mode of encoding
	<b>Possible Values:</b> .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 VARCHAR control, it has to be treated in a special way.
<b>Return value</b>	System.Boolean	File was/was not stored successfully.

### Example Code VB.NET

```
VcGanttASP1.SaveAsEx ("C:\ProjectData.txt", VcEncoding.vcANSIEncoding)
```

### Example Code C#

```
vcGanttASP1.SaveAsEx (@"C:\ProjectData.txt", VcEncoding.vcANSIEncoding);
```

## 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 timescale 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 timescale.

The time unit can be set on the **General** property page.

	Data Type	Explanation
<b>Parameter:</b>		
⇒ date	System.DateTime	Date
⇒ horAlignment	VcHorizontalAlignment	Horizontal alignment
	<b>Possible Values:</b> .vcHorCenterAligned - 1 .vcLeftAligned -3 .vcRightAligned -2	horizontally centered left aligned right aligned
⇒ gapAsNoOfTimeUnits	System.Int32	Number of time units
<b>Return value</b>	System.Boolean	Scrolling was/was not performed successfully.

#### Example Code VB.NET

```
VcGantt1.ScrollToDate("20.10.14", VcHorizontalAlignment.vcLeftAligned, 2)
```

#### Example Code C#

```
vcGantt1.ScrollToDate(Convert.ToDateTime("20.10.14"),  
VcHorizontalAlignment.vcRightAligned, 2);
```

## ScrollToGroupLine

**Method of VcGantt**

This method allows to scroll to the row containing a particular group node and to specify whether that group node should be displayed at the top, in the center or at the bottom of the screen.

	Data Type	Explanation
<b>Parameter:</b>		
⇒ group	VcGroup	Group to be scrolled to
⇒ verAlignment	VcVerticalAlignment	Vertical alignment
	<b>Possible Values:</b> .vcBottomAligned 2 .vcTopAligned 1 .vcVerCenterAligned - 1	bottom aligned top aligned vertically centered
<b>Return value</b>	System.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
<b>Parameter:</b>		
⇒ node	VcNode	Node
⇒ verAlignment	VcVerticalAlignment  <b>Possible Values:</b> .vcBottomAligned 2 .vcTopAligned 1 .vcVerCenterAligned - 1	Vertical alignment  bottom aligned top aligned vertically centered
<b>Return value</b>	System.Boolean	Scrolling was/was not performed successfully.

### Example Code VB.NET

```
Dim node As VcNode
node = VcGantt1.GetNodeByID(" 2")
VcGantt1.ScrollToNodeLine(node, VcVerticalAlignment.vcVerCenterAligned)
```

### Example Code C#

```
object[] objDataRecord = new object[5];

vcGantt1.ExtendedDataTablesEnabled = true;
vcGantt1.MinimumRowHeight = 1000;

vcGantt1.TimeScaleEnd = new DateTime(2010, 8, 1);
vcGantt1.TimeScaleStart = new DateTime(2010, 6, 1);

objDataRecord[2] = new DateTime(2010, 6, 3);
objDataRecord[3] = new DateTime(2010, 6, 10);
objDataRecord[4] = 5;

VcDataRecordCollection dataRecordCol =
vcGantt1.DataTableCollection.DataTableByName("Maindata").DataRecordCollection;
for (int i = 1; i < 100; i++)
{
    objDataRecord[0] = i;
    objDataRecord[1] = "Node " + i.ToString();

    dataRecordCol.Add(objDataRecord);
}
vcGantt1.EndLoading();
vcGantt1.ScrollToNode(vcGantt1.GetNodeByID("50"),
VcVerticalAlignment.vcTopAligned);
```

## 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
<b>Parameter:</b>		
⇒ node	VcNode	Node to be scrolled to
⇒ verAlignment	VcVerticalAlignment	Vertical alignment
	<b>Possible Values:</b> .vcBottomAligned 2 .vcTopAligned 1 .vcVerCenterAligned - 1	bottom aligned top aligned vertically centered
<b>Return value</b>	System.Boolean	Scrolling was (true) / was not (false) performed successfully.

### Example Code VB.NET

```
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)
```

**Example Code C#**

```
using NETRONIC.XGantt;
...
object[] objDataRecord = new object[2];
vcGantt1.ExtendedDataTablesEnabled = true;
VcDataRecordCollection dataRecordCol =
vcGantt1.DataTableCollection.DataTableByName("Maindata").DataRecordCollection;
    for (int i = 1; i < 100; i++)
    {
        objDataRecord[0] = i;
        objDataRecord[1] = "Node " + i.ToString();
        dataRecordCol.Add(objDataRecord);
    }
vcGantt1.EndLoading();
vcGantt1.ScrollToNodeLine(vcGantt1.GetNodeByID("50"),
VcVerticalAlignment.vcTopAligned);
```

**SortGroups****Method of VcGantt**

This method lets you start the sorting of groups in a grouped diagram in accordance with the defined sorting parameter **GroupSortingDataFieldIndex (GroupingLevel)**.

	Data Type	Explanation
<b>Return value</b>	System.Boolean	Groups were/were not sorted successfully.

**Example Code VB.NET**

```
VcGantt1.GroupSortingDataFieldIndex(0) = 12
VcGantt1.SortGroups()
```

**Example Code C#**

```
vcGantt1.set_GroupSortingDataFieldIndex(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 (**NodeSortingDataFieldIndex (sortLevel)** and **NodeSortingOrder (sortLevel)**). If a grouping is activated, the sorting will be done separately for each group.

	Data Type	Explanation
<b>Return value</b>	System.Boolean	Nodes were/were not sorted successfully.

**Example Code VB.NET**

```
VcGantt1.NodeSortingDataFieldIndex(0) = 3
VcGantt1.NodeSortingOrder(0) = VcNodeSortingOrder.vcAscending
VcGantt1.SortNodes()
```

**Example Code C#**

```
vcGantt1.set_NodeSortingDataFieldIndex(0,3);
vcGantt1.set_NodeSortingOrder(0, VcNodeSortingOrder.vcAscending);
vcGantt1.SortNodes();
```

## SuspendUpdate

**Method of VcGantt**

For projects comprising many nodes, updating procedures may be very time consuming if actions are repeated by 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
<b>Parameter:</b> ⇒ suspendFlag	System.Boolean	SuspendUpdate(True): Start of the SuspendUpdate method/ SuspendUpdate(False): end of the SuspendUpdate method
<b>Return value</b>	Void	

**Example Code VB.NET**

```
VcGantt1.SuspendUpdate(True)

If updateFlag Then
    For Each node In nodeCltn
        If (node.DataField(2) < "07.09.98") Then
            node.DataField(13) = "X"
            node.Update()
            counter = counter + 1
        End If
    Next
Else
    For Each node In nodeCltn
        If (node.DataField(2) < "07.09.98") Then
            node.DataField(13) = ""
            node.Update()
            counter = counter + 1
        End If
    Next
End If

VcGantt1.SuspendUpdate(False)
```

### Example Code C#

```

bool updateFlag = true;
VcNodeCollection nodeCltn = vcGantt1.NodeCollection;
int counter = 0;

vcGantt1.SuspendUpdate(true);
if (updateFlag == true)
{
    foreach (VcNode node in nodeCltn)
    {
        if (DateTime.Compare(Convert.ToDateTime(node.get_DataField(2)),
Convert.ToDateTime("12.09.07")) < 0)
        {
            node.set_DataField(13, "X");
            node.Update();
            counter = counter + 1;
        }
    }
}
else
{
    foreach(VcNode node in nodeCltn)
    {
        if (DateTime.Compare(Convert.ToDateTime(node.get_DataField(2)),
Convert.ToDateTime("12.09.07")) < 0)
        {
            node.set_DataField(13, "");
            node.Update();
            counter = counter + 1;
        }
    }
}
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
<b>Parameter:</b>		
⇒ linkRecordContent	System.Object	Content of the link record
<b>Return value</b>	VcLink	Link updated

### Example Code VB.NET

```
VcGantt1.UpdateLinkRecord("A100;A105;FS;0")
```

### Example Code C#

```
vcGantt1.UpdateLinkRecord("1;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
<b>Parameter:</b> ⇒ nodeRecordContent	System.Object	Content of the node record
<b>Return value</b>	VcNode	Node record was/was not updated successfully.

### Example Code VB.NET

```
VcGantt1.UpdateNodeRecord("A100;Activity 1;12.09.14;17.09.14;5;Planning")
```

### Example Code C#

```
vcGantt1.UpdateNodeRecord("A100;Activity 1;12.09.07;17.09.07;5;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
<b>Return value</b>	Void	

### Example Code VB.NET

```
VcGantt1.UpdateRowNumberFields()  
VcGantt1.SaveAs("c:\tmp\data.bar")
```

### Example Code C#

```
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 properties **FitChartIntoView** and **ZoomFactor**.

	Data Type	Explanation
<b>Parameter:</b> ⇨ zoomFactor	System.Int16	relative zoom factor  {11...999}, other values will remain unconsidered
<b>Return value</b>	System.Boolean	Zooming was/was not performed successfully.

#### Example Code VB.NET

```
VcGantt1.Zoom(120)
```

#### Example Code C#

```
vcGantt1.Zoom(120);
```

---

## Events

### VcBoxLeftClicking

#### 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 returned.

	Data Type	Explanation
<b>Parameter:</b> ⇨ sender	VcGantt	Reference to the object that triggered the event
⇨ e	VcBoxClickingEventArgs	Object specific to the event that is being handled

Properties of the VcBoxClickingEventArgs object

	Data Type	Explanation
<b>Properties:</b>		
⇒ box	VcBox	Box hit
⇒ x	System.Int32	X coordinate of the mouse cursor
⇒ y	System.Int32	Y coordinate of the mouse cursor
⇔ returnStatus	VcReturnStatus	Return status
	<b>Possible Values:</b> .vcRetStatDefault 2 .vcRetStatFalse 0 .vcRetStatNoPopup 4 .vcRetStatOK 1	The default behavior remains unchanged. The default behavior will not be performed. The popup of the context menu is inhibited. The default behavior will be performed.

### Example Code VB.NET

```
Private Sub VcGantt1_VcBoxLeftClicking(ByVal sender As Object, ByVal e As
NETRONICXGantt.VcBoxClickingEventArgs) Handles VcGantt1.VcBoxLeftClicking
    TextBox1.Text = e.Box.FieldText(1)
End Sub
```

### Example Code C#

```
private void vcGantt1_VcBoxLeftClicking(object sender,
VcGanttASPLibVcBoxClickingEventArgs e)
{
    textBox1.Text = e.Box.get_FieldText(1);
}
```

## VcComponentScrolled

### 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
<b>Parameter:</b>		
⇒ sender	VcGantt	Reference to the object that triggered the event
⇒ e	VcComponentScrolledEventArgs	Object specific to the event that is being handled

### Properties of the VcComponentScrolledEventArgs object

	Data Type	Explanation
<b>Properties:</b>		
⇒ component	VcComponentType  <b>Possible Values:</b> .vcAdditionalListComponent 1 .vcBottomListTitleComponent 14 .vcBottomRightListTitleComponent 17 .vcBottomTimeScaleComponent 15 .vcDiagramComponent 4 .vcHistogramComponent 8 .vcHistogramVerScaleComponent 7 .vcLegendComponent 10  .vcListComponent 0 .vcListTitleComponent 2 .vcTimeScaleComponent 3 .vcTopTitleComponent 11	Component type  additional table bottom title bar bottom right table bottom time scale diagram histogram numeric scale (vertical histogram scale) legend (currently functionless; return values 00) table table title upper time scale upper title bar
⇒ orientation	VcScrollOrientation  <b>Possible Values:</b> .vcHorizontal 1 .vcVertical 2	Scrolling direction  horizontal scrolling vertical scrolling
⇒ scrollAction	VcScrollAction  <b>Possible Values:</b> .vcScrollActionAutoscrollDown 102 .vcScrollActionAutoscrollLeft 101 .vcScrollActionAutoscrollRight 102 .vcScrollActionAutoscrollUp 101 .vcScrollActionMouseWheelDown 106	Type of scrolling  The view was automatically scrolled downward. The view was automatically scrolled towards the right. The view was automatically scrolled towards the left. The view was automatically scrolled upward. 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 <b>End</b> button or the context menu to the diagram end (right down)
.vcScrollActionScrollHome 103	Scrolling via the <b>Pos 1</b> 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 VB.NET

```
Private Sub VcGantt1_VcComponentScrolled(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcComponentScrolledEventArgs) Handles
VcGantt1.VcComponentScrolled
    If e.ScrollOrientation = VcScrollOrientation.vcHorizontal And e.ScrollAction
= VcScrollAction.vcScrollActionSBPageLeft Then
        MsgBox("Scrolled left")
    ElseIf e.ScrollOrientation = VcScrollOrientation.vcHorizontal And
e.ScrollAction = VcScrollAction.vcScrollActionSBPageRight Then
        MsgBox("Scrolled right")
    ElseIf e.ScrollOrientation = VcScrollOrientation.vcVertical And
e.ScrollAction = VcScrollAction.vcScrollActionSBPageUp Then
        MsgBox("Scrolled up")
    ElseIf e.ScrollOrientation = VcScrollOrientation.vcHorizontal And
e.ScrollAction = VcScrollAction.vcScrollActionSBPageDown Then
        MsgBox("Scrolled down")
    End If
End Sub
```

### Example Code C#

```
private void vcGantt1_VcComponentScrolled(object sender,
NETRONIC.XGantt.VcComponentScrolledEventArgs e)
{
    if (e.ScrollOrientation == VcScrollOrientation.vcHorizontal && e.ScrollAction
== VcScrollAction.vcScrollActionSBPageLeft)
        MessageBox.Show("Scrolled left");
    else if (e.ScrollOrientation == VcScrollOrientation.vcHorizontal &&
e.ScrollAction == VcScrollAction.vcScrollActionSBPageRight)
        MessageBox.Show("Scrolled right");
    else if (e.ScrollOrientation == VcScrollOrientation.vcVertical &&
e.ScrollAction == VcScrollAction.vcScrollActionSBPageUp)
        MessageBox.Show("Scrolled up");
    else if (e.ScrollOrientation == VcScrollOrientation.vcVertical &&
e.ScrollAction == VcScrollAction.vcScrollActionSBPageDown)
        MessageBox.Show("Scrolled down");
}
```

## VcComponentScrolling

### Event of VcGantt

This event occurs when you have ordered a scroll action, but before the integrated scrolling process is performed. This event lets you acquire for each interactive scroll action:

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.

If you set the `returnStatus` to **`vcRetStatFalse`**, the integrated scrolling process will be suppressed, and in your application, you can react to the event with 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 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
<b>Parameter:</b>		
⇒ sender	VcGantt	Reference to the object that triggered the event
⇒ e	VcComponentScrollingEventArgs	Object specific to the event that is being handled

### Properties of the VcComponentScrollingEventArgs object

	Data Type	Explanation
<b>Properties:</b>		
⇒ component	VcComponentType  <b>Possible Values:</b> .vcAdditionalListComponent 1 .vcBottomListTitleComponent 14 .vcBottomRightListTitleComponent 17 .vcBottomTimeScaleComponent 15 .vcDiagramComponent 4 .vcHistogramComponent 8 .vcHistogramVerScaleComponent 7  .vcLegendComponent 10  .vcListComponent 0 .vcListTitleComponent 2 .vcTimeScaleComponent 3 .vcTopTitleComponent 11	Component type  additional table bottom title bar bottom right table bottom time scale diagram histogram numeric scale (vertical histogram scale) legend (currently functionless; return values 00) table table title upper time scale upper title bar
⇒ histogramsHeightRatio		ratio of the histogram height to the complete diagram
⇒ orientation	VcScrollOrientation  <b>Possible Values:</b> .vcHorizontal 1 .vcVertical 2	Scrolling direction  horizontal scrolling vertical scrolling
⇒ scrollAction	VcScrollAction  <b>Possible Values:</b> .vcScrollActionAutoscrollDown 102 .vcScrollActionAutoscrollLeft 101 .vcScrollActionAutoscrollRight 102 .vcScrollActionAutoscrollUp 101 .vcScrollActionMouseWheelDown 106 .vcScrollActionMouseWheelLeft 105	Type of scrolling  The view was automatically scrolled downward. The view was automatically scrolled towards the right. The view was automatically scrolled towards the left. The view was automatically scrolled upward. While the mouse wheel was pressed, the mouse was moved downward. 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 <b>End</b> button or the context menu to the diagram end (right down)
	.vcScrollActionScrollHome 103	Scrolling via the <b>Pos 1</b> 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	System.Int32	Scrolling length (in pixels)
↔ returnStatus	VcReturnStatus	Return status
	<b>Possible Values:</b>	
	.vcRetStatDefault 2	The default behavior remains unchanged.
	.vcRetStatFalse 0	The default behavior will not be performed.
	.vcRetStatNoPopup 4	The popup of the context menu is inhibited.
	.vcRetStatOK 1	The default behavior will be performed.

**Example Code VB.NET**

```

Private Sub VcGantt1_VcComponentScrolling(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcComponentScrollingEventArgs) Handles
VcGantt1.VcComponentScrolling
    If e.ScrollOrientation = VcScrollOrientation.vcHorizontal And e.ScrollAction
= VcScrollAction.vcScrollActionSBPageLeft Then
        MsgBox("Scrolled left")
    ElseIf e.ScrollOrientation = VcScrollOrientation.vcHorizontal And
e.ScrollAction = VcScrollAction.vcScrollActionSBPageRight Then
        MsgBox("Scrolled right")
    ElseIf e.ScrollOrientation = VcScrollOrientation.vcVertical And
e.ScrollAction = VcScrollAction.vcScrollActionSBPageUp Then
        MsgBox("Scrolled up")
    ElseIf e.ScrollOrientation = VcScrollOrientation.vcHorizontal And
e.ScrollAction = VcScrollAction.vcScrollActionSBPageDown Then
        MsgBox("Scrolled down")
    End If
End Sub

```

**Example Code C#**

```

private void vcGantt1_VcComponentScrolling(object sender,
NETRONIC.XGantt.VcComponentScrollingEventArgs e)
{
    if (e.ScrollOrientation == VcScrollOrientation.vcHorizontal && e.ScrollAction
== VcScrollAction.vcScrollActionSBPageLeft)
        MessageBox.Show("Scrolled left");
    else if (e.ScrollOrientation == VcScrollOrientation.vcHorizontal &&
e.ScrollAction == VcScrollAction.vcScrollActionSBPageRight)
        MessageBox.Show("Scrolled right");
    else if (e.ScrollOrientation == VcScrollOrientation.vcVertical &&
e.ScrollAction == VcScrollAction.vcScrollActionSBPageUp)
        MessageBox.Show("Scrolled up");
    else if (e.ScrollOrientation == VcScrollOrientation.vcVertical &&
e.ScrollAction == VcScrollAction.vcScrollActionSBPageDown)
        MessageBox.Show("Scrolled down");
}

```

**VcCurveLeftClicking****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 this. The curve object hit and the position of the mouse (x,y-coordinates) are returned.

	Data Type	Explanation
<b>Parameter:</b>		
⇒ sender	VcGantt	Reference to the object that triggered the event
⇒ e	VcCurveClickingEventArgs	Object specific to the event that is being handled



## Properties of the VcCurveClickingEventArgs object

	Data Type	Explanation
<b>Properties:</b>		
⇒ curve	VcCurve	Curve hit in histogram
⇒ x	System.Int32	X coordinate of the mouse cursor
⇒ y	System.Int32	Y coordinate of the mouse cursor
⇔ returnStatus	VcReturnStatus	Return status
	<b>Possible Values:</b> .vcRetStatFalse 0 .vcRetStatOK 1	The curve will not be marked. The curve will be marked.

## Example Code VB.NET

```
Private Sub VcGantt1_VcCurveLeftClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcCurveClickingEventArgs) Handles VcGantt1.VcCurveLeftClicking
    e.Curve.LineColor = Color.Blue
End Sub
```

## Example Code C#

```
private void vcGantt1_VcCurveLeftClicking(object sender,
NETRONIC.XGantt.VcCurveClickingEventArgs e)
{
    e.Curve.LineColor = Color.LightSteelBlue;
}
```

## VcDataRecordModified

Event of VcGantt

This event occurs when the modification of the box is finished.

	Data Type	Explanation
<b>Parameter:</b>		
⇒ sender	VcGantt	Reference to the object that triggered the event
⇒ e	VcDataRecordModifiedEventArgs	Object specific to the event that is being handled

## Properties of the VcDataRecordModifiedEventArgs object

	Data Type	Explanation
<b>Properties:</b>		
⇒ dataRecord	VcDataRecord	Data record modified

**Example Code VB.NET**

```
Private Sub VcGanttASP1_VcDataRecordModified(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcDataRecordModifiedEventArgs) Handles
VcGanttASP1.VcDataRecordModified
    MsgBox("The data record has been modified")
End Sub
```

**Example Code C#**

```
private void vcGantt1_VcDataRecordModified(object sender,
NETRONIC.XGantt.VcDataRecordModifiedEventArgs e)
{
    MessageBox.Show("The data record has been modified");
}
```

## VcDataRecordModifying

**Event of VcGantt**

This event occurs when the user has modified a box interactively. The modified VcBox 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 **VcDataRecordModified**.

By setting the return status the modification can be inhibited.

	Data Type	Explanation
<b>Parameter:</b>		
⇒ sender	VcGantt	Reference to the object that triggered the event
⇒ e	VcDataRecordModifyingEventArgs	Object specific to the event that is being handled

### Properties of the VcDataRecordModifyingEventArgs object

	Data Type	Explanation
<b>Properties:</b>		
⇒ dataRecord	VcDataRecord	Data record modified
⇒ modificationType	VcModificationTypes	Modification type
	<b>Possible Values:</b>	
	.vcAnything 1	Modification type cannot be identified.
	.vcChangedGroup 16	Group of the node was changed (occurs with nodes only).
	.vcEndModified 4	The end date of the node was modified (occurs with nodes only).
	.vcHierarchyModified 64	Hierarchy of the nodes has been changed
	.vcModifiedByResourceScheduling 128	Modification by resource scheduling (occurs with data records only)

	.vcModifiedBySchedule 32 .vcMoved 8 .vcNothing 0 .vcStartModified 2	Modification by new date calculation Object was moved. No modification The start date of the node was modified (occurs with nodes only).
↔ returnStatus	VcReturnStatus <b>Possible Values:</b> .vcRetStatFalse 0 .vcRetStatOK 1	Return status  The modification will be revoked. The modification will be accepted.

## VcDataRecordNotFound

### 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
<b>Parameter:</b>		
↔ sender	VcGantt	
↔ e	VcDataRecordNotFoundEventArgs	

	Data Type	Explanation
<b>Properties:</b>		
↔ index	System.Int32	Index of the field that contains the key of the depending data record

## VcDiagramHorizontalScrolled

### 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
<b>Parameter:</b>		
⇒ sender	VcGantt	Reference to the object that triggered the event
⇒ e	VcDiagramHorizontalScrolledEventArgs	Object specific to the event that is being handled

### Properties of the VcDiagramHorizontalScrolledEventArgs object

	Data Type	Explanation
<b>Properties:</b>		
⇒ newStartDate	System.DateTime	New start date of the visible part of the diagram
⇒ newEndDate	System.DateTime	New final date of the visible part of the diagram
⇒ scrollAction	VcScrollAction	Scrolling type
	<b>Possible Values:</b>	
	.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 <b>End</b> button or the context menu to the diagram end (right down)

.vcScrollActionScrollHome 103	Scrolling via the <b>Pos 1</b> 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 VB.NET

```
Private Sub VcGantt1_VcDiagramHorizontalScrolled(ByVal sender As Object, ByVal e
As NETRONIC.XGantt.VcDiagramHorizontalScrolledEventArgs) Handles
VcGantt1.VcDiagramHorizontalScrolled
    MsgBox(e.CurStartDate + e.CurEndDate)
End Sub
```

### Example Code C#

```
private void vcGantt1_VcDiagramHorizontalScrolled(object sender,
NETRONIC.XGantt.VcDiagramHorizontalScrolledEventArgs e)
{
    MessageBox.Show(e.CurStartDate.ToString() + "\r\n" +
e.CurEndDate.ToString());
}
```

## VcDiagramHorizontalScrolling

### 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 with your own solution.

	Data Type	Explanation
<b>Parameter:</b>		
⇒ sender	VcGantt	Reference to the object that triggered the event
⇒ e	VcDiagramHorizontalScrollingEventArgs	Object specific to the event that is being handled

### Properties of the VcDiagramHorizontalScrollingEventArgs object

	Data Type	Explanation
<b>Properties:</b>		
⇒ curStartDate	System.DateTime	Current start date of the visible part of the diagram
⇒ curEndDate	System.DateTime	Current end date of the visible part of the diagram
⇒ scrollAction	VcScrollAction	Scrolling type
	<b>Possible Values:</b>	
	.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 <b>End</b> button or the context menu to the diagram end (right down)
	.vcScrollActionScrollHome 103	Scrolling via the <b>Pos 1</b> 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	VcReturnStatus	Return status
	<b>Possible Values:</b>	
	.vcRetStatDefault 2	The default behavior remains unchanged.

## 598 API Reference: VcGantt

.vcRetStatFalse 0	The default behavior will not be performed.
.vcRetStatNoPopup 4	The popup of the context menu is inhibited.
.vcRetStatOK 1	The default behavior will be performed.

### Example Code VB.NET

```
Private Sub VcGantt1_VcDiagramHorizontalScrolling(ByVal sender As Object, ByVal e As NETRONIC.XGantt.VcDiagramHorizontalScrollingEventArgs) Handles VcGantt1.VcDiagramHorizontalScrolling
    If e.CurStartDate > "01.01.2014" Then
        e.ReturnStatus = VcReturnStatus.vcRetStatFalse
    End If
End Sub
```

### Example Code C#

```
private void vcGantt1_VcDiagramHorizontalScrolling(object sender, NETRONIC.XGantt.VcDiagramHorizontalScrollingEventArgs e)
{
    if (DateTime.Compare(e.CurStartDate, Convert.ToDateTime("01.05.14 00:00:00")).Equals(true))
        e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
}
```

## VcDiagramLeftClicking

### 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
<b>Parameter:</b>		
⇒ sender	VcGantt	Reference to the object that triggered the event
⇒ e	VcDiagramClickingEventArgs	Object specific to the event that is being handled

### Properties of the VcDiagramClickingEventArgs object

	Data Type	Explanation
<b>Properties:</b>		
⇒ x	System.Int32	X coordinate of the mouse cursor
⇒ y	System.Int32	Y coordinate of the mouse cursor
↔ returnStatus	VcReturnStatus	Return status
	<b>Possible Values:</b>	
	.vcRetStatDefault 2	The default behavior remains unchanged.
	.vcRetStatFalse 0	The default behavior will not be performed.
	.vcRetStatNoPopup 4	The popup of the context menu is inhibited.

.vcRetStatOK 1	The default behavior will be performed.
----------------	---

**Example Code VB.NET**

```
Private Sub VcGantt1_VcDiagramLeftClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcDiagramClickingEventArgs) Handles
VcGantt1.VcDiagramLeftClicking
    MsgBox("x: " + e.X.ToString() + " y: " + e.Y.ToString())
End Sub
```

**Example Code C#**

```
private void vcGantt1_VcDiagramLeftClicking(object sender,
NETRONIC.XGantt.VcDiagramClickingEventArgs e)
{
    MessageBox.Show("x: " + e.X.ToString() + " y: " + e.Y.ToString());
}
```

## VcErrorOccurring

**Event of VcGantt**

This event occurs when an unexpected error occurs in the code of VARCHART XGantt. NETRONIC tries to avoid errors in its products; if still one occurs, this event will store it to a log file on the customer's computer and will notify the user in a convenient way. The parameter profile is provided by the ActiveX default, so some of the parameters that are passed are constant. The number of the event should always be checked, in order to prevent blocking all error types in the future program development.

	Data Type	Explanation
<b>Parameter:</b>		
⇒ sender	VcGantt	Reference to the object that triggered the event
⇒ e	VcErrorOccurringEventArgs	Object specific to the event that is being handled

### Properties of the VcErrorOccurringEventArgs object

	Data Type	Explanation
<b>Properties:</b>		
↔ ReturnStatus	System.Boolean	If the ReturnStatus is set to <b>vcRetStatFalse</b> , the popping up of the message box is suppressed.
⇒ Text	System.String	Error description



## VcFieldSelecting

### 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
<b>Parameter:</b>		
⇒ sender	VcGantt	Reference to the object that triggered the event
⇒ e	VcFieldSelectingEventArgs	Object specific to the event that is being handled

### Properties of the VcFieldSelectingEventArgs object

	Data Type	Explanation
<b>Properties:</b>		
⇒ editObject	VcObject	Object edited
⇒ editObjectType	VcObjectType	Object type
	<b>Possible Values:</b> .vcObjTypeBox 15 .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 <b>box</b> object type <b>curve</b> object type <b>date line</b> object type <b>group</b> object type <b>group in diagram area</b> object type <b>group in table area</b> object type <b>histogram</b> object type <b>layer</b> object type <b>link collection</b> object type <b>node in diagram area</b> object type <b>node in legend area</b> object type <b>node in table area</b> no object object type <b>numeric scale</b> object type <b>summary bar</b> object type <b>table</b> object type <b>table caption</b> object type <b>time scale</b>
⇒ fieldIndex	System.Int32	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	VcReturnStatus	
	<b>Possible Values:</b> .vcRetStatFalse 0 .vcRetStatOK 1	The field will not be selected. The field will be selected.

## VcGroupLeftClicking

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
<b>Parameter:</b>		
⇒ sender	VcGantt	Reference to the object that triggered the event
⇒ e	VcGroupClickingEventArgs	Object specific to the event that is being handled

### Properties of the VcGroupClickingEventArgs object

	Data Type	Explanation
<b>Properties:</b>		
⇒ group	VcGroup	Group hit
⇒ x	System.Int32	X coordinate of the mouse cursor
⇒ y	System.Int32	Y coordinate of the mouse cursor
↔ returnStatus	VcReturnStatus	Return status
	<b>Possible Values:</b> .vcRetStatDefault 2 .vcRetStatFalse 0 .vcRetStatNoPopup 4 .vcRetStatOK 1	The default behavior remains unchanged. The default behavior will not be performed. The popup of the context menu is inhibited. The default behavior will be performed.

#### Example Code VB.NET

```
Private Sub VcGantt1_VcGroupLeftClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcGroupClickingEventArgs) Handles VcGantt1.VcGroupLeftClicking
    MsgBox(e.Group.SubGroups.Count)
End Sub
```

#### Example Code C#

```
private void vcGantt1_VcGroupLeftClicking(object sender,
NETRONIC.XGantt.VcGroupClickingEventArgs e)
{
    MessageBox.Show(e.Group.SubGroups.Count.ToString());
}
```

## VcGroupModified

Event of VcGantt

This event occurs when the modification of the group is finished.

	Data Type	Explanation
<b>Parameter:</b>		
⇒ sender	VcGantt	Reference to the object that triggered the event
⇒ e	VcGroupModifiedEventArgs	Object specific to the event that is being handled

### Properties of the VcGroupModifiedEventArgs object

	Data Type	Explanation
<b>Properties:</b>		
⇒ group	VcGroup	Group modified

#### Example Code VB.NET

```
Private Sub VcGantt1_VcGroupModified(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcGroupModifiedEventArgs) Handles VcGantt1.VcGroupModified
    MsgBox("The group has been modified.")
End Sub
```

#### Example Code C#

```
private void vcGantt1_VcGroupModified(object sender,
NETRONIC.XGantt.VcGroupModifiedEventArgs e)
{
    MessageBox.Show("The group has been modified");
}
```

## VcGroupModifying

### 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. By the **modificationType** parameter you can obtain more detailed information of the type of modification.

The data passed by this event can be read, but must not be modified. For modifying them please use the event **VcGroupModified**.

By setting the return status the modification can be inhibited.

	Data Type	Explanation
<b>Parameter:</b>		
⇒ sender	VcGantt	Reference to the object that triggered the event
⇒ e	VcGroupModifyingEventArgs	Object specific to the event that is being handled

## Properties of the VcGroupModifyingEventArgs object

	Data Type	Explanation
<b>Properties:</b>		
⇔ Rückgabewert	Void	
⇒ oldGroup	VcGoup	Group before the modification
⇒ group	VcGroup	Group modified
⇒ group	VcGroup	Group to be modified
⇒ modificationType	VcGroupModificationTypes	Type of modification
	<b>Possible Values:</b> .vcGMTAnything 1 .vcGMTMinusPressed 2  .vcGMTNothing 0 .vcGMTPlusPressed 4	Modification type <b>not determined</b> Modification type <b>Minus symbol clicked on</b> Modification type <b>nothing</b> Modification type <b>Plus symbol clicked on</b>
⇒ modificationType	VcModificationTypes	Type of modification
	<b>Possible Values:</b> .vcAnything 1 .vcChangedGroup 16  .vcEndModified 4  .vcHierarchyModified 64  .vcModifiedByResourceScheduling 128  .vcModifiedBySchedule 32 .vcMoved 8 .vcNothing 0 .vcStartModified 2	Modification type cannot be identified. Group of the node was changed (occurs with nodes only). The end date of the node was modified (occurs with nodes only). Hierarchy of the nodes has been 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 was modified (occurs with nodes only).
⇔ returnStatus	VcReturnStatus	Return status
	<b>Possible Values:</b> .vcRetStatFalse 0 .vcRetStatOK 1	The modification will be revoked. The modification will be accepted.
⇔ returnStatus	VcReturnStatus	Return status
	<b>Possible Values:</b> .vcRetStatDefault 2  .vcRetStatFalse 0  .vcRetStatNoPopup 4  .vcRetStatOK 1	The default behavior remains unchanged. The default behavior will not be performed. The popup of the context menu is inhibited. The default behavior will be performed.

### Example Code VB.NET

```
Private Sub VcGantt1_VcGroupModifying(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcGroupModifyingEventArgs) Handles VcGantt1.VcGroupModifying
    Select Case e.ModificationType
        Case VcGroupModificationTypes.vcGMTNothing
            MsgBox("No modification")
        Case VcGroupModificationTypes.vcGMTAnything
            MsgBox("Any modification")
        Case VcGroupModificationTypes.vcGMTMinusPressed
            MsgBox("Collapsing group:" + e.Group.Name)
        Case VcGroupModificationTypes.vcGMTPlusPressed
            MsgBox("Expanding group" + e.Group.Name)
    End Select
End Sub
```

### Example Code C#

```
private void vcGantt1_VcGroupModifying(object sender,
NETRONIC.XGantt.VcGroupModifyingEventArgs e)
{
    switch (e.ModificationType)
    {
        case VcGroupModificationTypes.vcGMTNothing:
            MessageBox.Show("No modification");
            break;
        case VcGroupModificationTypes.vcGMTAnything:
            MessageBox.Show("Any modification");
            break;
        case VcGroupModificationTypes.vcGMTMinusPressed:
            MessageBox.Show("Collapsing group: " + e.Group.Name);
            break;
        case VcGroupModificationTypes.vcGMTPlusPressed:
            MessageBox.Show("Expanding group: " + e.Group.Name);
            break;
    }
}
```

## VcGroupsMarked

**Event of VcGantt**

This event occurs after the operation of marking or unmarking groups was finished.

	Data Type	Explanation
<b>Parameter:</b>		
⇒ sender	VcGantt	Reference to the object that triggered the event
⇒ e	VcGroupsMarkedEventArgs	Object specific to the event that is being handled

### Properties of the VcGroupsMarkedEventArgs object

	Data Type	Explanation
<b>Properties:</b>		
⇐ (no parameter)		No parameter

**Example Code VB.NET**

```
Private Sub VcGanttASP1_VcGroupsMarked(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcGroupsMarkedEventArgs) Handles VcGantt1.VcGroupsMarked
    MsgBox("Groups have been marked successfully.")
End Sub
```

**Example Code C#**

```
private void vcGanttASP1_VcGroupsMarked(object sender,
NETRONIC.XGantt.VcGroupsMarkedEventArgs e)
{
    MessageBox.Show("Groups have been marked successfully.");
}
```

## VcGroupsMarking

**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. The GroupCollection contains the groups selected by the most recent marking action of the user. 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 **VcGroupsMarked**.

	Data Type	Explanation
<b>Parameter:</b>		
⇒ sender	VcGantt	Reference to the object that triggered the event
⇒ e	VcNodesMarkingEventArgs	Object specific to the event that is being handled

### Properties of the VcNodesMarkingEventArgs object

	Data Type	Explanation
<b>Properties:</b>		
⇒ groupCollection	VcGroupCollection	GroupCollection that contains the nodes selected by the user. If the user has clicked in the diagram, the GroupCollection is empty.
↔ returnStatus	VcReturnStatus	Return status

**Example Code VB.NET**

```
Private Sub VcGantt1_VcGroupsMarking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcGroupsMarkingEventArgs) Handles VcGantt1.VcGroupsMarking
    If MsgBox("Mark this group?", MsgBoxStyle.YesNo, "Marking groups") =
MsgBoxResult.No Then
        e.ReturnStatus = VcReturnStatus.vcRetStatFalse
    End If
End Sub
```

**Example Code C#**

```
private void vcGantt1_VcGroupsMarking(object sender,
NETRONIC.XGantt.VcGroupsMarkingEventArgs e)
{
    DialogResult retVal = MessageBox.Show("Mark this group?", "Marking groups",
MessageBoxButtons.YesNo);
    if (retVal == DialogResult.No)
        e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
}
```

## VcHistogramLeftClicking

**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
<b>Parameter:</b>		
⇒ sender	VcGantt	Reference to the object that triggered the event
⇒ e	VcHistogramClickingEventArgs	Object specific to the event that is being handled

### Properties of the VcHistogramClickingEventArgs object

	Data Type	Explanation
<b>Properties:</b>		
⇒ histogram	VcHistogram	Histogram hit
⇒ x	System.Int32	X coordinate of the mouse cursor
⇒ y	System.Int32	Y coordinate of the mouse cursor
↔ returnStatus	VcReturnStatus	Return status
	<b>Possible Values:</b> .vcRetStatDefault 2 .vcRetStatFalse 0 .vcRetStatNoPopup 4 .vcRetStatOK 1	The default behavior remains unchanged. The default behavior will not be performed. The popup of the context menu is inhibited. The default behavior will be performed.

**Example Code VB.NET**

```
Private Sub VcGantt1_VcHistogramLeftClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcHistogramClickingEventArgs) Handles
VcGantt1.VcHistogramLeftClicking
    Call MsgBox("Histogram:" + e.Histogram.Name + " x:" + e.X.ToString() + " y: "
+ e.Y.ToString())
End Sub
```

**Example Code C#**

```
private void vcGantt1_VcHistogramLeftClicking(object sender,
NETRONIC.XGantt.VcHistogramClickingEventArgs e)
{
    MessageBox.Show("Histogram: " + e.Histogram.Name + " x: " + e.X.ToString() +
" y: " + e.Y.ToString());
}
```

## VcHistogramsHeightChanged

**Event of VcGantt**

This event occurs after the ratio of the diagram height to the histogram height modified by the user was changed. The collection of the histograms and the diagram / histogram height ratio are returned.

	Data Type	Explanation
<b>Parameter:</b>		
⇒ sender	VcGantt	Reference to the object that triggered the event
⇒ e	VcHistogramsHeightChangedEventArgs	Object specific to the event that is being handled

### Properties of the VcHistogramsHeightChangedEventArgs object

	Data Type	Explanation
<b>Properties:</b>		
⇒ histogramCollection	VcHistogramCollection	Histogram collection
⇒ histogramsHeightRatio	System.Int32	Height ratio of the histograms to the complete diagram

## VcHistogramsHeightChanging

**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 `vcRetStatFalse`, the modification will be revoked.

	Data Type	Explanation
<b>Parameter:</b>		
⇒ sender	VcGantt	Reference to the object that triggered the event
⇒ e	VcHistogramsHeightChangingEventArgs	Object specific to the event that is being handled

### Properties of the VcHistogramsHeightChangingEventArgs object

	Data Type	Explanation
<b>Properties:</b>		
⇒ histogramCollection	VcHistogramCollection	Histogram collection
⇒ histogramsHeightRatio	System.Int32	Height ratio of the histograms to the complete diagram
↔ returnStatus	VcReturnStatus	Return status
	<b>Possible Values:</b> .vcRetStatFalse 0 .vcRetStatOK 1	The height will not change. The height will change.

## VcLinksLeftClicking

### 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
<b>Parameter:</b>		
⇒ sender	VcGantt	Reference to the object that triggered the event
⇒ e	VcLinksClickingEventArgs	Object specific to the event that is being handled

### Properties of the VcLinksClickingEventArgs object

	Data Type	Explanation
<b>Properties:</b>		
⇒ linkCltn	VcLinkCollection	LinkCollection object hit
⇒ x	System.Int32	Y coordinate of the mouse cursor
⇒ y	System.Int32	Y coordinate of the mouse cursor
⇔ returnStatus	VcReturnStatus	Return status
	<b>Possible Values:</b> .vcRetStatDefault 2 .vcRetStatFalse 0 .vcRetStatNoPopup 4 .vcRetStatOK 1	The default behavior remains unchanged. The default behavior will not be performed. The popup of the context menu is inhibited. The default behavior will be performed.

### Example Code VB.NET

```
Private Sub VcGantt1_VcLinksLeftClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcLinksClickingEventArgs) Handles VcGantt1.VcLinksLeftClicking
    Dim linkCltn As VcLinkCollection
    Dim link As VcLink
    linkCltn = VcGantt1.LinkCollection
    'set certain data field of all links
    For Each link In linkCltn
        link.DataField(2) = "A"
    Next
End Sub
```

### Example Code C#

```
private void vcGantt1_VcLinksLeftClicking(object sender,
NETRONIC.XGantt.VcLinksClickingEventArgs e)
{
    VcLinkCollection linkCltn = vcGantt1.LinkCollection;
    // set certain data field of all links
    foreach (VcLink link in linkCltn)
        link.set_DataField(2, "A");
}
```

## VcNodeLeftClicking

### 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 as parameters.

	Data Type	Explanation
<b>Parameter:</b>		
⇒ sender	VcGantt	Reference to the object that triggered the event
⇒ e	VcNodeClickingEventArgs	Object specific to the event that is being handled

## Properties of the VcNodeClickingEventArgs object

	Data Type	Explanation
<b>Properties:</b>		
⇒ node	VcNode	Node hit
⇒ location	VcLocation	Location in the diagram
	<b>Possible Values:</b>	
	.vcInDiagram 1	Located in the node area
	.vcInTable 0	Located in the table area
⇒ x	System.Int32	X coordinate of the mouse cursor
⇒ y	System.Int32	Y coordinate of the mouse cursor
↔ returnStatus	VcReturnStatus	Return status
	<b>Possible Values:</b>	
	.vcRetStatDefault 2	The default behavior remains unchanged.
	.vcRetStatFalse 0	The default behavior will not be performed.
	.vcRetStatNoPopup 4	The popup of the context menu is inhibited.
	.vcRetStatOK 1	The default behavior will be performed.

## Example Code VB.NET

```
Private Sub VcGanttASP1_VcNodeLeftClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcNodeClickingEventArgs) Handles VcGantt1.VcNodeLeftClicking
    'change data field of the node
    e.Node.DataField(4) = 1 - Convert.ToInt64(e.Node.DataField(4))
End Sub
```

## Example Code C#

```
private void vcGanttASP1_VcNodeLeftClicking(object sender,
NETRONIC.XGantt.VcNodeClickingEventArgs e)
{
    //change data field of the node
    e.Node.set_DataField(4, Convert.ToInt64(e.Node.get_DataField(4)));
}
```

## VcNodeModified

Event of VcGantt

This event occurs when the modification of the node specified is completed.

	Data Type	Explanation
<b>Parameter:</b>		
⇒ sender	VcGantt	Reference to the object that triggered the event
⇒ e	VcNodeModifiedEventArgs	Object specific to the event that is being handled

## Properties of the VcNodeModifiedEventArgs object

	Data Type	Explanation
<b>Properties:</b>		
⇒ node	VcNode	Node modified
⇒ isLast	System.Boolean	The modified node is/is not the only node or the last node of a node collection.

### Example Code VB.NET

```
Private Sub VcGanttASP1_VcNodeModifying(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcNodeModifyingEventArgs) Handles VcGanttASP1.VcNodeModifying
    'revoke the modification if the node would change the group
    If e.ModificationType And VcModificationTypes.vcChangedGroup Then
        MsgBox("The node cannot be moved to a different group.")
    End If
End Sub
```

### Example Code C#

```
private void vcGanttASP1_VcNodeModifying(object sender,
NETRONIC.XGantt.VcNodeModifyingEventArgs e)
{
    //revoke the modification if the node would change the group
    if (VcModificationTypes.vcChangedGroup.Equals(true))
    {
        MessageBox.Show("The node cannot be moved into another group.");
    }
}
```

## VcNodeModifiedEx

Event of VcGantt

This event occurs when the modification of the marked node was completed.

	Data Type	Explanation
<b>Parameter:</b>		
⇒ sender	VcGantt	Reference to the object that triggered the event
⇒ e	VcNodeModifiedExEventArgs	Object specific to the event that is being handled

Properties of the VcNodeModifiedExEventArgs object

	Data Type	Explanation
<b>Properties:</b>		
⇒ node	VcNode	Node modified
⇒ isLast	System.Boolean	The modified node is/is not the only node or the last node of a node collection.
⇒ modificationType	VcModificationTypes	Modification type
	<b>Possible Values:</b>	

## 612 API Reference: VcGantt

.vcAnything 1	Modification type cannot be identified.
.vcChangedGroup 16	Group of the node was changed (occurs with nodes only).
.vcEndModified 4	The end date of the node was modified (occurs with nodes only).
.vcHierarchyModified 64	Hierarchy of the nodes has been 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 was modified (occurs with nodes only).

### Example Code VB.NET

```
Private Sub VcGanttASPl_VcNodeModifiedEx(ByVal sender As System.Object, _
    ByVal e As
NETRONIC.XGantt.VcNodeModifiedExEventArgs)
    Handles VcGanttASPl.VcNodeModifiedEx
    'modify a record in the underlying database of the application
    modifyDataRecord(e.Node.AllData)
End Sub
```

### Example Code C#

```
private void vcGanttASPl_VcNodeModifiedEx(object sender,
VcNodeModifiedExEventArgs e)
{
    //modify a record in the underlying database of the application
    modifyDataRecord(e.Node.AllData);
}
```

## VcNodeModifying

### 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 return status to **vcRetStatFalse**, the modification can be inhibited.

The data passed by this event can be read, but must not be modified. For modifying them please use the event **VcNodeModified**.

	Data Type	Explanation
<b>Parameter:</b>		
⇒ sender	VcGantt	Reference to the object that triggered the event
⇒ e	VcNodeModifyingEventArgs	Object specific to the event that is being handled

## Properties of the VcNodeModifyingEventArgs object

	Data Type	Explanation
<b>Properties:</b>		
⇒ oldNode	VcNode	Node before the modification
⇒ node	VcNode	Node to be modified
⇒ modificationType	VcModificationTypes	Type of modification  (A combination of the values is also possible.)
	<b>Possible Values:</b> .vcAnything 1 .vcChangedGroup 16  .vcEndModified 4  .vcHierarchyModified 64  .vcModifiedByResourceScheduling 128  .vcModifiedBySchedule 32 .vcMoved 8 .vcNothing 0 .vcStartModified 2	Modification type cannot be identified. Group of the node was changed (occurs with nodes only). The end date of the node was modified (occurs with nodes only). Hierarchy of the nodes has been 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 was modified (occurs with nodes only).
⇔ returnStatus	VcReturnStatus	Return status
	<b>Possible Values:</b> .vcRetStatFalse 0 .vcRetStatOK 1	The modification will be revoked. The modification will be accepted.

## Example Code VB.NET

```
Private Sub VcGanttASP1_VcNodeModifying(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcNodeModifyingEventArgs) Handles VcGanttASP1.VcNodeModifying
    ' revoke the modification if the node would change the group
    If e.ModificationType And VcModificationTypes.vcChangedGroup Then
        MsgBox("The node cannot be moved to a different group.")
        e.ReturnStatus = VcReturnStatus.vcRetStatFalse
    End If
End Sub
```

## Example Code C#

```
private void vcGanttASP1_VcNodeModifying(object sender,
NETRONIC.XGantt.VcNodeModifyingEventArgs e)
{
    //revoke the modification if the node would change the group
    if (e.ModificationType == VcModificationTypes.vcChangedGroup)
    {
        MessageBox.Show("The node cannot be moved into another group.");
        e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
    }
}
```

## VcNodesMarked

Event of VcGantt

This event occurs after the operation of marking or unmarking nodes was finished.

	Data Type	Explanation
<b>Parameter:</b>		
⇒ sender	VcGantt	Reference to the object that triggered the event
⇒ e	VcNodesMarkedEventArgs	Object specific to the event that is being handled

### Properties of the VcNodesMarkedEventArgs object

	Data Type	Explanation
<b>Properties:</b>		
⇐ (no parameter)		No parameter

#### Example Code VB.NET

```
Private Sub VcGanttASP1_VcNodesMarked(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcNodesMarkedEventArgs) Handles VcGantt1.VcNodesMarked
    MsgBox("Nodes have been marked successfully.")
End Sub
```

#### Example Code C#

```
private void vcGanttASP1_VcNodesMarked(object sender,
NETRONIC.XGantt.VcNodesMarkedEventArgs e)
{
    MessageBox.Show("Nodes have been marked successfully.");
}
```

## VcNodesMarking

Event of VcGantt

This event occurs when the user selects nodes for marking or when he unmarks marked nodes by a click into the empty 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 the empty diagram, the node collection will be empty.

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 **VcNodesMarked**.

	Data Type	Explanation
<b>Parameter:</b>		
⇒ sender	VcGantt	Reference to the object that triggered the event
⇒ e	VcNodesMarkingEventArgs	Object specific to the event that is being handled

### Properties of the VcNodesMarkingEventArgs object

	Data Type	Explanation
<b>Properties:</b>		
⇒ nodeCollection	VcNodeCollection	NodeCollection that contains the nodes selected by the user. If the user has clicked in the diagram, the NodeCollection is empty.
↔ returnStatus	VcReturnStatus	Return status
	<b>Possible Values:</b> .vcRetStatFalse 0 .vcRetStatOK 1	Marking has to be done manually. Marking is done automatically.

#### Example Code VB.NET

```
Private Sub VcGantt1_VcNodesMarking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcNodesMarkingEventArgs) Handles VcGantt1.VcNodesMarking
    If MsgBox("Mark this node?", MsgBoxStyle.YesNo, "Marking nodes") =
MsgBoxResult.No Then
        e.ReturnStatus = VcReturnStatus.vcRetStatFalse
    End If
End Sub
```

#### Example Code C#

```
private void vcGantt1_VcNodesMarking(object sender,
NETRONIC.XGantt.VcNodesMarkingEventArgs e)
{
    DialogResult retVal = MessageBox.Show("Mark this node?", "Marking nodes",
MessageBoxButtons.YesNo);
    if (retVal == DialogResult.No)
        e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
}
```

## VcNumericScaleLeftClicking

### 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
<b>Parameter:</b>		
⇒ sender	VcGantt	Reference to the object that triggered the event
⇒ e	VcNumericScaleClickingEventArgs	Object specific to the event that is being handled

### Properties of the VcNumericScaleClickingEventArgs object

	Data Type	Explanation
<b>Properties:</b>		
⇒ numericScale	VcNumericScale	Numeric scale hit
⇒ x	System.Int32	X coordinate of the mouse cursor
⇒ y	System.Int32	Y coordinate of the mouse cursor
↔ returnStatus	VcReturnStatus	Return status
	<b>Possible Values:</b> .vcRetStatDefault 2 .vcRetStatFalse 0 .vcRetStatNoPopup 4 .vcRetStatOK 1	The default behavior remains unchanged. The default behavior will not be performed. The popup of the context menu is inhibited. The default behavior will be performed.

#### Example Code VB.NET

```
Private Sub VcGantt1_VcNumericScaleLeftClicking(ByVal sender As Object, ByVal e
As NETRONIC.XGantt.VcNumericScaleClickingEventArgs) Handles
VcGantt1.VcNumericScaleLeftClicking
    e.NumericScale.BackgroundColor = Color.Blue
End Sub
```

#### Example Code C#

```
private void vcGantt1_VcNumericScaleLeftClicking(object sender,
NETRONIC.XGantt.VcNumericScaleClickingEventArgs e)
{
    e.NumericScale.BackgroundColor = Color.LightSteelBlue;
}
```

## VcObjectDrawing

### 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. You can finally prevent to have the object drawn by the component by setting the return status to **vcRetStatFalse**.

ObjectDrawing events are only triggered after the corresponding option was set to an object type. At the moment, the option is only available to the object type of layers. So at run time, the property **ObjectDrawEventsEnabled** of the object **VcLayer** can be set to **True**, or alternatively, at design time, on the **Objects** property page you can select **Layers** and set the option **ObjectDraw Events** for the layer being edited.

If you wish to add something to the layer that was drawn by VARCHART XGantt, you can do this by the event **VcObjectDrawn**.

	Data Type	Explanation
<b>Parameter:</b>		
⇒ sender	VcGantt	Reference to the object that triggered the event
⇒ e	VcObjectDrawingEventArgs	Object specific to the event that is being handled

### Properties of the VcObjectDrawingEventArgs object

	Data Type	Explanation
<b>Properties:</b>		
Graphics	System.Drawing.Graphics	Device context
ObjectToDraw	Object	Object to be drawn
ObjectType	VcObjectType	Type of object to be drawn
	<b>Possible Values:</b>	
	.vcObjTypeNodeInDiagram 2	object type <b>node in diagram area</b>
	.vcObjTypeNodeInLegend 17	object type <b>node in legend area</b>
	.vcObjTypeSummaryNode 14	object type <b>summary bar</b>
SubObject	Object	Subobject that is passed context-dependent
SubObjectType	VcObjectType	Type of subobject
	<b>Possible Values:</b>	
	.vcObjTypeLayer 8	object type <b>layer</b>
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.
ReturnStatus	VcReturnStatus	Return status
	<b>Possible Values:</b>	
	.vcRetStatFalse 0	The object will not be drawn.
	.vcRetStatOK 1	The object will be drawn.
LineWidth	System.Int32	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	System.Int16	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	System.Int16	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).

## VcObjectDrawn

### 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.

ObjectDrawing events are only triggered after the corresponding option was set to an object type. At the moment, the option is only available to the object type of layers. So at run time, the property **ObjectDrawEventsEnabled** of the object **VcLayer** can be set to **True**, or alternatively, at design time, on the **Objects** property page you can select **Layers** and set the option **ObjectDraw Events** for the layer being edited.

If you wish to suppress default drawing of layers and to replace it by programming code of your own, please use the event **VcObjectDrawing**.

	Data Type	Explanation
<b>Parameter:</b>		
⇒ sender	VcGantt	Reference to the object that triggered the event
⇒ e	VcObjectDrawnEventArgs	Object specific to the event that is being handled

### Properties of the VcObjectDrawnEventArgs object

	Data Type	Explanation
<b>Properties:</b>		
Graphics	System.Drawing.Graphics	Device context
ObjectType	VcObjectType	Type of object drawn
	<b>Possible Values:</b> .vcObjTypeNodeInDiagram 2	object type <b>node in diagram area</b>

	.vcObjTypeNodeInLegend 17	object type <b>node in legend area</b>
	.vcObjTypeSummaryNode 14	object type <b>summary bar</b>
SubObject	Object	Subobject that was passed context-dependent
SubObjectType	VcObjectType	Type of subobject
	<b>Possible Values:</b>	
	.vcObjTypeLayer 8	object type <b>layer</b>
CompleteRect	VcRect	Rectangle in device coordinates into which the complete object was 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	System.Int32	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	System.Int16	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	System.Int16	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).

## VcResourceSchedulingProgressing

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
<b>Parameter:</b>		
⇒ sender	VcGantt	Reference to the object that triggered the event
⇒ e	VcResourceSchedulingProgressingEventArgs	Object specific to the event that is being handled

Properties of the VcResourceSchedulingProgressingEventArgs object

	Data Type	Explanation
<b>Properties:</b>		
⇒ ScheduledJobCount	System.Int32	Number of scheduled jobs
⇒ TotalJobCount	System.Int32	Total number of jobs
⇐ ReturnStatus	System.Object	Return status  <b>vcRetStatFalse:</b> scheduling is abandoned  <b>vcRetStatDefault:</b> scheduling is continued

## VcResourceSchedulingWarning

Event of VcGantt

This event is triggered if the resource scheduling procedure finds inconsistencies in the data records (see method **process** in the object VcResourceScheduler2). This event detects certain errors in the data definition. You can cancel the scheduling procedure by setting the return status.

	Data Type	Explanation
<b>Properties:</b>		
⇐ WarningType	VcResSchedWarningType	Warning type
	<b>Possible Values:</b>	
	.vcResSched-AssignmentLoadPerItemsZero 23	In the assignment data set specified the content of the data field <b>LoadOrConsumptionPerItem</b> is evaluated to be 0. This leads to the assignment being ignored during scheduling.
	.vcResSched-AssignmentNoOperationID 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.
	.vcResSched-AssignmentNoResourceID 1	In the assignment data record also passed the data field of the ID of the resources data record is empty. Because of this, the assignment will be ignored in the ongoing procedure.
	.vcResSched-AssignmentNoDataRecords 0	No assignment data records exist; the parameter <b>DataRecord</b> is <b>null</b> .
	.vcResSched-AssignmentNoResourceID 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.
	.vcResSched-AssignmentOperationNotFound 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.
	.vcResSched-AssignmentTimingResourceMultiple 5	The assignment data record also passed represents a prohibited second or other assignment of an operation to a resource of the type <b>vcResSched-Timing</b> . Because of this, the assignment will be ignored in the ongoing procedure.

.vcResSchedOperation-LoadPerItemsZero 24	In the operation data set specified the content of the data field <b>LoadPerItem</b> 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 the 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-WorkInProgressOutOf-Range 20	This warning occurs if the quantity 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 <b>ResourceCalendarNameFieldIndex</b> 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 resource 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-InputCurveNot-Found 11	The input curve of the resource data record also passed was not found. Input curves for resources of the type <b>vcResSchedTiming</b> and <b>vcResSchedWork</b> are capacity curves; for resources of the resource type <b>vcResSchedMaterial</b> they are supply curves.
.vcResSchedResource-InputCurves-CompletelyZero 12	The values of the input curve of the resource data record also passed are all zero. Input curves for resources of the type <b>vcResSchedTiming</b> and <b>vcResSchedWork</b> are capacity curves; for resources of the resource type <b>vcResSchedMaterial</b> they are supply curves.
.vcResSchedResource-OutputCurveNot-Found 13	The output curve of the resource data record also passed was not found. Output curves for resources of the type <b>vcResSchedTiming</b> and <b>vcResSchedWork</b> are workload curves; for resources of the resource type <b>vcResSchedMaterial</b> 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 <b>vcSetCurve</b> (please see method <b>CurveType</b> of the object <b>VcCurve</b> ). Output curves for resources of the type <b>vcResSchedTiming</b> and <b>vcResSchedWork</b> are workload curves; for resources of the resource type <b>vcResSchedMaterial</b> they are stock curves.
.vcResSchedTask-CapacityBeyond-Limit 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 <b>LoadPerItem</b> in the operation and, if necessary, an efficiency factor in the resource to be allocated. The current limit is 100000.

	.vcResSchedTaskDue-DateEarlierThan-ReleaseDate 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.
	.vcResSchedTaskDue-DateEqualToRelease-Date 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.
	.vcResSchedTaskDue-DateOutOfRange 17	This warning occurs if the due date of a task is not between the PlanningStartDate and the Planning-EndDate 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.
	.vcResSchedTask-QuantityIsZero 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.
	.vcResSchedTask-ReleaseDateOutOf-Range 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	VcReturnStatus	Return status
	<b>Possible Values:</b> .vcRetStatFalse 0 .vcRetStatOK 1	<b>vcRetStatFalse:</b> scheduling is abandoned <b>vcRetStatDefault:</b> scheduling is continued  Resource scheduling will be cancelled. Resource scheduling will be continued.

## VcTableCaptionLeftClicking

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 hierarchically sorted, the activities will be sorted according to the table column hit.

	Data Type	Explanation
<b>Parameter:</b>		
⇒ sender	VcGantt	Reference to the object that triggered the event
⇒ e	VcTableCaptionClickingEventArgs	Object specific to the event that is being handled

Properties of the VcTableCaptionClickingEventArgs object

	Data Type	Explanation
<b>Properties:</b>		
⇒ table	VcTable	Table hit
⇒ columnNumber	System.Int32	Index of the table column hit
⇒ x	System.Int32	X coordinate of the mouse cursor
⇒ y	System.Int32	Y coordinate of the mouse cursor
⇔ returnStatus	VcReturnStatus	Return staus
	<b>Possible Values:</b>	
	.vcRetStatDefault 2	The default behavior remains unchanged.
	.vcRetStatFalse 0	The default behavior will not be performed.
	.vcRetStatNoPopup 4	The popup of the context menu is inhibited.
	.vcRetStatOK 1	The default behavior will be performed.

### Example Code VB.NET

```
Private Sub VcGantt1_VcTableCaptionLeftClicking(ByVal sender As Object, ByVal e
As NETRONIC.XGantt.VcTableClickingEventArgs) Handles
VcGantt1.VcTableCaptionLeftClicking
VcGantt1.LeftTable.TableFormatCollection.FirstFormat.FormatField(0).BackColor =
Color.Blue
End Sub
```

### Example Code C#

```
private void vcGantt1_VcTableCaptionLeftClicking(object sender,
NETRONIC.XGantt.VcTableClickingEventArgs e)
{
vcGantt1.LeftTable.TableFormatCollection.FirstFormat().get_FormatField(0).BackCo
lor = Color.LightSteelBlue;
}
```

## VcTableColumnWidthChanging

### 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
<b>Parameter:</b>		
⇒ sender	VcGantt	Reference to the object that triggered the event
⇒ e	VcTableColumnWidthChangingEventArgs	Object specific to the event that is being handled

Properties of the VcTableColumnWidthChangingEventArgs object



	Data Type	Explanation
<b>Properties:</b>		
⇒ table	VcTable	Table
⇒ index	System.Int16	Index of the column modified
⇒ currentWidth	System.Int32	New column width
⇔ returnStatus	VcReturnStatus	Return status
	<b>Possible Values:</b> .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 VB.NET

```
Private Sub VcGantt1_VcTableColumnWidthChanging(ByVal sender As Object, ByVal e
As NETRONIC.XGantt.VcTableColumnWidthChangingEventArgs) Handles
VcGantt1.VcTableColumnWidthChanging
    If e.CurrentWidth > 5000 Then
        e.ReturnStatus = VcReturnStatus.vcRetStatFalse
        VcGantt1.LeftTable.ColumnWidth(index) = 5000
    End If
End Sub
```

### Example Code C#

```
private void vcGantt1_VcTableColumnWidthChanging(object sender,
NETRONIC.XGantt.VcTableColumnWidthChangingEventArgs e)
{
    if (e.CurrentWidth > 5000)
    {
        e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
        vcGantt1.LeftTable.set_ColumnWidth(1,5000);
    }
}
```

## VcTableWidthChanging

### 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
<b>Parameter:</b>		
⇒ sender	VcGantt	Reference to the object that triggered the event
⇒ e	VcTableWidthChangingEventArgs	Object specific to the event that is being handled

Properties of the VcTableWidthChangingEventArgs object

	Data Type	Explanation
<b>Properties:</b>		
⇒ table	VcTable	Table
⇒ tableWidthRatio	System.Int32	Ratio of the table width to the Width of the the total diagram (including table)
⇔ returnStatus	VcReturnStatus	Return status
	<b>Possible Values:</b> .vcRetStatFalse 0 .vcRetStatOK 1	The width of the table will not be modified. The width of the table will be modified.

### Example Code VB.NET

```
Private Sub VcGantt1_VcTableWidthChanging(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcTableWidthChangingEventArgs) Handles
VcGantt1.VcTableWidthChanging
    If e.TableDiagramWidthRatio > 30 Then
        e.ReturnStatus = VcReturnStatus.vcRetStatFalse
        VcGantt1.LeftTableDiagramWidthRatio = 30
    End If
End Sub
```

### Example Code C#

```
private void vcGantt1_VcTableWidthChanging(object sender,
NETRONIC.XGantt.VcTableWidthChangingEventArgs e)
{
    if (e.TableDiagramWidthRatio > 30)
    {
        e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
        vcGantt1.LeftTableDiagramWidthRatio = 30;
    }
}
```

## VcTextEntrySupplying

### Event of VcGantt

This event only occurs when the VcGantt property **TextEntrySupplying-EventEnabled** 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
<b>Parameter:</b>		
⇒ sender	VcGantt	Reference to the object that triggered the event
⇒ e	VcTextEntrySupplyingEventArgs	Object specific to the event that is being handled

Properties of the VcTextEntrySupplyingEventArgs object

	Data Type	Explanation
<b>Properties:</b>		
⇒ controllIndex	VcTextEntryIndex	Text constant the contents of which is to be replaced
	<b>Possible Values:</b>	
	.vcTXECtxmenReOptimizeNodesInGroup 2136	Text in context menu: <b>Re-optimize nodes</b>
	.vcTXEDlgLegArrangement 2046	Text in the <b>Legend Attributes</b> dialog: <b>Arrangement</b>
	.vcTXEDlgLegBottomMargin 2052	Text in the <b>Legend Attributes</b> dialog: <b>Bottom margin:</b>
	.vcTXEDlgLegFixedToColumns 2048	Text in the <b>Legend Attributes</b> dialog: <b>Fixed to columns</b>
	.vcTXEDlgLegFixedToRows 2047	Text in the <b>Legend Attributes</b> dialog: <b>Fixed to rows</b>
	.vcTXEDlgLegFixedToRowsAndColumns 2049	Text in the <b>Legend Attributes</b> dialog: <b>Fixed to rows and columns</b>
	.vcTXEDlgLegIdcancel 2042	<b>Legend Attributes</b> dialog: <b>Cancel</b> button
	.vcTXEDlgLegIdd 2040	Dialog <b>Legend Attributes</b> : Text in Title Bar
	.vcTXEDlgLegIdok 2041	Button text in <b>Legend Attributes</b> dialog: <b>OK</b>
	.vcTXEDlgLegLegendElements 2045	Text in the <b>Legend Attributes</b> dialog: <b>Legendelements</b>
	.vcTXEDlgLegLegendFont 2053	<b>Legend Attributes</b> dialog: legend <b>Font...</b> button
	.vcTXEDlgLegLegendTitleFont 2044	<b>Legend Attributes</b> dialog: legend title <b>Font</b> button...
	.vcTXEDlgLegLegendTitleVisible 2043	Text in the <b>Legend Attributes</b> dialog: <b>Legend title visible</b>
	.vcTXEDlgLegMargins 2050	Text in the <b>Legend Attributes</b> dialog: <b>Margins</b>
	.vcTXEDlgLegTopMargin 2051	Text in the <b>Legend Attributes</b> dialog: <b>Top margin:</b>
	.vcTXEDlgNedTTGotoFirst 2032	<b>Edit data</b> dialog: tooltip text <b>Show first selected activity</b>
	.vcTXEDlgNedTTGotoLast 2035	<b>Edit data</b> dialog, Tooltip <b>"Show last selected activity"</b>
	.vcTXEDlgNedTTGotoNext 2034	<b>Edit data</b> dialog, tooltip text <b>Show next selected activity</b>
	.vcTXEDlgNedTTGotoPrev 2033	<b>Edit data</b> dialog: tooltip text <b>Show previous selected activity</b>
	.vcTXEDlgNedValuesColStr 2019	Text in the <b>Edit data</b> dialog: <b>Values</b>
	.vcTXEDlgTscEndDate 2012	Text in <b>Edit time scale</b> dialog: <b>End Date</b>
	.vcTXEDlgTscIdcancel 2010	<b>Edit time scale</b> dialog: button text <b>Cancel</b>
	.vcTXEDlgTscIdd 2008	<b>Edit time scale</b> dialog: text in title bar
	.vcTXEDlgTscIdok 2009	<b>Edit time scale</b> dialog: button text <b>OK</b>
	.vcTXEDlgTscScale 2013	Text in <b>Edit time scale</b> dialog: <b>Scale</b>
	.vcTXEDlgTscStartDate 2011	Text in time scale editor dialog: <b>Start Date</b>
	.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."
	.vcTXEErrTxtStartNotLaterThanPrevSect 2733	Message text: "Start date ""%s"" is not later than start date of previous section.\n\nThe old date will be inserted again."
	.vcTXEPrctMtWrongCharacter 2405	Message text: <b>Invalid character</b>
	.vcTXERibAM 2225	ribbon text for <b>am</b>
	.vcTXERibCW 2223	ribbon text for <b>calendar week</b>
	.vcTXERibDay0 2212	ribbon text for <b>Monday</b>
	.vcTXERibDay1 2213	ribbon text for <b>Tuesday</b>
	.vcTXERibDay2 2214	ribbon text for <b>Wednesday</b>
	.vcTXERibDay3 2215	ribbon text for <b>Thursday</b>
	.vcTXERibDay4 2216	ribbon text for <b>Friday</b>
	.vcTXERibDay5 2217	ribbon text for <b>Saturday</b>
	.vcTXERibDay6 2218	ribbon text for <b>Sunday</b>
	.vcTXERibMon0 2200	ribbon text for <b>January</b>
	.vcTXERibMon1 2201	ribbon text for <b>February</b>
	.vcTXERibMon10 2210	ribbon text for <b>November</b>
	.vcTXERibMon11 2211	ribbon text for <b>December</b>
	.vcTXERibMon2 2202	ribbon text for <b>March</b>
	.vcTXERibMon3 2203	ribbon text for <b>April</b>
	.vcTXERibMon4 2204	ribbon text for <b>Mai</b>
	.vcTXERibMon5 2205	ribbon text for <b>June</b>
	.vcTXERibMon6 2206	ribbon text for <b>July</b>
	.vcTXERibMon7 2207	ribbon text for <b>August</b>
	.vcTXERibMon8 2208	ribbon text for <b>September</b>
	.vcTXERibMon9 2209	ribbon text for <b>October</b>
	.vcTXERiboClock 2224	ribbon text for <b>o'clock</b>
	.vcTXERibPM 2226	ribbon text for <b>pm</b>
	.vcTXERibQuar0 2219	ribbon text for <b>first quarter</b>
	.vcTXERibQuar1 2220	ribbon text for <b>second quarter</b>
	.vcTXERibQuar2 2221	ribbon text for <b>third quarter</b>
	.vcTXERibQuar3 2222	ribbon text for <b>fourth quarter</b>
⇐ textEntry	System.String	Text entry to replace the default text
⇐ returnStatus	VcReturnStatus	Return status
	<b>Possible Values:</b>	
	.vcRetStatDefault 2	The default behavior remains unchanged.
	.vcRetStatFalse 0	The default behavior will not be performed.
	.vcRetStatNoPopup 4	The popup of the context menu is inhibited.
	.vcRetStatOK 1	The default behavior will be performed.

• Pointer mode	vcTXECTxmenArrowMode
Mode: Create node	vcTXECTxmenCreateNodeMode
Mode: Create link	vcTXECTxmenCreateLinkMode
Mode: Create box	vcTXECTxmenCreateBoxMode
Build sub diagram	vcTXECTxmenSubDiagram
Restore full diagram	vcTXECTxmenFullDiagram
Page setup...	vcTXECTxmenPageLayout
Print setup...	vcTXECTxmenFilePrintSetup
Print preview...	vcTXECTxmenFilePrintPreview
Print...	vcTXECTxmenFilePrint
Show world view	vcTXECTxmenShowWorldView
Show legend view	vcTXECTxmenShowLegendView
Export graphics...	vcTXECTxmenGraphicExport

*Constants of the diagram's context menu*

Edit data...	vcTXECTxmenEditNode
Delete nodes	vcTXECTxmenDeleteNode
Build sub diagram	vcTXECTxmenSubDiagram
Restore full diagram	vcTXECTxmenFullDiagram
Outline outdent	vcTXECTxmenGroupOutlineOutdent
Outline indent	vcTXECTxmenGroupOutlineIndent

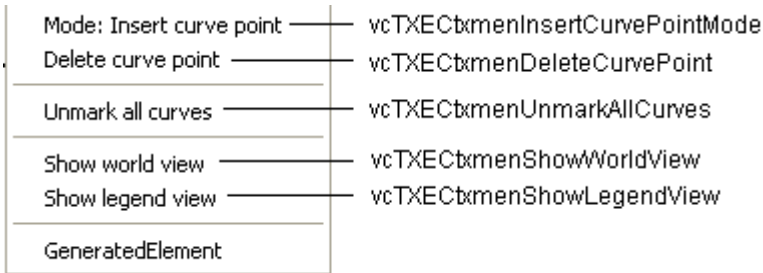
*Constants of the context menu for nodes*

Collapse Group	vcTXECTxmenGroupCollapse
Collapse Rows Below	vcTXECTxmenGroupCollapseRowsBelow
All Nodes In One Row	vcTXECTxmenGroupNodesInOneRow
Arrange Nodes Overlaid	vcTXECTxmenGroupNodesOverlaid
Delete group	vcTXECTxmenGroupDelete
Edit group data...	vcTXECTxmenEditGroup

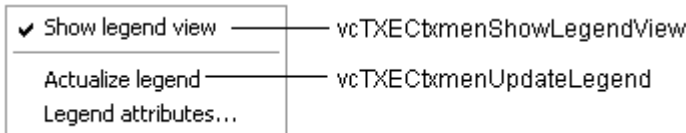
*Constants of the context menu for groups with no groups collapsed*

Expand Group	vcTXECTxmenGroupExpand
Expand Rows Below	vcTXECTxmenGroupExpandRowsBelow
All Nodes In One Row	
Arrange Nodes Optimized	vcTXECTxmenGroupNodesOptimized
Delete group	
Edit group data...	

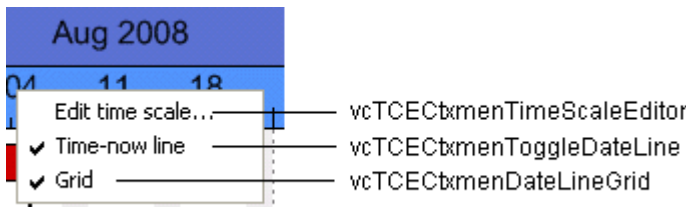
*Constants of the context menu for groups with no groups expanded*



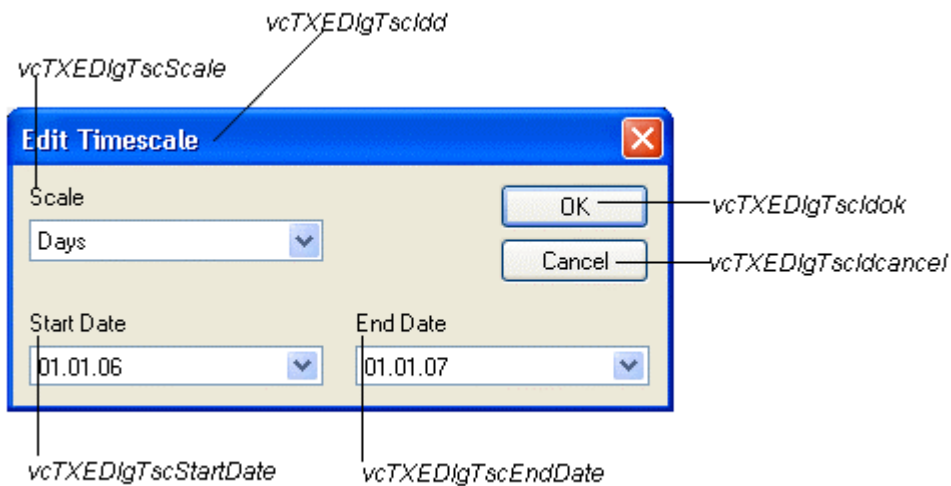
*Constants of the context menu for histograms*



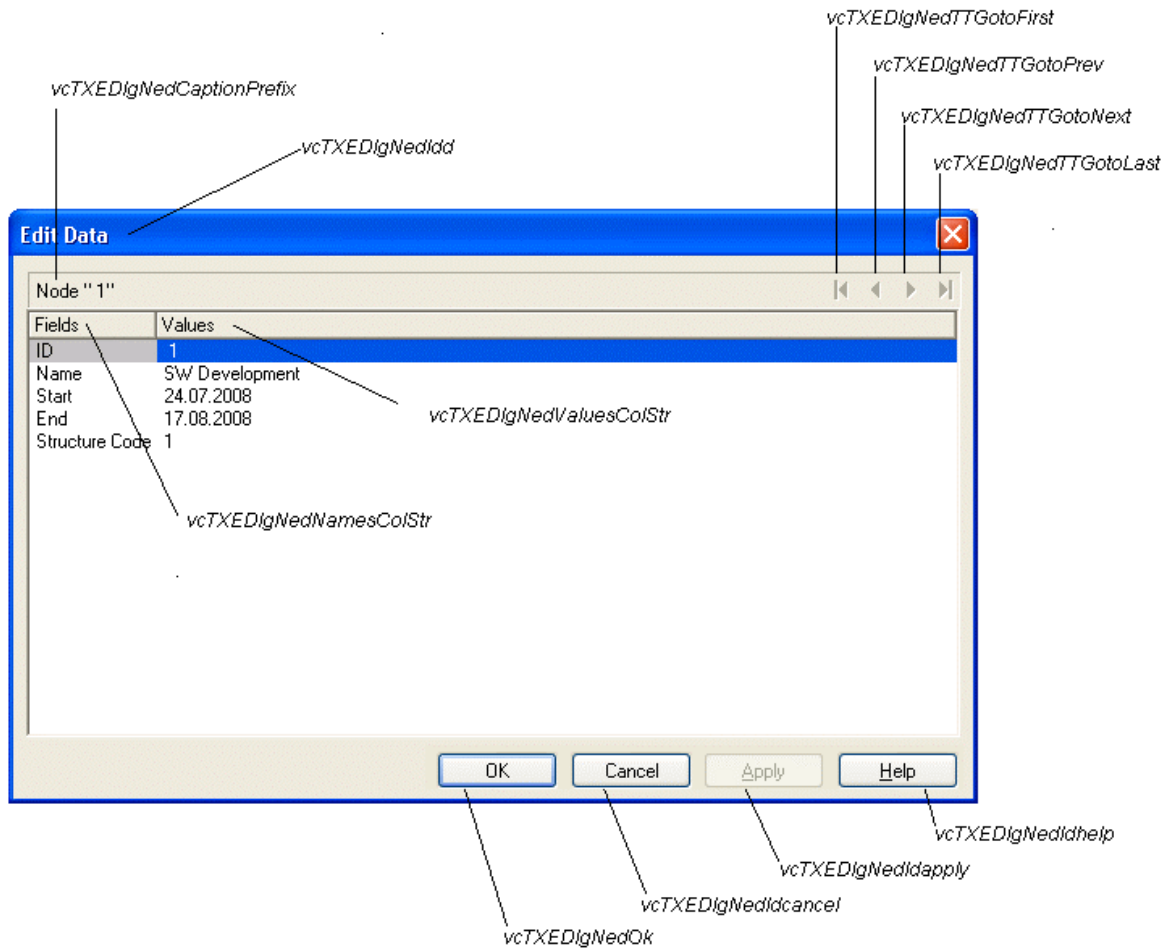
*Constants of the legend's context menu*



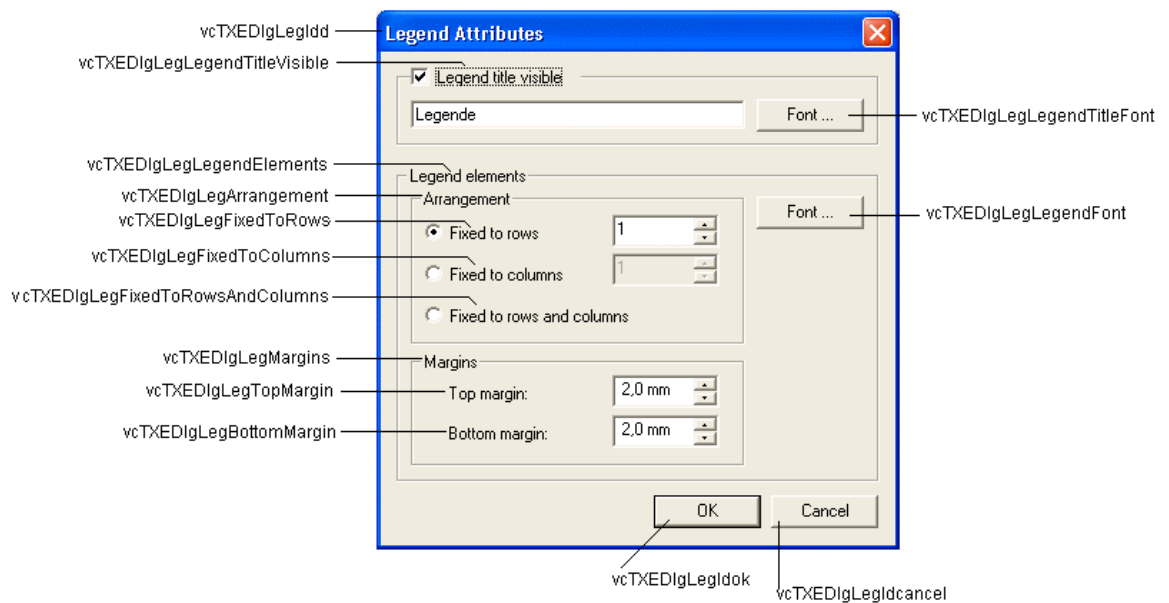
*Constants of the time scale's context menu*



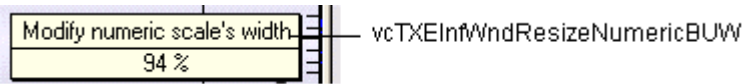
*Constants of the dialog **Edit Time Scale***



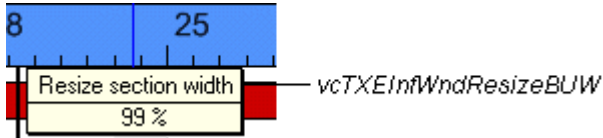
Constants of the dialog **Edit Data**



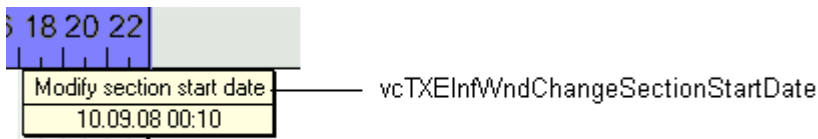
Constants of the **Legend attributes** dialog



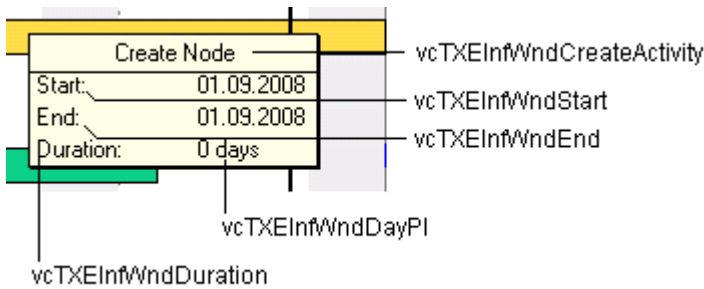
Constant of the tooltip text that appears on resizing the basic unit width of the **numeric scale in the histogram**



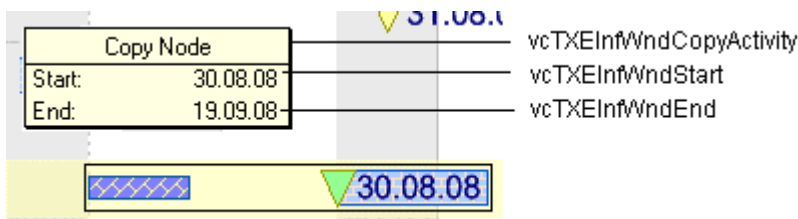
Constants of the tooltip text that appears on resizing the **time scale section width**



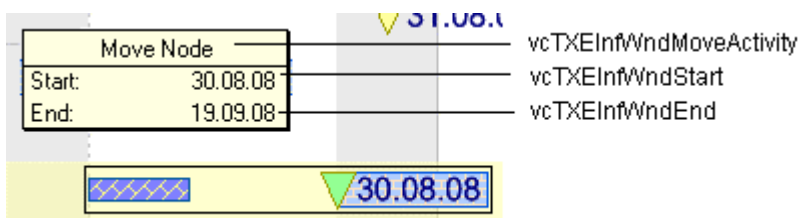
Constants of the tooltip text that appears on modifying the **start date of a time scale section**



Constants of the tooltip text that appears on **creating a node**



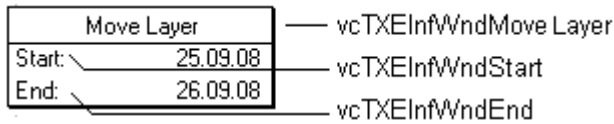
Constants of the tooltip text that appears on **copying a node**



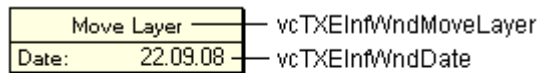
Constants of the tooltip text that appears on **moving a node**



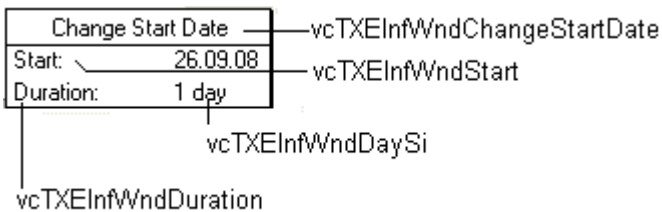
## 632 API Reference: VcGantt



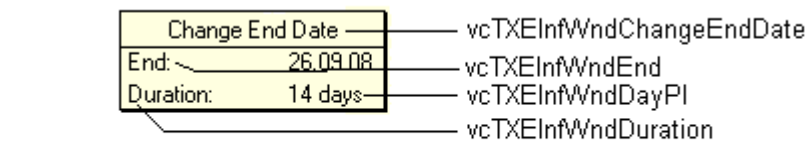
Constants of the tooltip text that appears on **moving a layer**



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**

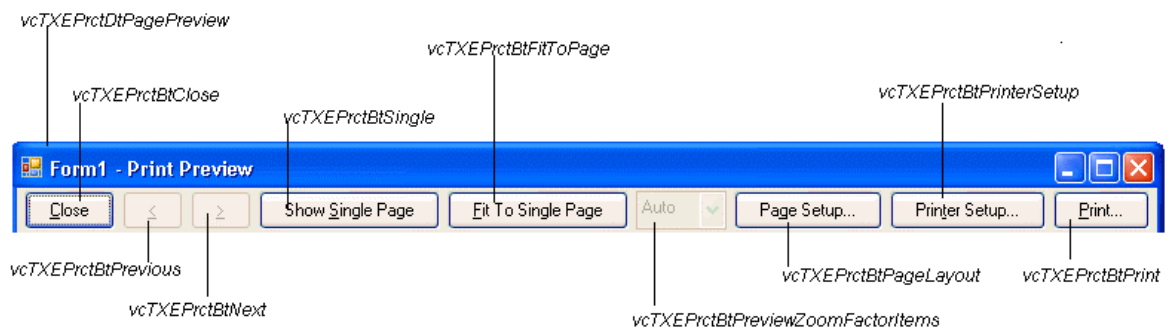


Constants of the tooltip text that appears on **modifying the end date of a node**

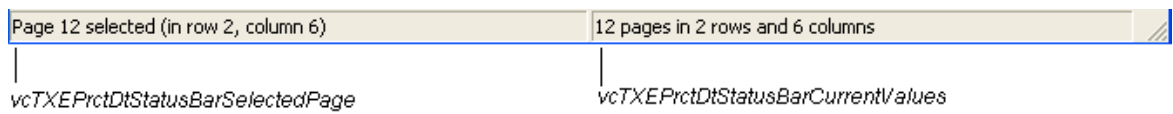


vcTXEPrctBtAll

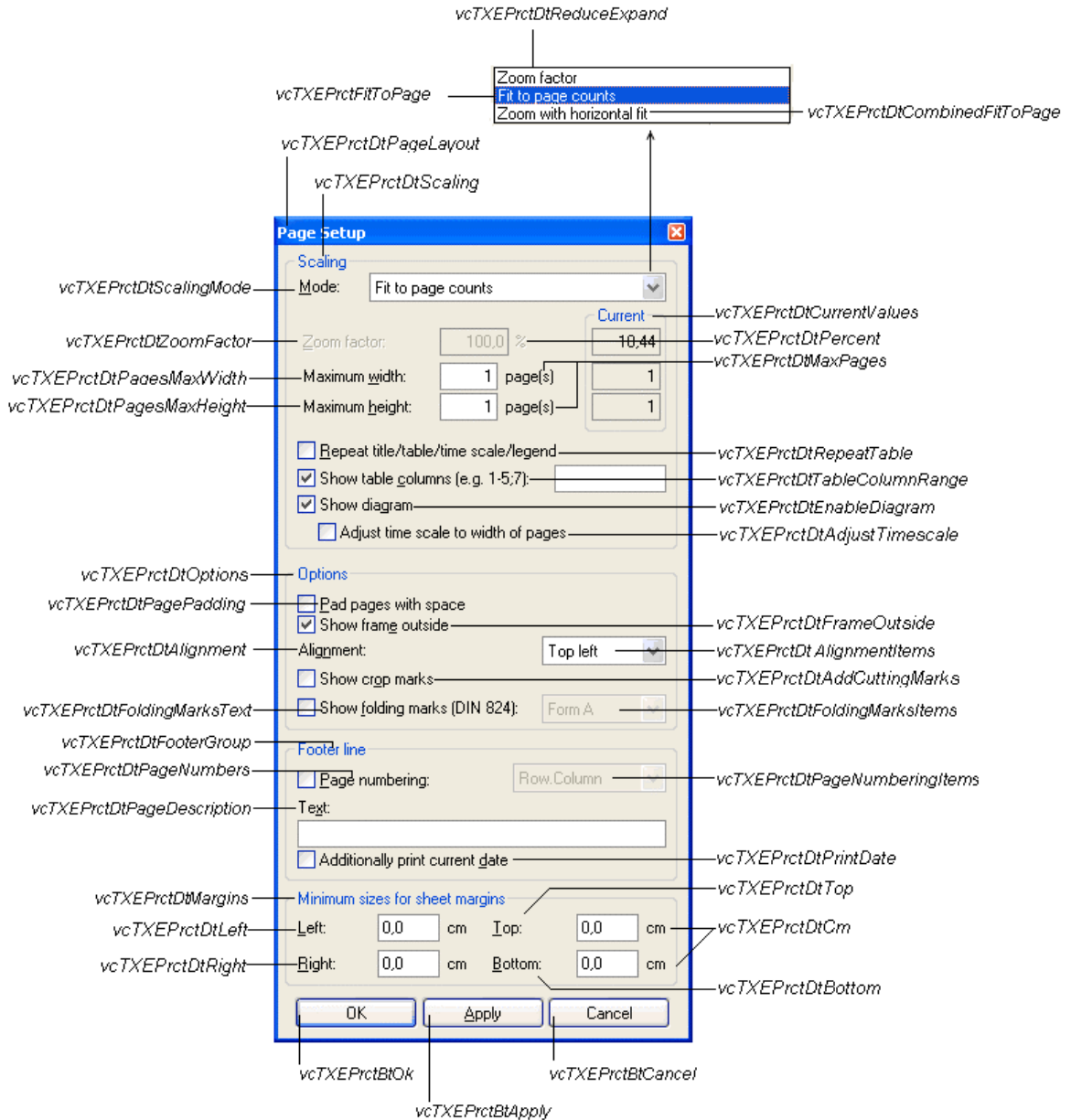
Constants of the button texts of the **Print Preview** dialog



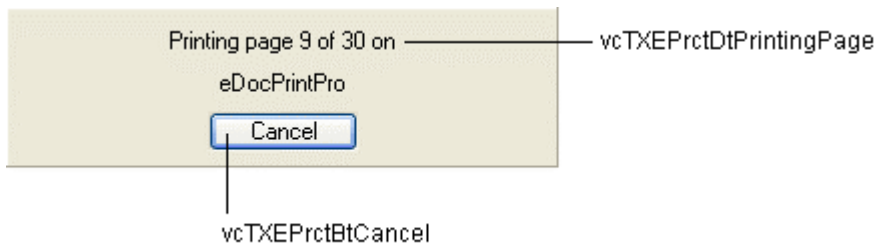
**Constants of the button texts of the *Print preview Overview***



**Constants of the status bar in the dialog *Print Preview***



Constants of the **Page Setup** dialog



Constants of the info box **Printing**

**Example Code VB.NET**

```
Private Sub VcGantt1_VcTextEntrySupplying(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcTextEntrySupplyingEventArgs) Handles
VcGantt1.VcTextEntrySupplying
    Select Case e.ControlIndex
        Case VcTextEntryIndex.vcTXEPrctBtNext
            e.Text = "Next page"
        Case VcTextEntryIndex.vcTXEPrctBtPrevious
            e.Text = "Previous page"
    End Select
End Sub
```

**Example Code C#**

```
private void vcGantt1_VcTextEntrySupplying(object sender,
NETRONIC.XGantt.VcTextEntrySupplyingEventArgs e)
{
    switch (e.ControlIndex)
    {
        case VcTextEntryIndex.vcTXEPrctBtNext:
            e.Text = "Next page";
            break;
        case VcTextEntryIndex.vcTXEPrctBtPrevious:
            e.Text = "Previous page";
            break;
    }
}
```

## VcTimeScaleLeftClicking

**Event of VcGantt**

This event occurs when the user clicks the left mouse button on the timescale. The TimeScale object and the mouse position (x,y-coordinates) are returned.

	Data Type	Explanation
<b>Parameter:</b>		
⇒ sender	VcGantt	Reference to the object that triggered the event
⇒ e	VcTimeScaleClickingEventArgs	Object specific to the event that is being handled

### Properties of the VcTimeScaleClickingEventArgs object

	Data Type	Explanation
<b>Properties:</b>		
⇒ timeScale	VcTimeScale	Timescale hit
⇒ x	System.Int32	X coordinate of the mouse cursor
⇒ y	System.Int32	Y coordinate of the mouse cursor
⇔ returnStatus	VcReturnStatus	Return status
	<b>Possible Values:</b> .vcRetStatDefault 2	The default behavior remains unchanged.

.vcRetStatFalse 0	The default behavior will not be performed.
.vcRetStatNoPopup 4	The popup of the context menu is inhibited.
.vcRetStatOK 1	The default behavior will be performed.

**Example Code VB.NET**

```
Private Sub VcGantt1_VcTimeScaleLeftClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcTimeScaleClickingEventArgs) Handles
VcGantt1.VcTimeScaleLeftClicking
    VcGantt1.TimeScaleCollection.Active.BackgroundColor = Color.Blue
End Sub
```

**Example Code C#**

```
private void vcGanttASP1_VcTimeScaleLeftClicking(object sender,
NETRONIC.XGantt.VcTimeScaleClickingEventArgs e)
{
    vcGantt1.TimeScaleCollection.Active.BackgroundColor = Color.LightSteelBlue;
}
```

## VcTimeScaleSectionRescaled

**Event of VcGantt**

This event occurs when the user has finished rescaling a time scale section. The TimeScale object, the section index and the new basicUnitWidth are passed.

	Data Type	Explanation
<b>Properties:</b>		
⇒ timeScale	VcTimeScale	Timescale
⇒ sectionIndex	System.Int16	Section index
⇒ newBasicUnitWidth	System.Int32	New width of the basic unit

## VcToolTipTextSupplying

**Event of VcGantt**

This event occurs if the VcGantt property **ToolTipTextSupplyingEventEnabled** is set to **True** or if the check box **VcToolTipSupplying events** 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 is positioned 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 suppress the tooltip.

	Data Type	Explanation
<b>Parameter:</b>		
⇒ sender	VcGantt	Reference to the object that triggered the event
⇒ e	VcToolTipTextSupplyingEventArgs	Object specific to the event that is being handled

### Properties of the VcToolTipTextSupplyingEventArgs object

	Data Type	Explanation
<b>Properties:</b>		
⇒ hitObject	VcObject	Object hit
⇒ hitObjectType	VcObjectType	Type of the object hit
	<b>Possible Values:</b> .vcObjTypeBox 15 .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 <b>box</b> object type <b>curve</b> object type <b>date line</b> object type <b>group</b> object type <b>group in diagram area</b> object type <b>group in table area</b> object type <b>histogram</b> object type <b>layer</b> object type <b>link collection</b> object type <b>node in diagram area</b> object type <b>node in legend area</b> object type <b>node in table area</b> no object object type <b>numeric scale</b> object type <b>summary bar</b> object type <b>table</b> object type <b>table caption</b> object type <b>time scale</b>
⇒ x	System.Int32	X coordinate of the mouse cursor
⇒ y	System.Int32	Y value of the mouse cursor
⇐ tooltipText	System.String	Tooltip text, can contain 1024 characters maximum
⇐ returnStatus	VcReturnStatus	Return status
	<b>Possible Values:</b> .vcRetStatDefault 2 .vcRetStatFalse 0 .vcRetStatNoPopup 4 .vcRetStatOK 1	The default behavior remains unchanged. The default behavior will not be performed. The popup of the context menu is inhibited. The default behavior will be performed.

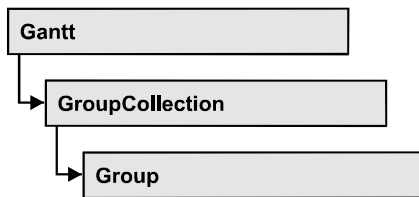
### Example Code VB.NET

```
Private Sub VcGantt1_VcToolTipTextSupplying(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcToolTipTextSupplyingEventArgs) Handles
VcGantt1.VcToolTipTextSupplying
    Dim node As VcNode
    Dim toolTipText as String
    If Convert.ToString(e.HitObject) = "NETRONIC.XGantt.VcNode" Then
        node = DirectCast(e.HitObject, VcNode)
        Select Case e.HitObjectType
            Case VcObjectType.vcObjTypeNodeInDiagram
                toolTipText = Convert.ToString(node.DataField(1))
            Case VcObjectType.vcObjTypeNodeInTable
                toolTipText = Convert.ToString(node.DataField(1))
        End Select
    End If
    If (toolTipText.Length > 0) Then
toolTipText = "<div style='border: 1px solid black; padding: 2px; background-
color: #ffffda; font-family: Arial, Helvetica, sans-serif; font-size: 8pt;
white-space: nowrap;'>" + toolTipText + "</div>"
    End If
    e.Text = toolTipText
End Sub
```

### Example Code C#

```
Private void vcGantt1_VcToolTipTextSupplying(object sender,
NETRONIC.XGantt.VcToolTipTextSupplyingEventArgs e)
{
    VcNode node;
    String toolTipText = "";
    if (e.HitObject.ToString() == "NETRONIC.XGantt.VcNode")
    {
        node = (VcNode)e.HitObject;
        switch(e.HitObjectType)
        {
            case VcObjectType.vcObjTypeNodeInDiagram:
                toolTipText = Convert.ToString(node.get_DataField(1));
                break;
            case VcObjectType.vcObjTypeNodeInTable:
                toolTipText = Convert.ToString(node.get_DataField(1));
                break;
        }
    }
    if (toolTipText->get_Length() > 0)
    {
        toolTipText = "<div style='border: 1px solid black; padding: 2px;
background-color: #ffffda; font-family: Arial, Helvetica, sans-serif; font-size:
8pt; white-space: nowrap;'>" + toolTipText + "</div>";
    }
    e.Text = toolTipText;
}
```

## 6.31 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

- DataField
- GroupingLevel
- ID
- Marked
- Name
- NodeCollection
- NodesAndGroupsBelowCollapsed
- NodesArrangedInOneRow
- NodesOptimized
- SubGroups
- SuperGroup
- Visible

### Methods

- DataRecord
- Delete
- RelatedDataRecord
- ReOptimizeNodes
- Update



## Properties

### DataField

Property of VcGroup

This property lets you set or retrieve the contents of a DataField of the group record. The group record is a copy of the node record of the first node added to the group. The data field referred to by its field index. To update the group, the **Update** method needs to be invoked.

The property DataField is an Indexed Property, which in C# is addressed by the methods set\_DataField (index, pvn) and get\_DataField (index).

	Data Type	Explanation
<b>Parameter:</b> ⇒ index	System.Int16	Index of the data field
<b>Property value</b>	Void	

#### Example Code VB.NET

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup
Dim nodecltn As VcNodeCollection
Dim node As VcNode

groupCltn = VcGantt1.GroupCollection
For Each group In groupCltn
    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
Next
group.Update()
Next
```

#### Example Code C#

```
VcGroupCollection groupCltn = vcGantt1.GroupCollection;
foreach (VcGroup group in groupCltn)
{
    VcNodeCollection nodeCltn = vcGantt1.NodeCollection;
    foreach (VcNode node in nodeCltn)
    {
        if (node.get_DataField(3) > group.get_DataField(3))
            group.set_DataField(3,node.get_DataField(3));
    }
    group.Update();
}
```

## GroupingLevel

Read Only Property of VcGroup

This property lets you retrieve 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	System.Int16	Grouping level of the group

### Example Code VB.NET

```
Dim group As VcGroup
Dim subGroup As VcGroup
Dim nodeCltn As VcNodeCollection
Dim node As VcNode

nodeCltn = VcGantt1.NodeCollection
node = nodeCltn.FirstNode
group = node.SuperGroup
If group.GroupingLevel > 0 Then
    subGroup = group.SuperGroup
End If
```

### Example Code C#

```
VcNodeCollection nodeCltn = vcGantt1.NodeCollection;
VcNode node = nodeCltn.FirstNode();
VcGroup group = node.SuperGroup;
VcGroup subGroup;

if (group.GroupingLevel > 0)
    subGroup = group.SuperGroup;
```

## ID

Read Only Property of VcGroup

By this property you can retrieve the ID of a group.

	Data Type	Explanation
Property value	System.String	Group ID

### Example Code VB.NET

```
Code-Beispiel VB.NET
Dim groupCltn As VcGroupCollection
Dim group As VcGroup
Dim groupID As String
groupCltn = VcGanttASP1.GroupCollection
group = groupCltn.FirstGroup
groupID = group.ID

MsgBox (group.ID)
```

## 642 API Reference: VcGroup

### Example Code C#

```
VcGroupCollection groupCltn = vcGanttASPI.GroupCollection;
VcGroup group = groupCltn.FirstGroup();
string groupID = group.ID;
MessageBox.Show(group.ID);
```

## Marked

Property of VcGroup

This property lets you set or retrieve whether a group is marked.

	Data Type	Explanation

### Example Code VB.NET

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup

groupCltn = VcGantt1.GroupCollection
groupCltn.SelectGroups(VcSelectionType.vcSelected)

For Each group In groupCltn
    group.Marked = False
Next
```

### Example Code C#

```
VcGroupCollection groupCltn = vcGantt1.GroupCollection;
groupCltn.SelectGroups(VcSelectionType.vcAll);

foreach (VcGroup group in groupCltn)
{
    Group.Marked = false;
}
```

## 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	System.String	Group name

**Example Code VB.NET**

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup
Dim groupName As String

groupCltn = VcGanttASPl.GroupCollection
group = groupCltn.FirstGroup
groupName = group.Name
```

**Example Code C#**

```
VcGroupCollection groupCltn = vcGanttASPl.GroupCollection;
VcGroup group = groupCltn.FirstGroup();
string groupName = group.Name;
```

**NodeCollection****Read Only Property of VcGroup**

This property lets you access all nodes that belong to a group.

	Data Type	Explanation
<b>Property value</b>	VcNodeCollection	NodeCollection object

**Example Code VB.NET**

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup
Dim nodeCltn As VcNodeCollection

groupCltn = VcGantt1.GroupCollection
group = groupCltn.FirstGroup
nodeCltn = group.NodeCollection
```

**Example Code C#**

```
VcGroupCollection groupCltn = vcGantt1.GroupCollection;
VcGroup group = groupCltn.FirstGroup();
VcNodeCollection nodeCltn = group.NodeCollection;
```

**NodesAndGroupsBelowCollapsed****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 using the **Collapsed** property, in addition groups that do not belong to a subgroup will be collapsed as well.

	Data Type	Explanation
<b>Property value</b>	System.Boolean	Rows below the top row are/are not collapsed

## 644 API Reference: VcGroup

### Example Code VB.NET

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup

groupCltn = VcGantt1.GroupCollection
group = groupCltn.GroupByName("A")

group.NodesAndGroupsBelowCollapsed = True
```

### Example Code C#

```
VcGroupCollection groupCltn = vcGantt1.GroupCollection;
VcGroup group = groupCltn.GroupByName("A");
group.NodesAndGroupsBelowCollapsed = true;
```

## NodesArrangedInOneRow

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
Property value	System.Boolean	All nodes of the group are/are not in the same row

### Example Code VB.NET

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup

groupCltn = VcGantt1.GroupCollection
group = groupCltn.GroupByName("A")

group.NodesArrangedInOneRow = True
```

### Example Code C#

```
VcGroupCollection groupCltn = vcGantt1.GroupCollection;
VcGroup group = groupCltn.GroupByName("A");
group.NodesArrangedInOneRow = true;
```

## NodesOptimized

Property of VcGroup

This property lets you set or retrieve whether (True) the node layout is optimized or if nodes overlap (False). The latter case may only occur when the **NodesArrangedInOneRow** property was set to **True**.

	Data Type	Explanation
Property value	System.Boolean	The node layout is/is not at its optimum

**Example Code VB.NET**

```
Dim group As VcGroup
For Each group In VcGantt1.GroupCollection
    group.NodesArrangedInOneRow = True
    group.NodesOptimized = True
Next
```

**Example Code C#**

```
foreach (VcGroup group in groupCltn)
{
    group.NodesArrangedInOneRow = true;
    group.NodesOptimized = true;
}
```

## SubGroups

**Read Only Property of VcGroup**

In a multi-level grouping arrangement, this property lets you retrieve subgroups, that are returned by a group collection object.

	Data Type	Explanation
Property value	VcGroupCollection	GroupCollection object containing the subgroups

**Example Code VB.NET**

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup
Dim subGroupCltn As VcGroupCollection

groupCltn = VcGantt1.GroupCollection
group = groupCltn.GroupByName("A")
subGroupCltn = group.SubGroups
```

**Example Code C#**

```
VcGroupCollection groupCltn = vcGantt1.GroupCollection;
VcGroup group = groupCltn.GroupByName("A");
VcGroupCollection 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

## 646 API Reference: VcGroup

### Example Code VB.NET

```
Dim group As VcGroup
Dim subGroup As VcGroup
Dim nodeCltn As VcNodeCollection
Dim node As VcNode

nodeCltn = VcGantt1.NodeCollection
node = nodeCltn.FirstNode
group = node.SuperGroup
If group.GroupingLevel > 0 Then
    superGroup = group.SuperGroup
End If
```

### Example Code C#

```
VcNodeCollection nodeCltn = vcGantt1.NodeCollection;
VcNode node = nodeCltn.FirstNode();
VcGroup group = node.SuperGroup;
VcGroup superGroup;

if (group.GroupingLevel > 0)
    superGroup = group.SuperGroup;
```

## 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	System.Boolean	Group visible/invisible

### Example Code VB.NET

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup

groupCltn = VcGantt1.GroupCollection
group = groupCltn.GroupByName("A")
group.Visible = True
```

### Example Code C#

```
VcGroupCollection groupCltn = vcGantt1.GroupCollection;
VcGroup group = groupCltn.GroupByName("A");

group.Visible = true;
```

## 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

### Delete

Method of VcGroup

This method lets you delete a group. Deleting a group is possible only if the group is empty. Activities have to be deleted from the group before the group can be deleted.

	Data Type	Explanation
Return value	System.Boolean	Group was/was not deleted successfully

#### Example Code VB.NET

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup
Dim nodeCltn As VcNodeCollection
Dim node As VcNode

groupCltn = VcGantt1.GroupCollection
group = groupCltn.GroupByName("A")
nodeCltn = group.NodeCollection
For Each node In nodeCltn
    node.Delete()
Next
group.Delete()
```

#### Example Code C#

```
VcGroupCollection groupCltn = vcGantt1.GroupCollection;
VcGroup group = groupCltn.GroupByName("A");
VcNodeCollection nodeCltn = group.NodeCollection;

foreach (VcNode node in nodeCltn)
{
    node.Delete();
}
group.Delete();
```



## RelatedDataRecord

Method of VcGroup

This property 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
<b>Parameter:</b> ⇒ index	System.Int16	Index of data field that holds the key
<b>Return value</b>	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
<b>Return value</b>	Void	

## Update

Method of VcGroup

This method lets you update a group after having changed a data field by the **DataField** property.

	Data Type	Explanation
<b>Return value</b>	System.Boolean	Group successfully/not successfully updated

**Example Code VB.NET**

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup
Dim nodeCltn As VcNodeCollection
Dim node As VcNode

groupCltn = VcGantt1.GroupCollection
group = groupCltn.GroupByName("A")
nodeCltn = group.NodeCollection

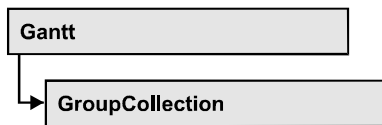
group.DataField(3) = nodeCltn.FirstNode.DataField(3)
For Each node In nodeCltn
    If node.DataField(3) > group.DataField(3) Then
        group.DataField(3) = node.DataField(3)
    End If
Next
group.Update()
```

**Example Code C#**

```
VcGroupCollection groupCltn = vcGantt1.GroupCollection;
VcGroup group = groupCltn.GroupByName("A");
VcNodeCollection nodeCltn = group.NodeCollection;

group.set_DataField(3, nodeCltn.FirstNode().get_DataField(3));
foreach(VcNode node in nodeCltn)
{
    if (node.get_DataField(3) > group.get_DataField(3))
        group.set_DataField(3, node.get_DataField(3));
}
group.Update();
```

## 6.32 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

- `Count`

### Methods

- `FirstGroup`
- `GetEnumerator`
- `GroupByName`
- `NextGroup`
- `SelectGroups`

---

## Properties

### Count

**Read Only Property of VcGroupCollection**

This property lets you retrieve the number of groups in the group collection.

	Data Type	Explanation
Property value	System.Int32	Number of nodes

### Example Code VB.NET

```

Dim groupCltn As VcGroupCollection
Dim group As VcGroup
Dim numberOfGroups As Integer

groupCltn = VcGanttASP1.GroupCollection
numberOfGroups = groupCltn.Count
  
```

**Example Code C#**

```
VcGroupCollection groupCltn = vcGanttASPl.GroupCollection;
int 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 VB.NET**

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup

groupCltn = VcGanttASPl.GroupCollection
group = groupCltn.FirstGroup
```

**Example Code C#**

```
VcGroupCollection groupCltn = vcGanttASPl.GroupCollection;
VcGroup group = groupCltn.FirstGroup();
```

### GetEnumerator

**Method of VcGroupCollection**

This method returns an Enumerator object which supports the iteration by language specific elements. It is implied in the For...Each construct of Visual Basic and C#. This object allows to iterate over the group objects included.

	Data Type	Explanation
Return value	VcObject	Reference object

## 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
<b>Parameter:</b>		
⇒ Rückgabewert	VcGroup	Group
⇒ groupName	System.String	Name of group
<b>Return value</b>	VcGroup	Group

### Example Code VB.NET

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup

groupCltn = VcGantt1.GroupCollection
group = groupCltn.GroupByName("Group A")
```

### Example Code C#

```
VcGroupCollection groupCltn = vcGantt1.GroupCollection;
VcGroup group = groupCltn.GroupByName("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
<b>Return value</b>	VcGroup	Subsequent group

### Example Code VB.NET

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup

groupCltn = VcGanttASP1.GroupCollection
group = groupCltn.FirstGroup
While Not group Is Nothing
    ListBox1.Items.Add(group.Name)
    group = groupCltn.NextGroup
End While
```

**Example Code C#**

```
VcGroupCollection groupCltn = vcGanttASPl.GroupCollection;
VcGroup group = groupCltn.FirstGroup();
while (group != null)
{
    listBox1.Items.Add(group.Name);
    group = groupCltn.NextGroup();
}
```

**SelectGroups**

Method of VcGroupCollection

This method lets you specify the groups that the group collection is to contain.

	Data Type	Explanation
<b>Parameter:</b> ⇒ groupSelType	VcGroupSelectionType  <b>Possible Values:</b> .vcAllGroups 0 .vcCollapsedGroups 1 .vcExpandedGroups 2 .vcInvisibleGroups 5 .vcSelectedGroups 3 .vcVisibleGroups 4	Type of group to be selected  All groups selected Collapsed groups selected Expanded groups selected Invisible groups selected Selected groups selected Visible groups selected
<b>Return value</b>	System.Int32	Number of groups selected

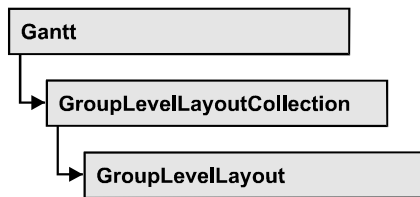
**Example Code VB.NET**

```
Dim groupCltn As VcGroupCollection
groupCltn = VcGantt1.GroupCollection
groupCltn.SelectGroups(VcGroupSelectionType.vcAllGroups)
```

**Example Code C#**

```
VcGroupCollection groupCltn = vcGantt1.GroupCollection;
groupCltn.SelectGroups(VcGroupSelectionType.vcAllGroups);
```

## 6.33 VcGroupLevelLayout



An object of the type `VcGroupLevelLayout` defines the content and the appearance of grouping levels: name of the grouping level, level number, the grouping field, sorting and sort order as various options concerning the layout of calendar and line grids and separation lines.

### Properties

- `AllNodesInOneRow`
- `CalendarGridName`
- `CalendarGridsVisible`
- `CalendarGridsWithChildGroups`
- `CalendarNameDataFieldIndex`
- `Collapsed`
- `DateLineGridName`
- `DateLineGridsVisible`
- `DateLineGridsWithChildGroups`
- `GroupDataFieldIndex`
- `GroupNodesVisible`
- `Level`
- `ModificationsAllowed`
- `Name`
- `NodesArrangedOptimized`
- `OptimizedNodesSortDataFieldIndex`
- `OptimizedNodesSortOrder`
- `OverlaidNodesSortDataFieldIndex`
- `OverlaidNodesSortOrder`
- `RowBackColorAsARGB`
- `RowBackColorDataFieldIndex`
- `RowBackColorMapName`
- `RowPattern`
- `RowPatternColorAsARGB`
- `RowPatternColorDataFieldIndex`
- `RowPatternColorMapName`

- RowPatternDataFieldIndex
- RowPatternMapName
- SeparationLineColor
- SeparationLineColorDataFieldIndex
- SeparationLineColorMapName
- SeparationLinesVisible
- SeparationLinesVisibleAtTop
- SeparationLineThickness
- SeparationLineType
- SortDataFieldIndex
- SortOrder
- Specification
- SummaryBarsVisible
- Visible

---

## Properties

### AllNodesInOneRow

Property of VcGroupLevelLayout

This property lets you specify/enquire whether (True) or not (False) the node objects of the group of this level are positioned the same row.

	Data Type	Explanation
Property value	System.Boolean	All nodes of the group are/are not in the same row

### 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
Property value	System.String	name of the calendar grid



## CalendarGridsVisible

Property of VcGroupLevelLayout

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 **Grouping** dialog.

	Data Type	Explanation
Property value	System.Boolean	Workfree periods are/are not accentuated

## 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	System.Boolean	calendar grid for subgroups are/are not displayed

## 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	System.Int32	Index of the data field which contains the name of the calendar

## Collapsed

Property of VcGroupLevelLayout

This property lets you set or retrieve, whether (True) or not (False) a group level is collapsed.

	Data Type	Explanation
Property value	System.Boolean	Group collapsed/expanded

## 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	System.String	name of the date line grid

## DateLineGridsVisible

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	System.Boolean	Date grids are/are not displayed.

## DateLineGridsWithChildGroups

Read Only 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	System.Boolean	date line grids for subgroups are/are not displayed

## GroupDataFieldIndex

Read Only 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	System.Int32	index used for grouping of this VcGroupLevelLayout object

## GroupNodesVisible

Property of VcGroupLevelLayout

This property lets you set or retrieve whether this level's group nodes are displayed. This property also can be set in the **Grouping** dialog.

	Data Type	Explanation
Property value	System.Boolean	group nodes are/are not visible

## 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	System.Int32	Grouping level of the group level layout

## 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	System.Boolean	Modifications allowed (True)/ not allowed (False)

**Example Code VB.NET**

```
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	System.String	Name of the group level

**NodesArrangedOptimized****Property of VcGroupLevelLayout**

This property lets you specify/enquire whether (True) the node layout on this group level is optimized or if nodes overlap (False). The **AllNodesInOne Row** property has to be set to **True** in both cases.

	Data Type	Explanation
Property value	System.Boolean	The node layout is/is not at its optimum

**Example Code VB.NET**

```
group.LevelLayout.NodesArrangedOptimized = True
```

**Example Code C#**

```
group.LevelLayout.NodesArrangedOptimized = 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	System.Int32	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	Direction of the sorting 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	System.Int32	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	Direction of the sorting order

## 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	System.Drawing.Color	ARGB color values {0...255},{0...255},{0...255},{0...255}

## 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	System.Int32	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	System.String	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	Pattern type

## 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	System.Drawing.Color	ARGB color values {0...255},{0...255},{0...255},{0...255}

## 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	System.Int32	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	System.String	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	System.Int32	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	System.String	Name of the pattern 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	System.Drawing.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	System.Int32	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	System.String	Name of the color map

## SeparationLinesVisible

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
Property value	System.Boolean	Separation lines are displayed/not displayed

## SeparationLinesVisibleAtTop

Property of VcGroupLevelLayout

This property lets you set or retrieve whether separation lines are to be displayed between groups of different grouping levels.

This property also can be set in the **Groupwise** section of the **Grouping** dialog.

	Data Type	Explanation
Property value	System.Boolean	Separation lines at top are displayed/not displayed

## SeparationLineThickness

Property of VcGroupLevelLayout

This property lets you set or retrieve the line thickness of a separation line between group 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

## 666 API Reference: VcGroupLevelLayout

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 **Grouping** dialog, section **Groupwise**, field **Separation Line**.

	Data Type	Explanation
Property value	System.Int32	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 specify/enquire 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  <b>Possible Values:</b> .vcDashed 4 .vcDashedDotted 5 .vcDotted 3 .vcNone 1 .vcSolid 2	Type of separation lines of hierarchy levels  Line dashed Line dashed-dotted Line dotted No line type assigned Line solid

## 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
<b>Parameter:</b> ⇒ sortlevel	System.Int32	Sorting level
<b>Property value</b>	System.Int32	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
<b>Parameter:</b> ⇒ sortLevel	System.Int32	Sorting level
<b>Property value</b>	SortOrderEnum	Direction of the sorting order

## 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 **VcGroupLevelLayout.AddBySpecification**.

	Data Type	Explanation
<b>Property value</b>	System.String	Specification of the group level layout

## SummaryBarsVisible

Property of VcGroupLevelLayout

This property lets you specify/enquire 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	System.Boolean	summary bars visible (True)/ invisible (False)

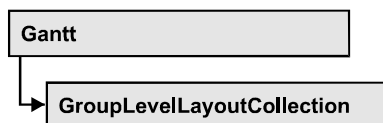
## Visible

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	System.Boolean	Group level visible/invisible

## 6.34 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

- Count

### Methods

- Add
- AddBySpecification
- Copy
- FirstGroupLevelLayout
- GetEnumerator
- GroupLevelLayoutByIndex
- GroupLevelLayoutByName
- NextGroupLevelLayout
- Remove
- Update

---

## Properties

### 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	System.Int32	Number of group level layouts

## 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
<b>Parameter:</b> ⇒ groupLevelLayoutName	System.String	name of group level layout
<b>Return value</b>	VcGroupLevelLayout	New group level layout object

### 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
<b>Parameter:</b> ⇒ Specification	System.String	Group level layout specification
<b>Return value</b>	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
<b>Parameter:</b>		
⇒ groupLevelLayoutName	System.String	Name of the group level layout to be copied
⇒ newGroupLevelLayoutName	System.String	Name of the new group level layout
<b>Return value</b>	VcGroupLevelLayout	Group level layout object

## 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
<b>Return value</b>	VcGroupLevelLayout	First group level layout

## GetEnumerator

### Method of VcGroupLevelLayoutCollection

This method returns an Enumerator object which supports the iteration by language specific elements. It is implied in the For...Each construct of Visual Basic and C#. This object allows to iterate over the group objects included.

	Data Type	Explanation
<b>Return value</b>	System.Object	Reference object



## 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
<b>Parameter:</b> ⇒ index	System.Int16	Index of the group level layout
<b>Return value</b>	VcGroupLevelLayout	Group level layout object returned

## 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
<b>Parameter:</b> ⇒ groupLevelLayoutName	System.String	Name of the group level layout
<b>Return value</b>	VcGroupLevelLayout	Group level layout

## 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
<b>Return value</b>	VcGroupLevelLayout	Subsequent group level layout

## 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
<b>Parameter:</b> ⇒ groupLevelLayoutName	System.String	Group level layout name
<b>Return value</b>	System.Boolean	Group level layout deleted (True)/not deleted (False)

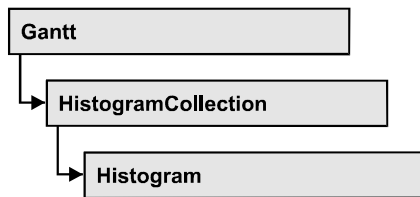
## Update

### 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
<b>Return value</b>	System.Boolean	update successful (True)/ not successful (False)

## 6.35 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

- `CalendarGridsVisible`
- `CalendarName`
- `CurveCollection`
- `Name`
- `NumericScaleCollection`
- `Visible`

### Methods

- `FitRangeIntoView`
- `GetCurrentYValues`
- `PutInOrderAfter`
- `ScrollToValue`
- `SetMaxYValue`
- `SetMinYValue`

---

## Properties

### CalendarGridsVisible

**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	System.Boolean	Workfree periods are/are not accentuated

## CalendarName

### Read Only 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	System.String	Character string that passes the calendar name

## CurveCollection

### Read Only Property of VcHistogram

This property gives access to the curve collection object, that is, to the curves that it contains.

	Data Type	Explanation
Property value	VcCurveCollection	CurveCollection object

### Example Code VB.NET

```
Dim histogramCltn As VcHistogramCollection
Dim histogram As VcHistogram
Dim curveCltn As VcCurveCollection

histogramCltn = VcGantt1.HistogramCollection
histogram = histogramCltn.HistogramByName("HISTOGRAM_1")
curveCltn = histogram.CurveCollection
```

### Example Code C#

```
VcHistogramCollection histogramCltn = vcGantt1.HistogramCollection;
VcHistogram histogram = histogramCltn.HistogramByName("HISTOGRAM_1");
VcCurveCollection curveCltn = histogram.CurveCollection;
```

## Name

### Read Only Property of VcHistogram

This property lets you retrieve the name of a histogram curve.

## 676 API Reference: VcHistogram

	Data Type	Explanation
Property value	System.String	Name of the histogram

### Example Code VB.NET

```
Dim histogramCltn As VcHistogramCollection
Dim histogram As VcHistogram

histogramCltn = VcGantt1.HistogramCollection
histogram = histogramCltn.Active
MsgBox(histogram.Name)
```

### Example Code C#

```
VcHistogramCollection histogramCltn = vcGantt1.HistogramCollection;
VcHistogram histogram = histogramCltn.Active;
MessageBox.Show(histogram.Name);
```

## NumericScaleCollection

### Read Only Property of VcHistogram

This property lets you access the NumericScaleCollection object, that contains all numeric scales available.

	Data Type	Explanation
Property value	VcNumericScaleCollection	NumericScaleCollection object

### Example Code VB.NET

```
Dim histogramCltn As VcHistogramCollection
Dim histogram As VcHistogram
Dim numericScaleCltn As VcNumericScaleCollection

histogramCltn = VcGantt1.HistogramCollection
histogram = histogramCltn.HistogramByName("HISTOGRAM_1")
numericScaleCltn = histogram.NumericScaleCollection
```

### Example Code C#

```
VcHistogramCollection histogramCltn = vcGantt1.HistogramCollection;
VcHistogram histogram = histogramCltn.HistogramByName("HISTOGRAM_1");
VcNumericScaleCollection numericScaleCltn = histogram.NumericScaleCollection;
```

## Visible

### Property of VcHistogram

This property lets you set or retrieve whether the histogram is visible.

	Data Type	Explanation
Property value	System.Boolean	Histogram visible (True)/ not visible (False)

**Example Code VB.NET**

```
Dim histogramCltn As VcHistogramCollection
Dim histogram As VcHistogram

histogramCltn = VcGantt1.HistogramCollection
histogram = histogramCltn.Active
histogram.Visible = True
```

**Example Code C#**

```
VcHistogramCollection histogramCltn = vcGantt1.HistogramCollection;
VcHistogram 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
<b>Parameter:</b>		
⇒ startValue	System.Int32	Start value of the section to be matched
⇒ endValue	System.Int32	End value of the section to be matched
⇒ gapAsNoOfTimeUnits	System.Int32	Parameter is not used
<b>Return value</b>	System.Boolean	Area could/could not be matched.

### 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**).

## 678 API Reference: VcHistogram

	Data Type	Explanation
<b>Parameter:</b>		
↵ minValue	System.Int32	Minimum Y-value of all curves
↵ maxValue	System.Int32	Maximum Y-value of all curves
<b>Return value</b>	System.Boolean	High-low values could (True) / could not (False) be successfully retrieved.

## 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
<b>Parameter:</b>		
⇒ refName	System.String	Name of the histogram, after which the current histogram shall be placed.
<b>Return value</b>	Void	

## 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
<b>Parameter:</b>		
⇒ value	System.Int32	Y value to be scrolled to
⇒ verAlignment	VcVerticalAlignment	Vertical alignment
	<b>Possible Values:</b> .vcBottomAligned 2 .vcTopAligned 1 .vcVerCenterAligned - 1	bottom aligned top aligned vertically centered
<b>Return value</b>	System.Boolean	Scrolling was/was not performed successfully.

**Example Code VB.NET**

```
Dim histogramCltn As VcHistogramCollection
Dim histogram As VcHistogram

histogramCltn = VcGantt1.HistogramCollection
histogram = histogramCltn.HistogramByName("HISTOGRAM_1")
histogram.ScrollToValue(7, VcVerticalAlignment.vcVerCenterAligned)
```

**Example Code C#**

```
VcHistogramCollection histogramCltn = vcGantt1.HistogramCollection;
VcHistogram histogram = histogramCltn.HistogramByName("HISTOGRAM_1");
histogram.ScrollToValue(7, VcVerticalAlignment.vcVerCenterAligned);
```

## SetMaxYValue

**Method of VcHistogram**

This method lets you set 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 y values of the curves.

	Data Type	Explanation
<b>Parameter:</b> ⇒ yValue	System.Int32	Maximum y-value
<b>Return value</b>	System.Int32	Maximum y value set (1)/not set (0)

**Example Code VB.NET**

```
Dim histogramCltn As VcHistogramCollection
Dim histogram As VcHistogram

histogramCltn = VcGantt1.HistogramCollection
histogram = histogramCltn.HistogramByName("HISTOGRAM_1")
histogram.SetMaxYValue(20)
```

**Example Code C#**

```
VcHistogramCollection histogramCltn = vcGantt1.HistogramCollection;
VcHistogram histogram = histogramCltn.HistogramByName("HISTOGRAM_1");
histogram.SetMaxYValue(20);
```

## SetMinYValue

**Method of VcHistogram**

This method lets you specify a minimum value of the numeric scale of the histogram.

	Data Type	Explanation
<b>Parameter:</b> ⇒ yValue	System.Int32	Minimum y-value



## 680 API Reference: VcHistogram

---

<b>Return value</b>	System.Int32	Minimum y value set (1)/not set (0)
---------------------	--------------	-------------------------------------

### Example Code VB.NET

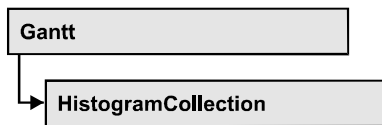
```
Dim histogramCltn As VcHistogramCollection
Dim histogram As VcHistogram

histogramCltn = VcGantt1.HistogramCollection
histogram = histogramCltn.HistogramByName("HISTOGRAM_1")
histogram.SetMinYValue(2)
```

### Example Code C#

```
VcHistogramCollection histogramCltn = vcGantt1.HistogramCollection;
VcHistogram histogram = histogramCltn.HistogramByName("HISTOGRAM_1");
histogram.SetMinYValue(2);
```

## 6.36 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

- Active
- Count

### Methods

- Delete
- FirstHistogram
- GetEnumerator
- HistogramByIndex
- HistogramByName
- NextHistogram

---

## Properties

### Active

**Property of VcHistogramCollection**

This property lets you set or retrieve the name of the histogram currently used.

The active histogram may be `NOTHING`, if user actions did not take place yet in the histogram area. A histogram can be activated for example by marking a curve.

## 682 API Reference: VcHistogramCollection

	Data Type	Explanation
Property value	VcHistogram	Currently used histogram

### Example Code VB.NET

```
Dim histogramCltn As VcHistogramCollection
Dim histogram As VcHistogram

histogramCltn = VcGantt1.HistogramCollection
histogram = histogramCltn.Active
```

### Example Code C#

```
VcHistogramCollection histogramCltn = vcGantt1.HistogramCollection;
VcHistogram 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	System.Int32	Number of histograms

### Example Code VB.NET

```
Dim histogramCltn As VcHistogramCollection
Dim numberOfHistograms As Integer

histogramCltn = VcGantt1.HistogramCollection
numberOfHistograms = histogramCltn.Count
```

### Example Code C#

```
VcHistogramCollection histogramCltn = vcGantt1.HistogramCollection;
int numberOfHistograms = histogramCltn.Count;
```

---

## Methods

### Delete

#### Method of VcHistogramCollection

By this method you can delete a histogram.

	Data Type	Explanation
--	-----------	-------------

## 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 VB.NET

```
Dim histogramCltn As VcHistogramCollection
Dim histogram As VcHistogram

histogramCltn = VcGantt1.HistogramCollection
histogram = histogramCltn.FirstHistogram
```

### Example Code C#

```
VcHistogramCollection histogramCltn = vcGantt1.HistogramCollection;
VcHistogram histogram = histogramCltn.FirstHistogram();
```

## GetEnumerator

Method of VcHistogramCollection

This method returns an Enumerator object which supports the iteration by language specific elements. It is implied in the For...Each construct of Visual Basic and C#. This object allows to iterate over the histogram objects included.

	Data Type	Explanation
Return value	VcObject	Reference object

## HistogramByIndex

Method of VcHistogramCollection

This method lets you access a histogram by its index. If a histogram does not exist at the index specified, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
<b>Parameter:</b> ⇒ index	System.Int16	Index of the histogram
<b>Return value</b>	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
<b>Parameter:</b> ⇒ histogramName	System.String	Name of the histogram
<b>Return value</b>	VcHistogram	Histogram

### Example Code VB.NET

```
Dim histogramCltn As VcHistogramCollection
Dim histogram As VcHistogram

histogramCltn = VcGantt1.HistogramCollection
histogram = histogramCltn.HistogramByName("Histogram_2")
```

### Example Code C#

```
VcHistogramCollection histogramCltn = vcGantt1.HistogramCollection;
VcHistogram histogram = histogramCltn.HistogramByName("Histogram_2");
```

## 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
<b>Return value</b>	VcHistogram	Succeeding histogram

**Example Code VB.NET**

```
Dim histogramCltn As VcHistogramCollection
Dim histogram As VcHistogram

histogramCltn = VcGantt1.HistogramCollection
histogram = histogramCltn.FirstHistogram

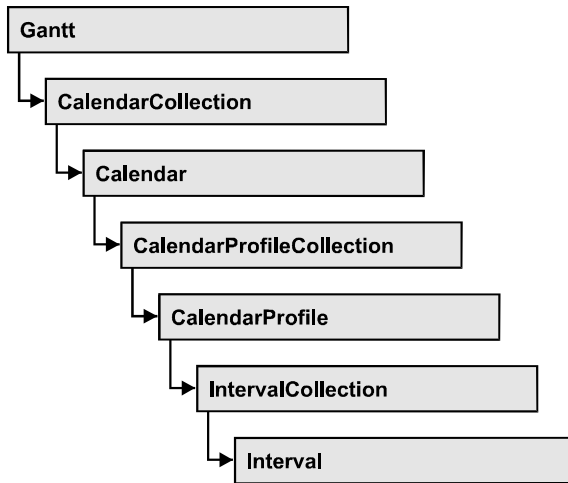
While Not histogram Is Nothing
    ListBox1.Items.Add(histogram.Name)
    histogram = histogramCltn.NextHistogram
End While
```

**Example Code C#**

```
VcHistogramCollection histogramCltn = vcGantt1.HistogramCollection;
VcHistogram histogram = histogramCltn.FirstHistogram();

while (histogram != null)
{
    listBox1.Items.Add(histogram.Name);
    histogram = histogramCltn.NextHistogram();
}
```

## 6.37 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 explicitly 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:

<b>vcCalendar-Interval</b>	<b>vcYearProfile-Interval</b>	<b>vcWeekProfile-Interval</b>	<b>vcDayProfile-Interval</b>	<b>vcShift-Interval</b>
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** specifies 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 **vcSeconds** without referring to a date. Example: 4 hours.

## Properties

- BackgroundColor
- CalendarProfileName
- DayInEndMonth
- DayInStartMonth
- Duration
- EndDateTime
- EndMonth
- EndTime
- EndWeekday
- LineColor
- LineThickness
- Pattern
- PatternColor
- Specification
- StartDateTime
- StartMonth
- StartTime
- StartWeekday
- Text
- TimeUnit
- Type
- UseGraphicalAttributes

---

## Properties

### BackgroundColor

Property of VcInterval

This property lets you set or retrieve the background 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 background color can also be set in the **Administrate Intervals** dialog.

	Data Type	Explanation
Property value	System.DrawingColor	ARGB color values {0...255},{0...255},{0...255},{0...255}

## 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	System.String	Name of the calendar profile

## 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	System.Int16	Day of last month

## DayInStartMonth

Property of VcInterval

This property returns or sets the day in 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	System.Int16	Day of first month

## 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 **Edit Shift Calendar** dialog. This feature can also be set in the **Administrate Intervals** dialog.

	Data Type	Explanation
Property value	System.Int32	Last weekday of interval

## EndTime

### 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	System.DateTime	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	VcMonth  <b>Possible Values:</b> .vcApril 4 .vcAugust 8 .vcDecember 12 .vcFebruary 2 .vcJanuary 1	End month of interval  <b>April</b> <b>August</b> <b>December</b> <b>February</b> <b>Januar</b>

.vcJuly 7	<b>July</b>
.vcJune 6	<b>une</b>
.vcMarch 3	<b>March</b>
.vcMay 5	<b>May</b>
.vcNovember 11	<b>November</b>
.vcOctober 10	<b>October</b>
.vcSeptember 9	<b>September</b>

## 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 **Administrative Intervals** dialog.

	Data Type	Explanation
Property value	System.DateTime	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 **Administrative Intervals** dialog.

	Data Type	Explanation
Property value	VcWeekday  <b>Possible Values:</b> .vcFriday 5 .vcMonday 1 .vcSaturday 6 .vcSunday 7 .vcThursday 4 .vcTuesday 2 .vcWednesday 3	Last weekday of interval  Week day <b>Friday</b> Week day <b>Monday</b> Week day <b>Saturday</b> Week day <b>Sunday</b> Week day <b>Thursday</b> Week day <b>Tuesday</b> Week day <b>Wednesday</b>

## LineColor

**Read Only Property of VcInterval**

This property lets you set or retrieve the line color of an interval and can also be set in the **Administrative Intervals** dialog. This feature can also be set in the **Administrative Intervals** dialog.

	Data Type	Explanation
Property value	System.Drawing.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 **Administrative Intervals** dialog.

	Data Type	Explanation
Property value	System.Int16	Line thickness  LineType {1...4}: line thickness in pixels  LineType {5...1000}: line thickness in 1/100 mm




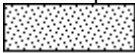
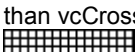






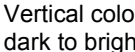





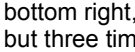
## Pattern

### 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	VcFillPattern	Pattern type <b>Default value:</b> As defined in the dialog
	<b>Possible Values:</b> .vc05PercentPattern... vc90PercentPattern 01 - 11	Dots in foreground color on background color, the density of the foreground color increasing with the percentage 
	.vcAeroGlassPattern 44	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 twice the line width
.vcDashedHorizontalPattern 2026	Dashed horizontal lines
.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
.vcHorizontalPattern 3	Horizontal lines
.vcLargeCheckerboardPattern 2044	Checkerboard pattern showing squares of twice the size of vcSmallCheckerBoardPattern
.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
.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 
.vcVerticalPattern 2	Vertical lines 
.vcVerticalTopLightedConvexPattern 42	Vertical color gradient from bright to dark 
.vcWavePattern 2031	Horizontal waves pattern 
.vcWeavePattern 2034	Interwoven stripes 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 

## PatternColor

**Read Only 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	System.Drawing.Color	RGB color values {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 **VcIntervalCollection.AddBySpecification**.

	Data Type	Explanation
Property value	String	Specification of the interval

## 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	System.DateTime	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	VcMonth  <b>Possible Values:</b> .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  <b>April</b> <b>August</b> <b>December</b> <b>February</b> <b>Januar</b> <b>July</b> <b>une</b> <b>March</b> <b>May</b> <b>November</b> <b>October</b> <b>September</b>

## 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	System.DateTime	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	VcWeekday  <b>Possible Values:</b>	Start weekday of interval

.vcFriday 5	Week day <b>Friday</b>
.vcMonday 1	Week day <b>Monday</b>
.vcSaturday 6	Week day <b>Saturday</b>
.vcSunday 7	Week day <b>Sunday</b>
.vcThursday 4	Week day <b>Thursday</b>
.vcTuesday 2	Week day <b>Tuesday</b>
.vcWednesday 3	Week day <b>Wednesday</b>

## Text

Property of VcInterval

This property lets you set or retrieve the text of the time ribbon *only for calendar profiles of the type **vcShiftProfile***. The text can also be set in the **Edit Shift Calendar** dialog.

	Data Type	Explanation
Property value	System.String	Annotation text of the time ribbon

## 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 time unit can also be set in the **Administrative Intervals** dialog.

	Data Type	Explanation
Property value	VcTimeUnit  <b>Possible Values:</b> .vcDay 5 .vcHour 6 .vcMinute 7 .vcSecond 8	Time unit  Time unit <b>day</b> Time unit <b>hour</b> Time unit <b>minute</b> Time unit <b>second</b>

## Type


Read Only Property of VcInterval

This property lets you enquire the type of the interval. This feature can also be set in the **Administrative Intervals** dialog.

	Data Type	Explanation
Property value	VcIntervalType  <b>Possible Values:</b> .vcCalendarInterval 139 .vcDayProfileInterval 4 .vcIntervalProfileInterval 5 .vcWeekProfileInterval 3 .vcYearProfileInterval 2	Type of the interval

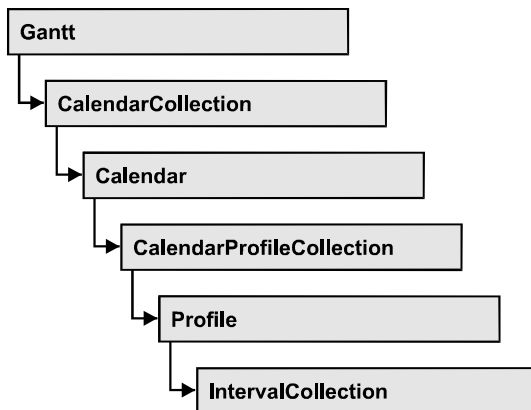
## UseGraphicalAttributes

### Read Only 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**.

	Data Type	Explanation
Property value	System.Boolean	Graphical attributes of the interval are displayed (True)/are not displayed (False)

## 6.38 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 **IntervalByIndex**. 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

- Count

### Methods

- Add
- AddBySpecification
- Copy
- FirstInterval
- IntervalByIndex
- IntervalByName
- NextInterval
- Remove
- Update

---

## Properties

### Count

**Read Only Property of VcIntervalCollection**

This property lets you retrieve the number of intervals in the interval collection.

	Data Type	Explanation
Property value	System.Int32	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
<b>Parameter:</b> ⇒ intervalName	System.String	Interval name
<b>Return value</b>	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
<b>Parameter:</b> ⇒ Specification	System.String	Interval specification
<b>Return value</b>	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
<b>Parameter:</b> ⇒ intervalName	System.String	Name of the interval to be copied
⇒ newIntervalName	System.String	Name of the new interval
<b>Return value</b>	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
<b>Return value</b>	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
<b>Parameter:</b> ⇒ Index	System.Int16	Index of the interval
<b>Return value</b>	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
<b>Parameter:</b> ⇒ intervalName	System.String	Name of the interval object
<b>Return value</b>	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
<b>Return value</b>	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
<b>Parameter:</b> ⇒ intervalName	System.String	interval name
<b>Return value</b>	System.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
<b>Return value</b>	System.Boolean	update successful (True)/ not successful (False)



## 6.39 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

- BackgroundColor
- BackgroundColorDataFieldIndex
- BackgroundColorMapName
- CompletionDataFieldIndex
- DurationDataFieldIndex
- EndDataFieldIndex
- FilterName
- Format
- GraphicsFileName
- GraphicsFileNameDataFieldIndex
- GraphicsFileNameMapName
- Height
- HeightDataFieldIndex
- HorizontalOffset
- LabelSizeDependence
- LegendText
- LineColor
- LineColorDataFieldIndex
- LineColorMapName
- LineThickness
- LineType
- Movable
- Name
- ObjectDrawEventsEnabled
- Pattern
- PatternColor
- PatternColorDataFieldIndex

- PatternColorMapName
- PatternDataFieldIndex
- PatternMapName
- Shape
- Sizeable
- Specification
- StartDataFieldIndex
- ThreeDEffect
- UsedAsOverlapLayer
- VerticalOffset
- VerticalOffsetDataFieldIndex
- VerticalOffsetMapName
- Visible
- VisibleInLegend

### Methods

- CalculateCurrentWidth
- PutInOrderAfter

---

## Properties

### BackgroundColor

**Property of VcLayer**

This property lets you set 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.

If in the property **BackgroundColorMapName** a map is specified, the map will set the background color in dependence on data.

	Data Type	Explanation
<b>Property value</b>	System.Drawing.Color	RGB color values {(0...255},{0...255},{0...255)}

## BackgroundColorDataFieldIndex

Property of VcLayer

This property lets you set or retrieve the data field index to be used with the property **BackgroundColorMapName**. If you set this property to **-1**, no map will be used.

	Data Type	Explanation
Property value	System.Int16	Data field index

### Example Code VB.NET

```
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

dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
dataRecCltn = dataTable.DataRecordCollection
dataRec1 = dataRecCltn.Add("1;Node 1;01.01.14;;8;Red")

VcGantt1.EndLoading()

mapCltn = VcGantt1.MapCollection
map = mapCltn.Add("MapColor")

map.Type = VcMapType.vcColorMap
mapEntry = map.CreateEntry
mapEntry.DataFieldValue = "Green"
mapEntry.Color = System.Drawing.Color.Green
mapEntry = map.CreateEntry
mapEntry.DataFieldValue = "Red"
mapEntry.Color = System.Drawing.Color.Red
mapCltn.Update()

layer = VcGantt1.LayerCollection.LayerByIndex(0)
layer.BackgroundColorMapName = "MapColor"
layer.BackgroundColorDataFieldIndex = 5
VcGantt1.LayerCollection.Update()
```

**Example Code C#**

```

VcDataTable dataTable =
vcGantt1.DataTableCollection.DataTableByName("Maindata");
VcDataRecordCollection dataRecCltn = dataTable.DataRecordCollection;

VcDataRecord dataRec1 = dataRecCltn.Add("1;Node 1;02.01.14;;8;Red");
vcGantt1.EndLoading();

VcMapCollection mapCltn = vcGantt1.MapCollection;
VcMap map = mapCltn.Add("MapColor");
map.Type = VcMapType.vcColorMap;

VcMapEntry mapEntry = map.CreateEntry();
mapEntry.DataFieldValue = "Green";
mapEntry.Color = System.Drawing.Color.Green;
mapEntry = map.CreateEntry();
mapEntry.DataFieldValue = "Red";
mapEntry.Color = System.Drawing.Color.Red;
mapCltn.Update();

VcLayer layer = vcGantt1.LayerCollection.LayerByIndex(0);
layer.BackgroundColorMapName = "MapColor";
layer.BackgroundColorDataFieldIndex = 5;
vcGantt1.LayerCollection.Update()

```

**BackgroundColorMapName****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 in the property **BackgroundColorDataFieldIndex**, then the background color is controlled by the map. If no data field entry applies, the background color of the layer that is specified in the property **BackgroundColor** will be used.

	<b>Data Type</b>	<b>Explanation</b>
<b>Property value</b>	System.String	Name of the color map

## 708 API Reference: VcLayer

### Example Code VB.NET

```
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

dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
dataRecCltn = dataTable.DataRecordCollection
dataRec1 = dataRecCltn.Add("1;Node 1;01.01.14;;8;Red")

VcGantt1.EndLoading()

mapCltn = VcGantt1.MapCollection
map = mapCltn.Add("MapColor")

map.Type = VcMapType.vcColorMap
mapEntry = map.CreateEntry
mapEntry.DataFieldValue = "Green"
mapEntry.Color = System.Drawing.Color.Green
mapEntry = map.CreateEntry
mapEntry.DataFieldValue = "Red"
mapEntry.Color = System.Drawing.Color.Red
mapCltn.Update()

layer = VcGantt1.LayerCollection.LayerByIndex(0)
layer.BackgroundColorMapName = "MapColor"
layer.BackgroundColorDataFieldIndex = 5
VcGantt1.LayerCollection.Update()
```

### Example Code C#

```
VcDataTable dataTable =
vcGantt1.DataTableCollection.DataTableByName("Maindata");
VcDataRecordCollection dataRecCltn = dataTable.DataRecordCollection;

VcDataRecord dataRec1 = dataRecCltn.Add("1;Node 1;02.01.14;;8;Red");
vcGantt1.EndLoading();

VcMapCollection mapCltn = vcGantt1.MapCollection;
VcMap map = mapCltn.Add("MapColor");
map.Type = VcMapType.vcColorMap;

VcMapEntry mapEntry = map.CreateEntry();
mapEntry.DataFieldValue = "Green";
mapEntry.Color = System.Drawing.Color.Green;
mapEntry = map.CreateEntry();
mapEntry.DataFieldValue = "Red";
mapEntry.Color = System.Drawing.Color.Red;
mapCltn.Update();

VcLayer layer = vcGantt1.LayerCollection.LayerByIndex(0);
layer.BackgroundColorMapName = "MapColor";
layer.BackgroundColorDataFieldIndex = 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	System.String	Index of the data field that contains the degree of completion

## 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	System.Int32	Index of the data field that contains the duration

## 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

## 710 API Reference: VcLayer

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	System.Int16	Index of the data field that contains the end value

### 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	System.String	Filter name

### Format

Read Only Property of VcLayer

This property lets you retrieve the format of the layer.

	Data Type	Explanation
Property value	VcLayerFormat	Layer format

### 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 name has to denote an existing graphics file. *Available formats:*

- \*.BMP (Microsoft Windows Bitmap)
- \*.EMF (Enhanced Metafile)
- \*.EMF, with EMF+ included

- \*.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, 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	System.String	Name of the graphics file



## 712 API Reference: VcLayer

### Example Code VB.NET

```
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

dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
dataRecCltn = dataTable.DataRecordCollection
dataRec1 = dataRecCltn.Add("1;Node 1;03.01.14;;;8;Pic1.bmp")
VcGantt1.EndLoading()

mapCltn = VcGantt1.MapCollection
map = mapCltn.Add("MapGraphic")

map.Type = VcMapType.vcGraphicsFileMap
mapEntry = map.CreateEntry
mapEntry.GraphicsFileName = "c:\Pic1.bmp"
mapCltn.Update()

layer = VcGantt1.LayerCollection.LayerByIndex(0)
layer.Shape = VcLayerShape.vcBitmapLayer
layer.GraphicsFileName = "c:\Pic1.bmp"
layer.GraphicsFileNameMapName = "MapGraphic"
layer.GraphicsFileNameDataFieldIndex = 5
VcGantt1.LayerCollection.Update()
```

### Example Code C#

```
VcDataTable dataTable =
vcGantt1.DataTableCollection.DataTableByName("Maindata");
VcDataRecordCollection dataRecCltn = dataTable.DataRecordCollection;

VcDataRecord dataRec1 = dataRecCltn.Add("1;Node 1;02.03.14;;;8;Pic1.bmp");
vcGantt1.EndLoading();

VcMapCollection mapCltn = vcGantt1.MapCollection;
VcMap map = mapCltn.Add("MapGraphic");
map.Type = VcMapType.vcGraphicsFileMap;

VcMapEntry mapEntry = map.CreateEntry();
mapEntry.GraphicsFileName = @"c:\Pic1.bmp";
mapCltn.Update();

VcLayer layer = vcGantt1.LayerCollection.LayerByIndex(0);
layer.Shape = VcLayerShape.vcBitmapLayer;
layer.GraphicsFileName = "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 loaded 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	System.Int16	Index of the data field

### Example Code VB.NET

```
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

dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
dataRecCltn = dataTable.DataRecordCollection
dataRec1 = dataRecCltn.Add("1;Node 1;03.01.14;;8;Pic1.bmp")
VcGantt1.EndLoading()

mapCltn = VcGantt1.MapCollection
map = mapCltn.Add("MapGraphic")

map.Type = VcMapType.vcGraphicsFileMap
mapEntry = map.CreateEntry
mapEntry.GraphicsFileName = "c:\Pic1.bmp"
mapCltn.Update()

layer = VcGantt1.LayerCollection.LayerByIndex(0)
layer.Shape = VcLayerShape.vcBitmapLayer
layer.GraphicsFileName = "c:\Pic1.bmp"
layer.GraphicsFileNameMapName = "MapGraphic"
layer.GraphicsFileNameDataFieldIndex = 5
VcGantt1.LayerCollection.Update()
```

### Example Code C#

```
VcDataTable dataTable =
vcGantt1.DataTableCollection.DataTableByName("Maindata");
VcDataRecordCollection dataRecCltn = dataTable.DataRecordCollection;

VcDataRecord dataRec1 = dataRecCltn.Add("1;Node 1;02.03.14;;8;Pic1.bmp");
vcGantt1.EndLoading();

VcMapCollection mapCltn = vcGantt1.MapCollection;
VcMap map = mapCltn.Add("MapGraphic");
map.Type = VcMapType.vcGraphicsFileMap;

VcMapEntry mapEntry = map.CreateEntry();
mapEntry.GraphicsFileName = @"c:\Pic1.bmp";
mapCltn.Update();

VcLayer layer = vcGantt1.LayerCollection.LayerByIndex(0);
layer.Shape = VcLayerShape.vcBitmapLayer;
layer.GraphicsFileName = "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	System.String	Name of the graphics map

### Example Code VB.NET

```
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

dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
dataRecCltn = dataTable.DataRecordCollection
dataRec1 = dataRecCltn.Add("1;Node 1;03.01.14;;8;Pic1.bmp")
VcGantt1.EndLoading()

mapCltn = VcGantt1.MapCollection
map = mapCltn.Add("MapGraphic")

map.Type = VcMapType.vcGraphicsFileMap
mapEntry = map.CreateEntry
mapEntry.GraphicsFileName = "c:\Pic1.bmp"
mapCltn.Update()

layer = VcGantt1.LayerCollection.LayerByIndex(0)
layer.Shape = VcLayerShape.vcBitmapLayer
layer.GraphicsFileName = "c:\Pic1.bmp"
layer.GraphicsFileNameMapName = "MapGraphic"
layer.GraphicsFileNameDataFieldIndex = 5
VcGantt1.LayerCollection.Update()
```

**Example Code C#**

```

VcDataTable dataTable =
vcGantt1.DataTableCollection.DataTableByName("Maindata");
VcDataRecordCollection dataRecCltn = dataTable.DataRecordCollection;

VcDataRecord dataRec1 = dataRecCltn.Add("1;Node 1;02.03.14;;8;Pic1.bmp");
vcGantt1.EndLoading();

VcMapCollection mapCltn = vcGantt1.MapCollection;
VcMap map = mapCltn.Add("MapGraphic");
map.Type = VcMapType.vcGraphicsFileMap;

VcMapEntry mapEntry = map.CreateEntry();
mapEntry.GraphicsFileName = @"c:\Pic1.bmp";
mapCltn.Update();

VcLayer layer = vcGantt1.LayerCollection.LayerByIndex(0);
layer.Shape = VcLayerShape.vcBitmapLayer;
layer.GraphicsFileName = "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
<b>Property value</b>	System.Int32	Height in 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 **Vc.LayerCollection.Update()**.

	Data Type	Explanation
<b>Property value</b>	System.Int32	Data field index

## Example Code VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim layer As VcLayer

layer = VcGantt1.LayerCollection.FirstLayer
mapCltn = VcGantt1.MapCollection
mapCltn.SelectMaps(VcMapType.vcMillimeterMap)
map = mapCltn.FirstMap
layer.HeightMapName = map.Name
ayer.HeightDataFieldIndex = VcGantt1.DetectFieldIndex("Maindata", "LayerHeight")
VcGantt1.LayerCollection.Update()
```

## Example Code C#

```
VcLayer layer = vcGantt1.LayerCollection.FirstLayer();
VcMapCollection mapCltn = vcGantt1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcMillimeterMap);
VcMap 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.

	Data Type	Explanation
Property value	System.Int16	Horizontal offset in % -50 ... 50

## LabelSizeDependence

Property of VcLayer

This property lets you set or retrieve, whether and how the size of the label is to be dependent on the size of the layer.

	Data Type	Explanation
Property value	VcLabelSizeDependence  <b>Possible Values:</b> .vcFixedToBar 1 .vcTextHeightIndependent 39 .vcTextWidthIndependent 40	Dependence of the label on the layer size  restricted by layer siz independent on text height independent on text width

**Example Code VB.NET**

```
Dim layerCltn As VcLayerCollection
Dim layer As VcLayer

layerCltn = VcGantt1.LayerCollection
layer = layerCltn.LayerByName("Start End")
layer.LabelSizeDependence = VcLabelSizeDependence.vcFixedToBar
```

**Example Code C#**

```
VcLayerCollection layerCltn = vcGantt1.LayerCollection;
VcLayer layer = layerCltn.LayerByName("Start-End");
layer.LabelSizeDependence = VcLabelSizeDependence.vcFixedToBar;
```

**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
<b>Property value</b>	System.String	Legend text of the layer <b>Default value:</b> " " (content of the property <b>Name</b> )

**LineColor****Property of VcLayer**

This property lets you set or retrieve the color of the (border) line of the layer.

	Data Type	Explanation
<b>Property value</b>	System.Drawing.Color	RGB color values {0...255},{0...255},{0...255}

**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
Property value	System.Int16	Data field index

## 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
Property value	System.String	Name of the color map

## 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	System.Int16	Line thickness  LineType {1...4}: line thickness in pixels  LineType {5...1000}: line thickness in 1/100 mm <b>Default value:</b> As defined in the dialog

## 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	VcLineType	Line type  ({0...255},{0...255},{0...255})
	<b>Possible Values:</b>	
	.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
------------------	-------------

## Movable

Property of VcLayer

This property lets you set or retrieve whether a layer can be moved interactively.

	Data Type	Explanation
Property value	System.Boolean	Movable (True)/ not Movable (False) <b>Default value:</b> True

### Example Code VB.NET

```
Dim layer As VcLayer

layer = VcGantt1.LayerCollection.LayerByName("layer1")
layer.Movable = False
```

### Example Code C#

```
VcLayer VcLayer = vcGantt1.LayerCollection.LayerByName("layer1");
layer.Movable = false;
```

## Name

Read Only Property of VcLayer

This property lets you retrieve the name of a layer.

	Data Type	Explanation
Property value	System.String	Name of the layer

### Example Code VB.NET

```
Dim layerCltn As VcLayerCollection
Dim layer As VcLayer

layerCltn = VcGantt1.LayerCollection
For Each layer In layerCltn
    ListBox1.Items.Add(layer.Name)
Next
```

### Example Code C#

```
VcLayerCollection layerCltn = vcGantt1.LayerCollection;
foreach(VcLayer layer in layerCltn)
    listBox1.Items.Add(layer.Name);
```

## ObjectDrawEventsEnabled

Property of VcLayer

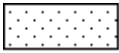


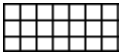


If this property is set to **true**, the events **VcObjectDrawn** and **VcObjectDrawing** are enabled for nodes which are drawn with this layer.


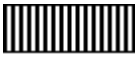
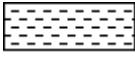
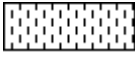

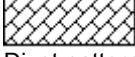
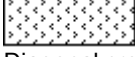

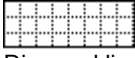


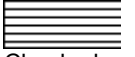



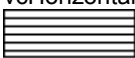

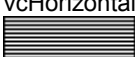
	Data Type	Explanation
Property value	System.Boolean	ObjectDraw events enabled (True) or disabled (False) <b>Default value:</b> False

## Pattern

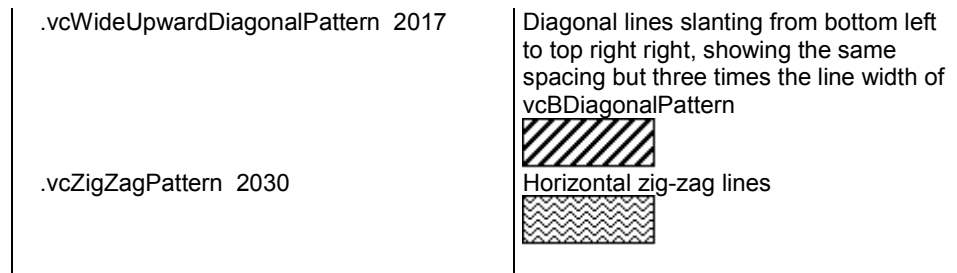
Property of VcLayer

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 dependent on the data.

	Data Type	Explanation
Property value	VcFillPattern	Pattern type <b>Default value:</b> As defined in the dialog
	<b>Possible Values:</b> .vc05PercentPattern... vc90PercentPattern 01 - 11  .vcAeroGlassPattern 44	Dots in foreground color on background color, the density of the foreground color 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 twice the line width 
.vcDashedHorizontalPattern 2026	Dashed horizontal lines 
.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 
.vcHorizontalPattern 3	Horizontal lines 
.vcLargeCheckerboardPattern 2044	Checkerboard pattern showing squares of twice the size of vcSmallCheckerBoardPattern 
.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 
.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
.vcVerticalPattern 2	Vertical lines
.vcVerticalTopLightedConvexPattern 42	Vertical color gradient from bright to dark
.vcWavePattern 2031	Horizontal waves pattern
.vcWeavePattern 2034	Interwoven stripes pattern
.vcWideDownwardDiagonalPattern 2016	Diagonal lines slanting from top left to bottom right, showing the same spacing but three times the line width of vcDiagonalPattern



## PatternColor

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.

If by the property **PatternColorMapName** a map was specified, the map will set the pattern in dependence of data.

	Data Type	Explanation
Property value	System.Drawing.Color	RGB color values {0...255},{0...255},{0...255}

## 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
Property value	System.Int16	Data field index

## 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 **PatternColor** will be used.

	Data Type	Explanation
Property value	System.String	Name of the color map

## PatternDataFieldIndex

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	System.Int16	Data field index

## 726 API Reference: VcLayer

### Example Code VB.NET

```
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

dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
dataRecCltn = dataTable.DataRecordCollection
dataRec1 = dataRecCltn.Add("1;Node 1;01.01.14;;8;Horizontal")
VcGantt1.EndLoading()

mapCltn = VcGantt1.MapCollection
map = mapCltn.Add("MapPattern")

map.Type = VcMapType.vcPatternMap
mapEntry = map.CreateEntry
mapEntry.DataFieldValue = "Diagonal"
mapEntry.Pattern = VcFillPattern.vcFDiagonalPattern
mapEntry = map.CreateEntry
mapEntry.DataFieldValue = "Horizontal"
mapEntry.Pattern = VcFillPattern.vcHorizontalPattern
mapCltn.Update()

layer = VcGantt1.LayerCollection.LayerByIndex(0)
layer.PatternMapName = "MapPattern"
layer.PatternDataFieldIndex = 5
VcGantt1.LayerCollection.Update()
```

### Example Code C#

```
VcDataTable dataTable =
vcGantt1.DataTableCollection.DataTableByName("Maindata");
VcDataRecordCollection dataRecCltn = dataTable.DataRecordCollection;

VcDataRecord dataRec1 = dataRecCltn.Add("1;Node 1;02.01.14;;8;Horizontal");
vcGantt1.EndLoading();

VcMapCollection mapCltn = vcGantt1.MapCollection;
VcMap map = mapCltn.Add("MapPattern");
map.Type = VcMapType.vcPatternMap;

VcMapEntry mapEntry = map.CreateEntry();
mapEntry.DataFieldValue = "Diagonal";
mapEntry.Pattern = VcFillPattern.vcFDiagonalPattern;
mapEntry = map.CreateEntry();
mapEntry.DataFieldValue = "Horizontal";
mapEntry.Pattern = VcFillPattern.vcHorizontalPattern;
mapCltn.Update();

VcLayer layer = vcGantt1.LayerCollection.LayerByIndex(0);
layer.PatternMapName = "MapPattern";
layer.PatternDataFieldIndex = 5;
vcGantt1.LayerCollection.Update();
```

## PatternMapName

### 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 **PatternDataFieldIndex**, 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.

	<b>Data Type</b>	<b>Explanation</b>
<b>Property value</b>	System.String	Name of the pattern map

### Example Code VB.NET

```
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

dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
dataRecCltn = dataTable.DataRecordCollection
dataRec1 = dataRecCltn.Add("1;Node 1;01.01.14;;8;Horizontal")
VcGantt1.EndLoading()

mapCltn = VcGantt1.MapCollection
map = mapCltn.Add("MapPattern")

map.Type = VcMapType.vcPatternMap
mapEntry = map.CreateEntry
mapEntry.DataFieldValue = "Diagonal"
mapEntry.Pattern = VcFillPattern.vcFDiagonalPattern
mapEntry = map.CreateEntry
mapEntry.DataFieldValue = "Horizontal"
mapEntry.Pattern = VcFillPattern.vcHorizontalPattern
mapCltn.Update()

layer = VcGantt1.LayerCollection.LayerByIndex(0)
layer.PatternMapName = "MapPattern"
layer.PatternDataFieldIndex = 5
VcGantt1.LayerCollection.Update()
```



## 728 API Reference: VcLayer

### Example Code C#

```
VcDataTable dataTable =
vcGantt1.DataTableCollection.DataTableByName("Maindata");
VcDataRecordCollection dataRecCltn = dataTable.DataRecordCollection;

VcDataRecord dataRec1 = dataRecCltn.Add("1;Node 1;02.01.14;;;8;Horizontal");
vcGantt1.EndLoading();

VcMapCollection mapCltn = vcGantt1.MapCollection;
VcMap map = mapCltn.Add("MapPattern");
map.Type = VcMapType.vcPatternMap;

VcMapEntry mapEntry = map.CreateEntry();
mapEntry.DataFieldValue = "Diagonal";
mapEntry.Pattern = VcFillPattern.vcFDiagonalPattern;
mapEntry = map.CreateEntry();
mapEntry.DataFieldValue = "Horizontal";
mapEntry.Pattern = VcFillPattern.vcHorizontalPattern;
mapCltn.Update();

VcLayer layer = vcGantt1.LayerCollection.LayerByIndex(0);
layer.PatternMapName = "MapPattern";
layer.PatternDataFieldIndex = 5;
vcGantt1.LayerCollection.Update();
```

## Shape

Property of VcLayer

This property lets you set or retrieve the shape of the layer.

Property value	Data Type	Explanation
	VcLayerShape	Layer shape
	<b>Possible Values:</b> .vcBitmapLayer 103007 .vcInvisibleSymbolLayer 101000 .vcLineLayer 2 .vcRectangleLayer 1 .vcSymbolLayer1 101001 .vcSymbolLayer10 101010 .vcSymbolLayer11 101032 .vcSymbolLayer12 101033 .vcSymbolLayer13 101034 .vcSymbolLayer14 101035 .vcSymbolLayer15 101036 .vcSymbolLayer16 101037 .vcSymbolLayer17 101038 .vcSymbolLayer18 101039 .vcSymbolLayer19 101040 .vcSymbolLayer2 101002 .vcSymbolLayer20 101041 .vcSymbolLayer21 101042 .vcSymbolLayer22 103001 .vcSymbolLayer3 101003 .vcSymbolLayer4 101004 .vcSymbolLayer5 101005 .vcSymbolLayer6 101006 .vcSymbolLayer7 101007 .vcSymbolLayer8 101008 .vcSymbolLayer9 101009	

.vcTriangleBottomLeftLayer 1566
.vcTriangleBottomRightLayer 1564

## Sizeable

Property of VcLayer

This property lets you set or retrieve whether the layer size can be changed interactively.

	Data Type	Explanation
Property value	VcLayerSizeability	Mode of layer sizeability <b>Default value:</b> True

### Example Code VB.NET

```
Dim layer As VcLayer

layer = vcGantt1.LayerCollection.LayerByName("layer1")
layer.Sizeable = VcLayerSizeability.vcSizeableLeftRight
```

### Example Code C#

```
VcLayer VcLayer = VcGantt1.LayerCollection.LayerByName("layer1");
layer.Sizeable = VcLayerSizeability.vcSizeableLeftRight;
```

## 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	System.String	Specification of the layer

## 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	System.Int16	Index of the data field that contains the start value

### ThreeDEffect

Property of VcLayer

This property lets you set or retrieve whether the layer will be highlighted by a 3D effect.

	Data Type	Explanation
Property value	System.Boolean	3D effect switched on (True)/switched off (False) <b>Default value:</b> False

### 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. Also see **OverlapLayerEnabled** and **OverlapLayerName** at the Gantt object.

	Data Type	Explanation
Property value	System.Boolean	<b>True:</b> layer is used as an overlap layer; <b>False:</b> layer is not used as an overlap layer. <b>Default value:</b> False

#### Example Code VB.NET

```
Dim layer As VcLayer

layer = VcGantt1.LayerCollection.LayerByName("layer1")
layer.UsedAsOverlapLayer = False
```

#### Example Code C#

```
VcLayer VcLayer = vcGantt1.LayerCollection.LayerByName("layer1");
layer.UsedAsOverlapLayer = false;
```

## 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 dependent on the data.

	Data Type	Explanation
Property value	System.Int32	Vertical offset. Unit: 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	System.Int16	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 **VerticalOffsetDataFieldIndex**, 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	System.String	Name of the millimetre map

## Visible

Property of VcLayer

This property lets you set or retrieve whether a layer is visible.

	Data Type	Explanation
Property value	System.Boolean	Layer visible/invisible <b>Default value:</b> True

### Example Code VB.NET

```
Dim layerCltn As VcLayerCollection
Dim layer As VcLayer

layerCltn = VcGantt1.LayerCollection
layer = layerCltn.LayerByName("Start End")
layer.Visible = False
```

### Example Code C#

```
VcLayerCollection layerCltn = vcGantt1.LayerCollection;
VcLayer 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	System.Boolean	Layer visible in legend (True)/ invisible in legend (False) <b>Default value:</b> True

### Example Code VB.NET

```
Dim layerCltn As VcLayerCollection
Dim layer As VcLayer

layerCltn = VcGantt1.LayerCollection
layer = layerCltn.LayerByName("Start End")
layer.VisibleInLegend = False
```

### Example Code C#

```
VcLayerCollection layerCltn = vcGantt1.LayerCollection;
VcLayer layer = layerCltn.LayerByName("Start-End");
layer.VisibleInLegend = false;
```

## Methods

### CalculateCurrentWidth

**Method of VcLayer**

This method calculates 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
<b>Parameter:</b> ⇒ node	VcNode	Node, in the layer definition of which the layer is looked for.
<b>Return value</b>	System.Int32	Width of the layer in 1/100 mm

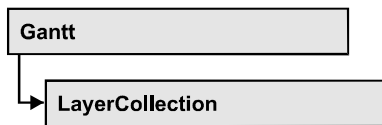
### PutInOrderAfter

**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
<b>Parameter:</b> ⇐ refName	System.String	Name of the layer, after which the current layer shall be placed.
<b>Return value</b>	Void	

## 6.40 VcLayerCollection



The LayerCollection object automatically contains all available layers . You can access all objects in an iterative loop by **For Each layer In LayerCollection** 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

- Count

### Methods

- Add
- AddBySpecification
- Copy
- FirstLayer
- GetEnumerator
- LayerByIndex
- LayerByName
- NextLayer
- Remove
- Update

---

## Properties

### Count

**Read Only Property of VcLayerCollection**

This property lets you retrieve the number of layers in the layer collection.

	Data Type	Explanation
Property value	System.Int32	Number of layers

**Example Code VB.NET**

```
Dim numberOfLayers As Integer

numberOfLayers = VcGantt1.LayerCollection.Count
```

**Example Code C#**

```
int 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 has not been used before, the new layer object will be returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
<b>Parameter:</b> ⇒ layerName	System.String	Layer name
<b>Return value</b>	VcLayer	New layer object

**Example Code VB.NET**

```
newlayer = VcGantt1.LayerCollection.Add("test1")
```

**Example Code C#**

```
VcLayer 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
<b>Parameter:</b> ⇒ specification	System.String	Specification of the layer



<b>Return value</b>	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
<b>Parameter:</b>		
⇒ layerName	System.String	Name of the layer to be copied
⇒ newLayerName	System.String	Name of the new layer
<b>Return value</b>	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
<b>Return value</b>	VcLayer	First Layer

#### Example Code VB.NET

```
Dim layerCltn As VcLayerCollection
Dim layer As VcLayer

layerCltn = VcGantt1.LayerCollection
layer = layerCltn.FirstLayer
```

#### Example Code C#

```
VcLayerCollection layerCltn = vcGantt1.LayerCollection;
VcLayer layer = layerCltn.FirstLayer();
```

## GetEnumerator

Method of VcLayerCollection

This method returns an Enumerator object which supports the iteration by language specific elements. It is implied in the For...Each construct of Visual Basic and C#. This object allows to iterate over the layer objects included.

	Data Type	Explanation
<b>Return value</b>	VcObject	Reference object

## LayerByIndex

Method of VcLayerCollection

This method lets you access a layer by its index. If a layer does not exist at the index specified, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
<b>Parameter:</b> ⇒ index	System.Int16	Index of the layer
<b>Return value</b>	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
<b>Parameter:</b> ⇒ layerName	System.String	Name of layer
<b>Return value</b>	VcLayer	Layer

### Example Code VB.NET

```
Dim layerCltn As VcLayerCollection
Dim layer As VcLayer

layerCltn = VcGantt1.LayerCollection
layer = layerCltn.LayerByName("Start-End")
```

## 738 API Reference: VcLayerCollection

### Example Code C#

```
VcLayerCollection layerCltn = vcGantt1.LayerCollection;  
VcLayer 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	Next Layer

### Example Code VB.NET

```
Dim layerCltn As VcLayerCollection  
Dim layer As VcLayer  
  
layerCltn = VcGantt1.LayerCollection  
layer = layerCltn.FirstLayer  
While Not layer Is Nothing  
    ListBox1.Items.Add(layer.Name)  
    layer = layerCltn.NextLayer  
End While
```

### Example Code C#

```
VcLayerCollection layerCltn = vcGantt1.LayerCollection;  
VcLayer layer = layerCltn.FirstLayer();  
  
while (layer != null)  
{  
    listBox1.Items.Add(layer.Name);  
    layer = layerCltn.NextLayer();  
}
```

## 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
<b>Parameter:</b> ⇒ layerName	System.String	Layer name
<b>Return value</b>	System.Boolean	Layer deleted (True)/not deleted (False)

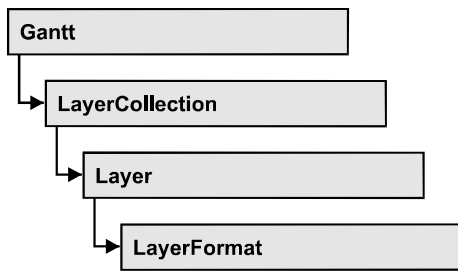
## Update

### Method of VcLayerCollection

This method lets you update a layer collection after having modified it.

	Data Type	Explanation
<b>Return value</b>	System.Boolean	Update successful (True)/ not successful (False)

## 6.41 VcLayerFormat



A layer format specifies the annotation of layers. With **For Each formatfield In LayerFormat** you can retrieve all layers.

### Properties

- FormatField
- FormatFieldCount

### Methods

- CopyFormatField
- GetEnumerator
- RemoveFormatField

---

## Properties

### 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.

The property FormatField is an Indexed Property, which in C# is addressed by the methods set\_FormatField (index, pvn) and get\_FormatField (index) .

	Data Type	Explanation
<b>Parameter:</b> index	System.Int16	Index of the layer format field 0 ... FormatFieldCount-1
<b>Property value</b>	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	System.Int16	Number of fields of the layer format

## 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
<b>Parameter:</b> ⇨ position	VcFormatFieldPosition  <b>Possible Values:</b> .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	System.Int16	Index of the reference layer format field
<b>Return value</b>	VcLayerFormatField	Layer format field object

### GetEnumerator

Method of VcLayerFormat

This method returns an Enumerator object. It is implied in the For...Each construct of Visual Basic and C#.NET which supports the iteration by language specific elements. This object allows to iterate over the layer objects included.

## 742 API Reference: VcLayerFormat

	Data Type	Explanation
Return value	VcObject	Reference object

### RemoveFormatField

Method of VcLayerFormat

This method lets you remove a layer format field by its index. After that, the program will set all layer format field indexes newly in order to number them consecutively.

	Data Type	Explanation
<b>Parameter:</b> ⇒ index	System.Int16	Index of the layer format field to be deleted

## 6.42 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 do, but it has an index that defines its position in the layer format.

### Properties

- Alignment
- ConstantText
- FormatName
- Index
- MinimumWidth
- Priority
- TextDataFieldIndex
- TextFont
- TextFontColor
- TextFontColorDataFieldIndex
- TextFontColorMapName
- TextFontDataFieldIndex
- TextFontMapName
- TextLineCount
- TextLineCountDataFieldIndex
- TextLineCountMapName
- TruncatedTextSuppressed

### Methods

- CalculateLineCount

---

## Properties

### Alignment

**Property of VcLayerFormatField**

This property lets you set or retrieve the alignment of the content of the layer format field.



## 744 API Reference: VcLayerFormatField

	Data Type	Explanation
Property value	VcFormatFieldAlignment  <b>Possible Values:</b> .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

### 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	System.String	Constant text

### FormatName

Read Only Property of VcLayerFormatField

This property lets you retrieve the name of the layer format to which this field belongs.

	Data Type	Explanation
Property value	System.String	Name of the layer format

### Index

Read Only Property of VcLayerFormatField

This property lets you retrieve the index of the layer format field in the associated layer format.

	Data Type	Explanation
Property value	System.Int16	Index of the layer format field

## MinimumWidth

### Property of VcLayerFormatField

This property lets you set or retrieve the minimum width of the layer format field in mm if the label size dependence allows it.

	Data Type	Explanation
Property value	System.Int16	Minimum width (in mm) of the layer format field 0 ... 99

## Priority

### Property of VcLayerFormatField

This property lets you set 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 probability to get the space necessary.

	Data Type	Explanation
Property value	System.Int16	priority of the layer format field {-9...9}

## 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	System.Int16	Index of the data field

## 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 was set, the map will control the text font color in dependence of the data.

	Data Type	Explanation
Property value	Font	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 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	System.Drawing.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	System.Int16	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 format field is of the type **vcFFText**. 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 selected from the map. If no map entry applies, the font color specified by the property **TextFontColor** will be used.

	Data Type	Explanation
Property value	System.String	Name of the font color map

## TextFontDataFieldIndex

Property of VcLayerFormatField

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	System.Int16	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**, the font will be selected from the map. If no data field entry applies, the font that is specified by the property **TextFont** will be used.

	Data Type	Explanation
Property value	System.String	Name of the font 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	System.Int16	Number of lines

## TextLineCountDataFieldIndex

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 **TextLineCountMapName**. If you set this property to **-1**, no map will be used.

	Data Type	Explanation
Property value	System.Int16	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 set to "", no map will be used. If a map name and additionally a data field index is specified by the property **TextLineCountDataFieldIndex**, the corresponding number of text lines will be selected from the map. If no data field entry applies, the number of text lines specified by the property **TextLineCount** will be used.

	Data Type	Explanation
Property value	System.String	Name of the numeric map

## TruncatedTextSuppressed

Property of VcLayerFormatField

This property lets you set or retrieve, whether text which does not fit completely in the layer format field is to be suppressed or clipped.

	Data Type	Explanation
Property value	System.Boolean	Property active (True)/ not active (False)

---

## Methods

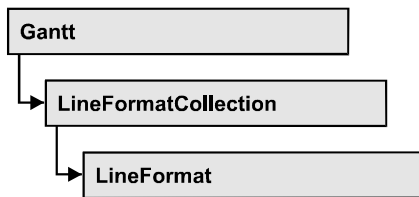
### CalculateLineCount

Method of VcLayerFormatField

**For outside 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 inside 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
<b>Parameter:</b> ⇒ node	VcNode	Node
<b>Return value</b>	System.Int32	Calculated number of lines

## 6.43 VcLineFormat



An object of the type VcLineFormat defines the contents and the appearance of lines, for example in a date line grid.

### Properties

- FormatField
- FormatFieldCount
- Name
- Specification

### Methods

- CopyFormatField
- RemoveFormatField

---

## Properties

### FormatField

**Read Only Property of VcLineFormat**

This property lets you retrieve a VcLineFormatField object by its index. The index has to be in the range from 0 to FormatFieldCount-1.

	Data Type	Explanation
<b>Parameter:</b> index	System.Int16	Index of the line format field
<b>Property value</b>	VcNodeFormatField	Line format field

## FormatFieldCount

**Read Only Property of VcLineFormat**

This property allows to determine the number of fields in a line format.

	Data Type	Explanation
Property value	System.Int16	Number of fields of the line format

### Example Code VB.NET

```
Dim lineFormat As VcLineFormat

lineFormat = VcGanttASP1.LineFormatCollection.FirstFormat
MessageBox(lineFormat.FormatFieldCount)
```

### Example Code C#

```
VcLineFormat nodeFormat = vcGanttASP1.LineFormatCollection.FirstFormat();
MessageBox.Show(lineFormat.FormatFieldCount.ToString());
```

## Name

**Property of VcLineFormat**

This property lets you set or retrieve the name of the line format.

	Data Type	Explanation
Property value	System.String	Name of the line format

### Example Code VB.NET

```
Dim lineFormat As VcLineFormat

lineFormat = VcGanttASP1.LineFormatCollection.FirstFormat
MessageBox(lineFormat.Name)
```

### Example Code C#

```
VcLineFormat lineFormat = vcGanttASP1.LineFormatCollection.FirstFormat();
MessageBox.Show(lineFormat.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 smoothly be stored to text files or data bases. This allows for persistency. A specification can be used to create a line format by the method **VcLineFormatCollection.AddBySpecification**.



	Data Type	Explanation
Property value	System.String	Specification of the line format

## Methods

### CopyFormatField

Method of VcLineFormat

This method allows to copy a line format field, returning the new VcLineFormatField object. It contains the next consecutive unused index.

	Data Type	Explanation
<b>Parameter:</b> ⇨ position	VcFormatFieldPosition  <b>Possible Values:</b> .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	System.Int16	Index of the reference line format field
<b>Return value</b>	VcNodeFormatField	Line format field object

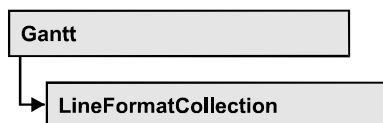
### RemoveFormatField

Method of VcLineFormat

This method lets you remove a line format field by its index. After that, the program will re-set all node format field indexes in order to number them consecutively.

	Data Type	Explanation
<b>Parameter:</b> ⇨ index	System.Int16	Index of the line format field to be deleted

## 6.44 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 **FormatByName** 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

- Count

### Methods

- Add
- AddBySpecification
- Copy
- FirstFormat
- FormatByIndex
- FormatByName
- NextFormat
- Remove

---

## Properties

### 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	System.Int32	Number of line formats

### Example Code VB.NET

```
Dim lineFormatCltn As VcLineFormatCollection
Dim numberOfLineformats As Integer

lineFormatCltn = VcGanttASPl.LineFormatCollection
numberOfLineformats = lineFormatCltn.Count
```

### Example Code C#

```
VcLineFormatCollection lineFormatCltn = vcGanttASPl.LineFormatCollection;
int numberOfLineformats = lineFormatCltn.Count;
```

---

## Methods

### Add

#### Method of VcLineFormatCollection

By this method you can create a line format as a member of the LineFormatCollection. If the name has not been used before, the new line object will be returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
<b>Parameter:</b> ⇒ formatName	System.String	Line format name
<b>Return value</b>	VcLineFormat	New line format object

### Example Code VB.NET

```
Dim newLineFormat = VcGanttASPl.LineFormatCollection.Add("lineFormat1")
```

### Example Code C#

```
newLineFormat = vcGanttASPl.LineFormatCollection.Add("lineFormat1");
```

### AddBySpecification

#### Method of VcLineFormatCollection

This method lets you create a line format by using a line 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
<b>Parameter:</b> ⇒ formatSpecification	System.String	Line format specification
<b>Return value</b>	VcLineFormat	New line format object

## Copy

### Method of VcLineFormatCollection

By this method you can copy a line format. If the line format to be copied exists, and if the name for the new line format does not yet exist, the new line format object will be returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
<b>Parameter:</b> ⇒ newFormatName	System.String	Name of the new line format
<b>Return value</b>	VcLineFormat	Line format object

### Example Code VB.NET

```
Dim lineFormatCltn As VcLineFormatCollection
Dim lineFormat As VcLineFormat

lineFormatCltn = VcGanttASp1.LineFormatCollection
lineFormat = lineFormatCltn.Copy("CurrentLineFormat", "NewLineFormat")
```

### Example Code C#

```
VcLineFormatCollection lineFormatCltn = vcGanttASp1.LineFormatCollection;
VcLineFormat 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
<b>Return value</b>	VcLineFormat	First line format

## 756 API Reference: VcLineFormatCollection

### Example Code VB.NET

```
Dim format As VcLineFormat  
  
format = VcGanttASPl.LineFormatCollection.FirstFormat
```

### Example Code C#

```
VcLineFormat format = vcGanttASPl.LineFormatCollection.FirstFormat();
```

## FormatByIndex

### Method of VcLineFormatCollection

This method lets you access a line format by its index. If a line format does not exist at the index specified, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcLineFormat	Line format object returned

### Example Code VB.NET

```
Dim formatLineCltn As VcLineFormatCollection  
Dim formatLine As VcLineFormat  
  
formatLineCltn = VcGanttASPl.LineFormatCollection  
formatLine = formatLineCltn.FormatByIndex(2)
```

### Example Code C#

```
VcLineFormatCollection lineFormatCltn = vcGanttASPl.LineFormatCollection;  
VcLineFormat 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
<b>Parameter:</b> ⇒ formatName	System.String	Name of the line format
<b>Return value</b>	VcLineFormat	Line format

**Example Code VB.NET**

```
Dim formatLineCltn As VcLineFormatCollection
Dim formatLine As VcLineFormat

formatLineCltn = VcGanttASPl.LineFormatCollection
formatLine = formatLineCltn.FormatByName("Standard")
```

**Example Code C#**

```
VcLineFormatCollection lineFormatCltn = vcGanttASPl.LineFormatCollection;
VcLineFormat format = lineFormatCltn.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
<b>Return value</b>	VcLineFormat	Subsequent line format

**Example Code VB.NET**

```
Dim formatLineCltn As VcLineFormatCollection
Dim formatLine As VcLineFormat

formatLineCltn = VcGanttASPl.LineFormatCollection
formatLine = formatLineCltn.FirstFormat

While Not formatLine Is Nothing
    ListLine1.Items.Add(formatLine.Name)
    formatLine = formatLineCltn.NextFormat
End While
```

**Example Code C#**

```
VcLineFormatCollection lineFormatCltn = vcGanttASPl.LineFormatCollection;
VcLineFormat lineFormat = lineFormatCltn.FirstFormat();

while (lineFormat != null)
{
    ListLine.Items.Add(lineFormat.Name);
    lineFormat = lineFormatCltn.NextFormat();
}
```

**Remove****Method of VcLineFormatCollection**

This method lets you delete a line format. If the line format is used by another object, it cannot be deleted. Then **False** will be returned, otherwise **True**.

## 758 API Reference: VcLineFormatCollection

	Data Type	Explanation
<b>Parameter:</b> ⇒ FormatName	System.String	Line format name
<b>Return value</b>	System.Boolean	Line format deleted (True) / not deleted (False)

### Example Code VB.NET

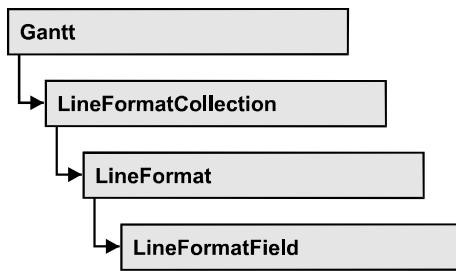
```
Dim lineFormatCltn As VcLineFormatCollection
Dim lineFormat As VcLineFormat

lineFormatCltn = VcGanttASp1.LineFormatCollection
lineFormat = lineFormatCltn.FormatByIndex(1)
lineFormatCltn.Remove(lineFormat.Name)
```

### Example Code C#

```
VcLineFormatCollection lineFormatCltn = vcGanttASp1.LineFormatCollection;
VcLineFormat lineFormat = lineFormatCltn.FormatByIndex(1);
lineFormatCltn.Remove(lineFormat.Name);
```

## 6.45 VcLineFormatField



An object of the type `VcLineFormatField` represents a field of a `VcLineFormat`-Object. A line format field does not have a name as many other objects, but it has an index that defines its position in the line format.

### Properties

- Alignment
- BackgroundColor
- BackgroundColorDataFieldIndex
- BackgroundColorMapName
- ConstantText
- DateOutputFormat
- FormatName
- Index
- Pattern
- TextDataFieldIndex
- 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	VcFormatFieldAlignment	Alignment of the field content
	<b>Possible Values:</b> .vcFFABottom 28 .vcFFABottomLeft 27 .vcFFABottomRight 29 .vcFFACenter 25 .vcFFALeft 24 .vcFFARight 26 .vcFFATop 22 .vcFFATopLeft 21 .vcFFATopRight 23	Bottom Bottom left Bottom right Center Left Right Top Top left Top right

## BackgroundColor

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.

If the line format field shall have the color of the line format, select the value -1.

If by the property **BackgroundColorMapName** a map was specified, the map will set the background color in dependence on data.

	Data Type	Explanation
Property value	System.Drawing.Color	RGB color values {{0...255},{0...255},{0...255}} <b>Default value:</b> -1

## BackgroundColorDataFieldIndex

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 **BackgroundColorMapName**. If you set this property to **-1**, no map will be used.

	Data Type	Explanation
Property value	System.Int16	Data field index

## BackgroundColorMapName

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. If you specify a map name and in addition set a data field index by the property **BackgroundColorDataFieldIndex**, then the background color will be set by the map. If none of the map entries applies, the background color specified by the property **BackgroundColor** will apply.

	Data Type	Explanation
Property value	System.String	Name of the color map

## 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 **TextDataFieldIndex** was set to -1.

	Data Type	Explanation
Property value	System.String	Constant text

## DateOutputFormat

Property of VcLineFormatField

This property lets you set or retrieve the date output format. To compose the date you can use the below codes:

- D: first character 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:	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 <b>VcTextEntrySupplying</b> )
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 <b>VcTextEntrySupplying</b> )
Q:	one-digit figure for the quarter: 1-4
TQ:	name of quarter (adjustable by using the event <b>VcTextEntrySupplying</b> )
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 <b>VcTextEntrySupplying</b> )
TH:	"am" or "pm" (adjustable by using the event <b>VcTextEntrySupplying</b> )
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 a prefix.

	Data Type	Explanation
Property value	System.String {DMYhms:;}	Date

**Example Code VB.NET**

```
VcGantt1.DateOutputFormat = "DD.MM.YY"
```

**Example Code C#**

```
vcGantt1.DateOutputFormat = "DD.MM.YY";
```

**FormatName****Read Only Property of VcLineFormatField**

This property lets you retrieve the name of the line format to which this field belongs.

	Data Type	Explanation
Property value	System.String	Name of the line format

**Index****Read Only Property of VcLineFormatField**

This property lets you retrieve the index of the line format field in the associated line format.

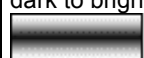
	Data Type	Explanation
Property value	System.Int16	Index of the line format field

**Pattern****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	VcFieldFillPattern  Possible Values:	Pattern type

.vcAeroGlassPattern 44	Vertical color gradient in the color of the fill pattern
.vcFieldNoPattern 1276	No fill pattern
.vcFieldVerticalBottomLightedConvexPattern 43	Vertical color gradient from bright to dark
.vcFieldVerticalConcavePattern 40	Vertical color gradient from dark to bright to dark
.vcFieldVerticalConvexPattern 41	Vertical color gradient from bright to dark to bright
.vcFieldVerticalTopLightedConvexPattern 42	Vertical color gradient from dark to bright



## TextDataFieldIndex

Property of VcLineFormatField

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. This property only works if the type of the data field is **vcFFTText**. If you set the value of the index to -1, the content of the property **ConstantText** will be returned instead.

	Data Type	Explanation
Property value	System.Int16	Index of the data field

## 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 on the data.

	Data Type	Explanation
Property value	System.Drawing.Color {True}	Font color of the line format

## 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	System.Int16 {True}	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 by the property **TextFontColor** will be used.

	Data Type	Explanation
Property value	System.String	Name of the font 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	System.Int16	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 by the property **TextFont** will be used.

	Data Type	Explanation
Property value	System.String	Name of the font map

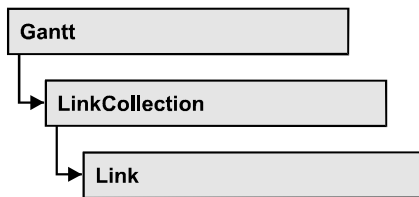
## TextLineCount

Property of VcLineFormatField

This property lets you set or retrieve the number of lines, if the size of the annotation field allows for it

	Data Type	Explanation
Property value	System.Int16	Number of lines

## 6.46 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
- Delete
- RelatedDataRecord
- Update

---

## 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	System.String	All data of the link



### Example Code VB.NET

```
Dim linkCltn As VcLinkCollection
Dim link As VcLink
Dim allDataOfLink As String

linkCltn = VcGanttASP1.LinkCollection
link = linkCltn.FirstLink
allDataOfLink = link.AllData
```

### Example Code C#

```
VcLinkCollection linkCltn = vcGanttASP1.LinkCollection;
VcLink link = linkCltn.FirstLink();
string allDataOfLink = link.AllData.ToString();
```

## 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.

The property DataField is an Indexed Property, which in C# is addressed by the methods set\_DataField (index, pvn) and get\_DataField (index).

	Data Type	Explanation
<b>Parameter:</b>		
⇒ index	System.Int16	Index of the data field
<b>Property value</b>	System.Object	Content of data field

### Example Code VB.NET

```
Dim linkCltn As VcLinkCollection
Dim link As VcLink
Dim message As String

linkCltn = VcGanttASP1.LinkCollection
For Each link In linkCltn
    message = "Delete link from " + link.DataField(1) + " to " +
link.DataField(2) + " ?"
    If MsgBox(message, MsgBoxStyle.OKCancel, "Delete Link") = MsgBoxResult.OK
Then
        link.Delete()
    End If
Next
```

**Example Code C#**

```
VcLinkCollection linkCltn = vcGanttASP1.LinkCollection;

foreach (VcLink link in linkCltn)
{
    DialogResult retVal = MessageBox.Show("Delete link from " +
link.get_DataField(1) + " to " + link.get_DataField(2) + " ?", "Deleting curve
point", MessageBoxButtons.OKCancel);
    if (retVal == DialogResult.OK)
        link.Delete();
}
```

**ID****Read Only Property of VcLink**

By this property you can retrieve the ID of a link.

	Data Type	Explanation
Property value	System.String	Link ID

**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 VB.NET**

```
Dim linkCltn As VcLinkCollection
Dim link As VcLink
Dim node As VcNode
Dim nodeName As String

linkCltn = VcGanttASP1.LinkCollection
link = linkCltn.FirstLink
node = link.PredecessorNode
nodeName = node.DataField(1)
```

**Example Code C#**

```
VcLinkCollection linkCltn = vcGanttASP1.LinkCollection;
VcLink link = linkCltn.FirstLink();
VcNode node = link.PredecessorNode;
string nodeName = node.get_DataField(1).ToString();
```

## 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 VB.NET

```
Dim linkCltn As VcLinkCollection
Dim link As VcLink
Dim node As VcNode
Dim nodeName As String

linkCltn = VcGanttASPI.LinkCollection
link = linkCltn.FirstLink
node = link.SuccessorNode
nodeName = node.DataField(1)
```

### Example Code C#

```
VcLinkCollection linkCltn = vcGanttASPI.LinkCollection;
VcLink link = linkCltn.FirstLink();

VcNode node = link.SuccessorNode;
string nodeName = node.get_DataField(1).ToString();
```

---

## 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

### Delete

Method of VcLink

By this method you can delete a link.

	Data Type	Explanation
<b>Return value</b>	System.Boolean	Link was/was not successfully deleted

**Example Code VB.NET**

```
link.Delete()
```

**Example Code C#**

```
link.Delete();
```

**RelatedDataRecord****Method of VcLink**

This method 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
<b>Parameter:</b> ⇒ index	System.Int16	Index of data field that holds the key
<b>Return value</b>	VcDataRecord	Related data record returned

**Update****Method of VcLink**

When a data field of a link was edited by the **DataField** property, you can update the diagram by the **Update** method.

	Data Type	Explanation
<b>Return value</b>	System.Boolean	Link was/was not successfully updated

**Example Code VB.NET**

```
Dim linkCltn As VcLinkCollection
Dim link As VcLink

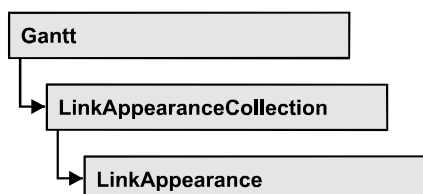
linkCltn = VcGanttASP1.LinkCollection
link = linkCltn.FirstLink
link.DataField(2) = 10
link.Update()
```

## 772 API Reference: VcLink

### Example Code C#

```
VcLinkCollection linkCltn = vcGanttASP1.LinkCollection;  
VcLink link = linkCltn.FirstLink();  
  
link.set_DataField(2, 10);  
link.Update();
```

## 6.47 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
- PredecessorPortSymbol
- RoutingType
- SuccessorLayerName
- SuccessorPortSymbol
- Visible

### Methods

- PutInOrderAfter

---

## Properties

### FilterName

**Read Only Property of VcLinkAppearance**

This property lets you retrieve the filter that is used for a link appearance. This property can be also set on the **Link** property page.

	Data Type	Explanation
Property value	System.String	Filter name

## 774 API Reference: VcLinkAppearance

### Example Code VB.NET

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance
Dim filterOfLinkApp As String

linkAppearanceCltn = VcGanttASP1.LinkAppearanceCollection
linkAppearance = linkAppearanceCltn.LinkAppearanceByName("Blue")
filterOfLinkApp = linkAppearance.FilterName
```

### Example Code C#

```
VcLinkAppearanceCollection linkAppearanceCltn =
vcGanttASP1.LinkAppearanceCollection;
VcLinkAppearance linkAppearance =
linkAppearanceCltn.LinkAppearanceByName("Blue");
string filterOfLinkApp = linkAppearance.FilterName;
```

## LineColor

### Property of VcLinkAppearance

This property lets you set or retrieve the line color of a LinkAppearance object.

This property can be also set in the **Line Attributes** dialog box that can be opened by the **Link** property page.

	Data Type	Explanation
Property value	System.Drawing.Color	RGB color values

### Example Code VB.NET

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance

linkAppearanceCltn = VcGanttASP1.LinkAppearanceCollection
linkAppearance = linkAppearanceCltn.LinkAppearanceByName("Blue")
linkAppearance.LineColor = Color.Blue
```

### Example Code C#

```
VcLinkAppearanceCollection linkAppearanceCltn =
vcGanttASP1.LinkAppearanceCollection;
VcLinkAppearance linkAppearance =
linkAppearanceCltn.LinkAppearanceByName("Blue");
linkAppearance.LineColor = Color.LightSteelBlue;
```

## 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
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 can be also set in the **Line Attributes** dialog box that can be opened by the **Link** property page.

	Data Type	Explanation
<b>Property value</b>	System.Int32	Line thickness LineType {1...4}: line thickness in pixels LineType {5...1000}: line thickness in 1/100 mm <b>Default value:</b> As defined on property page

#### Example Code VB.NET

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance

linkAppearanceCltn = VcGanttASP1.LinkAppearanceCollection
linkAppearance = linkAppearanceCltn.LinkAppearanceByName("Standard")
linkAppearance.LineThickness = 4
```

#### Example Code C#

```
VcLinkAppearanceCollection linkAppearanceCltn =
vcGanttASP1.LinkAppearanceCollection;
VcLinkAppearance linkAppearance =
linkAppearanceCltn.LinkAppearanceByName("Standard");
linkAppearance.LineThickness = 4;
```





**Example Code VB.NET**

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance

linkAppearanceCltn = VcGanttASP1.LinkAppearanceCollection
linkAppearance = linkAppearanceCltn.LinkAppearanceByName("Blue")
linkAppearance.LineType = 5
```

**Example Code C#**

```
VcLinkAppearanceCollection linkAppearanceCltn =
vcGanttASP1.LinkAppearanceCollection;
VcLinkAppearance linkAppearance =
linkAppearanceCltn.LinkAppearanceByName("Blue");
linkAppearance.LineType = VcLineType.vcLineType5;
```

**Name****Read Only Property of VcLinkAppearance**

This property lets you retrieve the name of a LinkAppearance object.

This property can also be set on the **Links** property page.

	Data Type	Explanation
Property value	System.String	Name

**Example Code VB.NET**

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance
Dim nameLinkApp As String

linkAppearanceCltn = VcGanttASP1.LinkAppearanceCollection
linkAppearance = linkAppearanceCltn.FirstLinkAppearance
nameLinkApp = linkAppearance.Name
```

**Example Code C#**

```
VcLinkAppearanceCollection linkAppearanceCltn =
vcGanttASP1.LinkAppearanceCollection;
VcLinkAppearance linkAppearance = linkAppearanceCltn.FirstLinkAppearance();
string 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	System.String	Character string that passes the layer name

## PredecessorPortSymbol

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	VcLinkPredecessorPortSymbol	Symbol on the predecessor node <b>Default value:</b> vcLPSNone

### Example Code VB.NET

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance

linkAppearanceCltn = VcGanttASP1.LinkAppearanceCollection
linkAppearance = linkAppearanceCltn.FirstLinkAppearance
linkAppearance.PredecessorPortSymbol =
VcLinkPredecessorPortSymbol.vcLPSDoubleSemiCircle
```

### Example Code C#

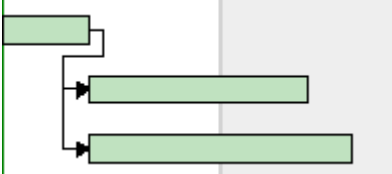
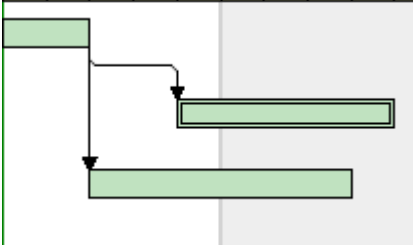
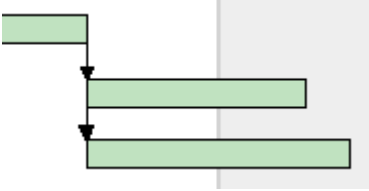
```
VcLinkAppearanceCollection linkAppearanceCltn =
vcGanttASP1.LinkAppearanceCollection;
VcLinkAppearance linkAppearance = linkAppearanceCltn.FirstLinkAppearance();
linkAppearance.PredecessorPortSymbol =
VcLinkPredecessorPortSymbol.vcLPSFilledDoubleSemiCircle;
```

## 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
<b>Property value</b>	VcRoutingType	Routing type <b>Default value:</b> vcLRTOrthogonal
	<b>Possible Values:</b> .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 can 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. 

## SuccessorLayerName

Property of VcLinkAppearance

This property lets you specify or retrieve to which layer of the successor 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
<b>Property value</b>	System.String	Character string that passes the layer name

## SuccessorPortSymbol

Property of VcLinkAppearance

This property lets you assign/retrieve a port symbol to a link, that 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	VcLinkSuccessorPortSymbol	Symbol on the successor node <b>Default value:</b> vcLSSNone

### Example Code VB.NET

```
VcLinkAppearanceCollection linkAppearanceCltn =
VcGanttASP1.LinkAppearanceCollection;
VcLinkAppearance linkAppearance = linkAppearanceCltn.FirstLinkAppearance();
linkAppearance.SuccessorPortSymbol =
VcLinkSuccessorPortSymbol.vcLSSFilledDoubleArrow;
```

### Example Code C#

```
VcLinkAppearanceCollection linkAppearanceCltn =
vcGanttASP1.LinkAppearanceCollection;
VcLinkAppearance linkAppearance = linkAppearanceCltn.FirstLinkAppearance();
linkAppearance.SuccessorPortSymbol =
VcLinkSuccessorPortSymbol.vcLSSFilledDoubleArrow;
```

## Visible

Property of VcLinkAppearance

This property lets you set or retrieve whether the link is visible.

This property can also be set on the **Links** property page.

	Data Type	Explanation
Property value	System.Boolean	Property active/not active <b>Default value:</b> True

### Example Code VB.NET

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance

linkAppearanceCltn = VcGanttASP1.LinkAppearanceCollection
linkAppearance = linkAppearanceCltn.FirstLinkAppearance
linkAppearance.Visible = False
```

**Example Code C#**

```
VcLinkAppearanceCollection linkAppearanceCltn =
vcGanttASP1.LinkAppearanceCollection;
VcLinkAppearance linkAppearance = linkAppearanceCltn.FirstLinkAppearance();
linkAppearance.Visible = false;
```

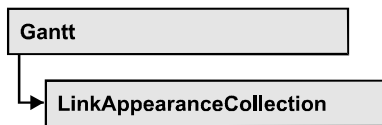
---

**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 appearance 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
<b>Parameter:</b> refLinkAppearanceName	System.String	Name of the link appearance behind which the current link appearance is to be set

## 6.48 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 **LinkAppearanceByIndex**. The number of link appearances in the collection object can be retrieved by the property **Count**.

### Properties

- Count

### Methods

- Add
- AddBySpecification
- Copy
- FirstLinkAppearance
- GetEnumerator
- LinkAppearanceByIndex
- LinkAppearanceByName
- NextLinkAppearance
- Remove
- Update

---

## Properties

### 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	System.Int32	Number of link appearance objects

**Example Code VB.NET**

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim numberOfLinkAppearance As Integer

linkAppearanceCltn = VcGanttASP1.LinkAppearanceCollection
numberOfLinkAppearance = linkAppearanceCltn.Count
```

**Example Code C#**

```
VcLinkAppearanceCollection linkAppearanceCltn =
vcGanttASP1.LinkAppearanceCollection;
int numberOfLinkAppearance = 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 link appearance object will be returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned. All attributes of the new link appearance by default are set to transparent.

	Data Type	Explanation
<b>Parameter:</b> ⇒ newName	System.String	Link appearance name
<b>Return value</b>	VcLinkAppearance	New link appearance object

**Example Code VB.NET**

```
newLinkAppearance = VcGanttASP1.LinkAppearanceCollection.Add("linkapp1")
```

**Example Code C#**

```
newLinkAppearance = vcGanttASP1.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 subsequent session the link appearance can be created again from the specification and is identified by its name.

	Data Type	Explanation
<b>Parameter:</b> ⇒ linkAppearanceSpecification	System.String	Link appearance specification
<b>Return value</b>	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" (Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
<b>Parameter:</b> ⇒ fromName	System.String	Name of the link appearance to be copied
⇒ newName	System.String	Name of the new link appearance
<b>Return value</b>	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
<b>Return value</b>	VcLinkAppearance	First linkAppearance object

**Example Code VB.NET**

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance

linkAppearanceCltn = VcGanttASP1.LinkAppearanceCollection
linkAppearance = linkAppearanceCltn.FirstLinkAppearance
```

**Example Code C#**

```
VcLinkAppearanceCollection linkAppearanceCltn =
vcGanttASP1.LinkAppearanceCollection;
VcLinkAppearance linkAppearance = linkAppearanceCltn.FirstLinkAppearance();
```

## GetEnumerator

**Method of VcLinkAppearanceCollection**

This method returns an Enumerator object which supports the iteration by language specific elements. It is implied in the For...Each construct of Visual Basic and C#. This object allows to iterate over the link appearance objects included.

	Data Type	Explanation
<b>Return value</b>	VcObject	Reference object

## LinkAppearanceByIndex

**Method of VcLinkAppearanceCollection**

This method lets you access a link appearance object by its index. If a linkAppearance object does not exist at the index specified, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
<b>Parameter:</b> ⇒ index	System.Int16	Index of the link appearance object
<b>Return value</b>	VcLinkAppearance	LinkAppearance object returned

## LinkAppearanceByName

Method of VcLinkAppearanceCollection

This method retrieves a link appearance object by its name.

	Data Type	Explanation
<b>Parameter:</b>		
⇒ linkAppearanceName	System.String	Name of the link appearance object
<b>Return value</b>	VcLinkAppearance	LinkAppearance object

### Example Code VB.NET

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance
linkAppearanceCltn = VcGanttASP1.LinkAppearanceCollection
linkAppearance = linkAppearanceCltn.LinkAppearanceByName("Standard")
```

### Example Code C#

```
VcLinkAppearanceCollection linkAppearanceCltn =
vcGanttASP1.LinkAppearanceCollection;
VcLinkAppearance 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
<b>Return value</b>	VcLinkAppearance	Succeeding linkAppearance object

### Example Code VB.NET

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance

linkAppearanceCltn = VcGanttASP1.LinkAppearanceCollection
linkAppearance = linkAppearanceCltn.FirstLinkAppearance
While Not linkAppearance Is Nothing
    linkAppearance.Visible = False
    ListBox1.Items.Add("Name: " + linkAppearance.Name)
    linkAppearance = linkAppearanceCltn.NextLinkAppearance
End While
```

**Example Code C#**

```
VcLinkAppearanceCollection linkAppearanceCltn =
vcGanttASP1.LinkAppearanceCollection;
VcLinkAppearance linkAppearance = linkAppearanceCltn.FirstLinkAppearance();
while (linkAppearance != null)
{
    linkAppearance.Visible = false;
    listBox1.Items.Add("Name: " + linkAppearance.Name);
    linkAppearance = linkAppearanceCltn.NextLinkAppearance();
}
```

**Remove****Method of VcLinkAppearanceCollection**

This method lets you delete a link appearance. If the link appearance is being used in a different object, it cannot be deleted. Then **False** will be returned, otherwise **True**.

	Data Type	Explanation
<b>Parameter:</b>		
⇒ name	System.String	Name of the link appearance
<b>Return value</b>	System.Boolean	Link appearance deleted (True)/not deleted (False)

**Update****Method of VcLinkAppearanceCollection**

This method lets you update a link appearance collection after having modified it.

	Data Type	Explanation
<b>Return value</b>	System.Boolean	Link appearance collection was/was not successfully updated

**Example Code VB.NET**

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance

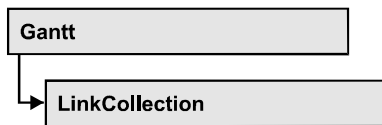
linkAppearanceCltn = VcGanttASP1.LinkAppearanceCollection
linkAppearance = linkAppearanceCltn.LinkAppearanceByIndex(0)
linkAppearanceCltn.Remove(linkAppearance.Name)
linkAppearanceCltn.Update()
```

## 788 API Reference: VcLinkAppearanceCollection

### Example Code C#

```
VcLinkAppearanceCollection linkAppearanceCltn =  
vcGanttASP1.LinkAppearanceCollection;  
VcLinkAppearance linkAppearance = linkAppearanceCltn.LinkAppearanceByIndex(0);  
linkAppearanceCltn.Remove(linkAppearance.Name);  
linkAppearanceCltn.Update();
```

## 6.49 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

- Count

### Methods

- FirstLink
- GetEnumerator
- NextLink
- SelectLinks

---

## Properties

### Count

**Read Only Property of VcLinkCollection**

This property lets you retrieve the number of links in the link collection.

	Data Type	Explanation
Property value	System.Int32	Number of links

#### Example Code VB.NET

```

Dim linkCltn As VcLinkCollection
Dim numberOfLinks As Integer

linkCltn = VcGanttASP1.LinkCollection
numberOfLinks = linkCltn.Count
  
```

#### Example Code C#

```

VcLinkCollection linkCltn = vcGanttASP1.LinkCollection;
int numberOfLinks = 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 VB.NET

```
Dim linkCltn As VcLinkCollection
Dim link As VcLink

linkCltn = VcGanttASP1.LinkCollection
link = linkCltn.FirstLink
```

#### Example Code C#

```
VcLinkCollection linkCltn = vcGanttASP1.LinkCollection;
VcLink link = linkCltn.FirstLink();
```

### GetEnumerator

Method of VcLinkCollection

This method returns an Enumerator object which supports the iteration by language specific elements. It is implied in the For...Each construct of Visual Basic and C#. This object allows to iterate over the link objects included.

	Data Type	Explanation
Return value	VcObject	Reference object

### 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
<b>Return value</b>	VcLink	Succeeding link

**Example Code VB.NET**

```
Dim linkCltn As VcLinkCollection
Dim link As VcLink

linkCltn = VcGanttASP1.LinkCollection
link = linkCltn.FirstLink
While Not link Is Nothing
    ListBox1.Items.Add(link.AllData)
    link = linkCltn.NextLink
End While
```

**Example Code C#**

```
VcLinkCollection linkCltn = vcGanttASP1.LinkCollection;
VcLink link = linkCltn.FirstLink();

while (link != null)
{
    listBox1.Items.Add(link.AllData);
    link = linkCltn.NextLink();
}
```

## SelectLinks

Method of VcLinkCollection

This method lets you specify the links that the link collection is to contain.

	Data Type	Explanation
<b>Parameter:</b> ⇒ selectionType	VcSelectionType  <b>Possible Values:</b> .vcAll 0 .vcAllLinksCausingCycles 7  .vcAllLinksInCycles 6	Links to be selected  All objects in the diagram will be selected 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. 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.
<b>Return value</b>	System.Int32	Number of links selected

**Example Code VB.NET**

```
Dim linkCltn As VcLinkCollection
linkCltn = VcGanttASP1.LinkCollection
linkCltn.SelectGroups (vcAllMarked)
```

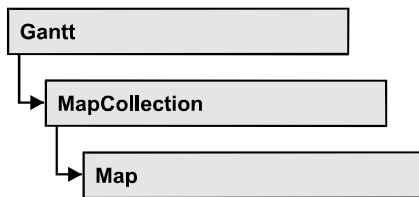


## 792 API Reference: VcLinkCollection

### Example Code C#

```
VcLinkCollection linkCltn = vcGanttASP1.LinkCollection;  
linkCltn.SelectGroups (vcAllMarked);
```

## 6.50 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 **ForEach mapEntry In Map** you can retrieve all data field entries in an iterative loop.

### Properties

- ConsiderFilterEntries
- Count
- GetEnumerator
- Name
- Specification
- Type

### Methods

- CreateEntry
- DeleteEntry
- FirstMapEntry
- NextMapEntry

---

## Properties

### 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	System.Int32	Number of map entries

#### Example Code VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim numberOfEntries As Integer

mapCltn = VcGanttASp1.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)
map = mapCltn.MapByName("Map1")
numberOfEntries = map.Count
```

#### Example Code C#

```
VcMapCollection mapCltn = vcGanttASp1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcAnyMap);
VcMap map = mapCltn.MapByName("Map1");
int numberOfEntries = map.Count;
```

### GetEnumerator

Read Only Property of VcMap

This method returns an Enumerator object which supports the iteration by language specific elements. It is implied in the For...Each construct of Visual Basic and C#. This object allows to iterate over the map entries included.

	Data Type	Explanation
Property value	VcObject	Reference object

### Name

Read Only Property of VcMap

This property lets you retrieve the name of a map.

	Data Type	Explanation
Property value	System.String	Name

### Example Code VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapName As String

mapCltn = VcGanttASPl.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)
map = mapCltn.FirstMap
mapName = map.Name
```

### Example Code C#

```
VcMapCollection mapCltn = vcGanttASPl.MapCollection;
mapCltn.SelectMaps(VcMapType.vcAnyMap);
VcMap map = mapCltn.FirstMap();
string 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	System.String	Specification of the map

## Type

### Property of VcMap

This property lets you enquire/set the map type.

	Data Type	Explanation
Property value	VcMapType  <b>Possible Values:</b> .vcAnyMap 0 .vcColorMap 1 .vcFontMap 8 .vcGraphicsFileMap 7 .vcMillimeterMap 9 .vcNumberMap 10	Map type  <b>any</b> (used only for selecting) <b>Colors</b> <b>Fonts</b> <b>Graphics file</b> <b>Millimetres</b> <b>Numbers</b>

**Example Code VB.NET**

```
Dim mapCltn As VcMapCollection
Dim map As VcMap

mapCltn = VcGanttASP1.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)
map = mapCltn.MapByName("Map1")
map.Type = VcMapType.vcPatternMap
```

**Example Code C#**

```
VcMapCollection mapCltn = vcGanttASP1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcAnyMap);
VcMap map = mapCltn.MapByName("Map1");
map.Type = VcMapType.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 VB.NET**

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry

mapCltn = VcGanttASP1.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)
map = mapCltn.MapByName("Map1")
mapEntry = map.CreateEntry
mapCltn.Update
```

**Example Code C#**

```
VcMapCollection mapCltn = vcGanttASP1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcAnyMap);
VcMap map = mapCltn.MapByName("Map1");
VcMapEntry mapEntry = map.CreateEntry();
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
<b>Parameter:</b> ⇒ mapEntry	VcMapEntry	Map entry
<b>Return value</b>	System.Boolean	Map entry was/was not deleted successfully

### Example Code VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry

mapCltn = VcGanttASPl.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)
map = mapCltn.MapByName("Map1")
mapEntry = map.FirstMapEntry
map.DeleteEntry(mapEntry)
mapCltn.Update
```

### Example Code C#

```
VcMapCollection mapCltn = vcGanttASPl.MapCollection;
mapCltn.SelectMaps(VcMapType.vcAnyMap);
VcMap map = mapCltn.MapByName("Map1");
VcMapEntry 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
<b>Return value</b>	VcMapEntry	First map entry

**Example Code VB.NET**

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry

mapCltn = VcGanttASP1.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)

map = mapCltn.FirstMap
mapEntry = map.FirstMapEntry
```

**Example Code C#**

```
VcMapCollection mapCltn = vcGanttASP1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcAnyMap);
VcMap map = mapCltn.FirstMap();
VcMapEntry mapEntry = map.FirstMapEntry();
```

**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	Succeeding map entry

**Example Code VB.NET**

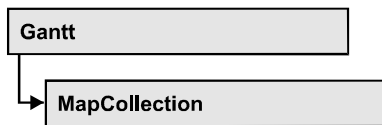
```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry

mapCltn = VcGanttASP1.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)
map = mapCltn.FirstMap
mapEntry = map.FirstMapEntry
While Not mapEntry Is Nothing
    ListBox1.Items.Add(mapEntry.LegendText)
    mapEntry = map.NextMapEntry
End While
```

**Example Code C#**

```
VcMapCollection mapCltn = vcGanttASP1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcAnyMap);
VcMap map = mapCltn.FirstMap();
VcMapEntry mapEntry = map.FirstMapEntry();
while (mapEntry != null)
{
    listBox1.Items.Add(mapEntry.LegendText);
    mapEntry= map.NextMapEntry();
}
```

## 6.51 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

- Count

### Methods

- Add
- AddBySpecification
- Copy
- FirstMap
- GetEnumerator
- MapByIndex
- MapByName
- NextMap
- Remove
- SelectMaps
- Update

---

## Properties

### Count

**Read Only Property of VcMapCollection**

This property lets you retrieve the number of maps in the `MapCollection` object.



## 800 API Reference: VcMapCollection

	Data Type	Explanation
Property value	System.Int32	Number of maps

### Example Code VB.NET

```
Dim mapCltn As VcMapCollection
Dim numberOfMaps As Integer

mapCltn = VcGanttASP1.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)
numberOfMaps = mapCltn.Count
```

### Example Code C#

```
VcMapCollection mapCltn = vcGanttASP1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcAnyMap);
int 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 has not been used before, the new map object will be returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
<b>Parameter:</b> ⇒ mapName	System.String	Map name
<b>Return value</b>	VcMap	New map object

### Example Code VB.NET

```
newMap = VcGanttASP1.MapCollection.Add("Map1")
```

### Example Code C#

```
VcMap newMap = vcGanttASP1.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
<b>Parameter:</b> ⇒ specification	System.String	Map specification
<b>Return value</b>	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
<b>Parameter:</b> ⇒ mapName	System.String	Name of the map to be copied
⇒ newMapName	System.String	Name of the new map
<b>Return value</b>	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). Beforehand, you have to specify a set of maps by the method **SelectMaps**.

	Data Type	Explanation
<b>Return value</b>	VcMap	First map

### Example Code VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap

mapCltn = VcGanttASp1.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)
map = mapCltn.FirstMap
```

### Example Code C#

```
VcMapCollection mapCltn = vcGanttASp1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcAnyMap);
VcMap map = mapCltn.FirstMap();
```

## GetEnumerator

Method of VcMapCollection

This method returns an Enumerator object which supports the iteration by language specific elements. It is implied in the For...Each construct of Visual Basic and C#. This object allows to iterate over the map objects included.

	Data Type	Explanation
Return value	VcObject	Reference object

## MapByIndex

Method of VcMapCollection

This method lets you access a map by its index. If a map does not exist at the index specified, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter: ⇒ index	System.Int16	Index of the map
Return value	VcMap	Map object returned

## MapByName

Method of VcMapCollection

By this method you can get a map by its name. Beforehand, you have to specify a set of maps 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
<b>Parameter:</b> ⇒ mapName	System.String	Name of the map
<b>Return value</b>	VcMap	Map

**Example Code VB.NET**

```
Dim mapCltn As VcMapCollection
Dim map As VcMap

mapCltn = VcGanttASP1.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)
map = mapCltn.MapByName("Map1")
```

**Example Code C#**

```
VcMapCollection mapCltn = vcGanttASP1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcAnyMap);
VcMap map = mapCltn.MapByName("Map1");
```

## 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
<b>Return value</b>	VcMap	Succeeding map

**Example Code VB.NET**

```
Dim mapCltn As VcMapCollection
Dim map As VcMap

mapCltn = VcGanttASP1.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)
map = mapCltn.FirstMap
While Not map Is Nothing
    ListBox1.Items.Add(map.Name)
    map = mapCltn.NextMap
End While
```

**Example Code C#**

```
VcMapCollection mapCltn = vcGanttASP1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcAnyMap);
VcMap map = mapCltn.FirstMap();
while (map != null)
{
    listBox1.Items.Add(map.Name);
    map = mapCltn.NextMap();
}
```

## 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
<b>Parameter:</b> ⇒ mapName	System.String	Map name
<b>Return value</b>	System.Boolean	Map deleted (True)/not deleted (False)

## SelectMaps

### Method of VcMapCollection

This method lets you specify which map types your map collection should contain.

	Data Type	Explanation
<b>Parameter:</b> ⇒ selectionType	VcMapType  <b>Possible Values:</b> .vcAnyMap 0 .vcColorMap 1 .vcFontMap 8 .vcGraphicsFileMap 7 .vcMillimeterMap 9 .vcNumberMap 10 .vcPatternMap 3	Map type to be selected  <b>any</b> (used only for selecting) <b>Colors</b> <b>Fonts</b> <b>Graphics file</b> <b>Millimetres</b> <b>Numbers</b> <b>Patterns</b>
<b>Return value</b>	System.Int32	Number of maps selected

### Example Code VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap

mapCltn = VcGanttASP1.MapCollection
mapCltn.SelectMaps (VcMapType.vcAnyMap)
```

### Example Code C#

```
VcMapCollection mapCltn = vcGanttASP1.MapCollection;
mapCltn.SelectMaps (VcMapType.vcAnyMap);
```

## Update

### Method of VcMapCollection

This method has to be used when map modifications have been made and you want to 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
<b>Return value</b>	System.Boolean	Update successful (True)/ not successful (False)

### Example Code VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry

mapCltn = VcGanttASP1.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)
map = mapCltn.MapByName("Map1")
mapEntry = map.FirstMapEntry
While Not mapEntry.DataFieldValue = "A"
    mapEntry = map.NextMapEntry
End While

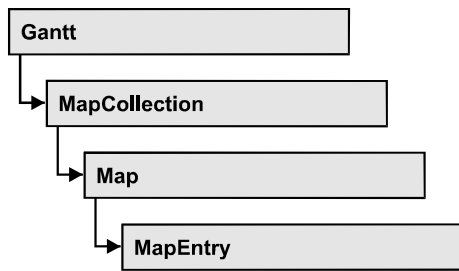
mapEntry.Color = Color.Blue
mapCltn.Update()
```

### Example Code C#

```
VcMapCollection mapCltn = vcGanttASP1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcAnyMap);
VcMap map = mapCltn.MapByName("Map1");
VcMapEntry mapEntry = map.FirstMapEntry();
while (mapEntry.DataFieldValue != "A")
    mapEntry = map.NextMapEntry();

mapEntry.Color = Color.LightSteelBlue;
mapCltn.Update();
```

## 6.52 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

- Color
- DataFieldValue
- FontBody
- FontName
- FontSize
- GraphicsFileName
- LegendText
- Millimeter
- Number
- Pattern

---

## Properties

### Color

#### 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.

	Data Type	Explanation
Property value	System.Drawing.Color	RGB color values {0...255},{0...255},{0...255}

**Example Code VB.NET**

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry
Dim colorOfMapEntry As Color

mapCltn = VcGanttASp1.MapCollection
mapCltn.SelectMaps(VcMapType.vcColorMap)
map = mapCltn.MapByName("Map1")
mapEntry = map.FirstMapEntry
colorOfMapEntry = mapEntry.Color
```

**Example Code C#**

```
VcMapCollection mapCltn = vcGanttASp1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcColorMap);
VcMap map = mapCltn.MapByName("Map1");
VcMapEntry mapEntry = map.FirstMapEntry();
Color 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	System.String	Content of the data field

**Example Code VB.NET**

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry
Dim dataFieldValue As String

mapCltn = VcGanttASp1.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)
map = mapCltn.MapByName("Map1")
mapEntry = map.FirstMapEntry

dataFieldValue = mapEntry.DataFieldValue
```

**Example Code C#**

```
VcMapCollection mapCltn = vcGanttASp1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcAnyMap);
VcMap map = mapCltn.MapByName("Map1");
VcMapEntry mapEntry = map.FirstMapEntry();
string 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	VcFontBody  <b>Possible Values:</b> .vcBold 2 .vcBoldItalic 4 .vcItalic 3 .vcRegular 1	Font body  bold bold and italic italic regular

### Example Code VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry
Dim fontBodyOfMapEntry As VcFontBody

mapCltn = VcGanttASp1.MapCollection
mapCltn.SelectMaps(VcMapType.vcFontMap)
map = mapCltn.MapByName("Map1")
mapEntry = map.FirstMapEntry
fontBodyOfMapEntry = VcFontBody.vcBold
```

### Example Code C#

```
VcMapCollection mapCltn = vcGanttASp1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcFontMap);
VcMap map = mapCltn.MapByName("Map1");
VcMapEntry mapEntry = map.FirstMapEntry();
VcFontBody fontBodyOfMapEntry = VcFontBody.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	System.String	Font type

**Example Code VB.NET**

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry
Dim fontNameOfMapEntry As String

mapCltn = VcGanttASP1.MapCollection
mapCltn.SelectMaps(VcMapType.vcFontMap)
map = mapCltn.MapByName("Map1")
mapEntry = map.FirstMapEntry
fontNameOfMapEntry = "Arial"
```

**Example Code C#**

```
VcMapCollection mapCltn = vcGanttASP1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcFontMap);
VcMap map = mapCltn.MapByName("Map1");
VcMapEntry mapEntry = map.FirstMapEntry();
string fontNameOfMapEntry = "Arial";
```

## FontSize

**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	System.Int32	Font size

**Example Code VB.NET**

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry
Dim fontSizeOfMapEntry As Integer

mapCltn = VcGanttASP1.MapCollection
mapCltn.SelectMaps(VcMapType.vcFontMap)
map = mapCltn.MapByName("Map1")
mapEntry = map.FirstMapEntry
fontSizeOfMapEntry = 14
```

**Example Code C#**

```
VcMapCollection mapCltn = vcGanttASP1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcFontMap);
VcMap map = mapCltn.MapByName("Map1");
VcMapEntry mapEntry = map.FirstMapEntry();
int fontSizeOfMapEntry = 14;
```

## GraphicsFileName

**Property of VcMapEntry**

*for Graphic File Maps:* This property lets you set or retrieve the graphics file name of a map entry. *Available formats:*

## 810 API Reference: VcMapEntry

- \*.BMP (Microsoft Windows Bitmap)
- \*.EMF (Enhanced Metafile)
- \*.EMF, with EMF+ included
- \*.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, 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	System.String	Name of the graphics file

### Example Code VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry
Dim exeName As String
Dim exeDir As String

mapCltn = VcGanttASP1.MapCollection
mapCltn.SelectMaps(VcMapType.vcGraphicsFileMap)
map = mapCltn.MapByName("Map1")
mapEntry = map.FirstMapEntry

exeName = System.Environment.GetCommandLineArgs(0)
exeDir = System.IO.Path.GetDirectoryName(exeName)
mapEntry.GraphicsFileName = exeDir + "\Bitmaps\picture1.bmp"
```

**Example Code C#**

```
VcMapCollection mapCltn = vcGanttASP1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcGraphicsFileMap);
VcMap map = mapCltn.MapByName("Map1");
VcMapEntry mapEntry = map.FirstMapEntry();

String exeName = Environment.GetCommandLineArgs()[0];
mapEntry.GraphicsFileName = System.IO.Path.GetDirectoryName(exeName) +
@"\..\Bitmaps\picture1.bmp";
```

**LegendText****Property of VcMapEntry**

This property lets you set or retrieve the legend text of a map entry.

	Data Type	Explanation
Property value	System.String	Legend text

**Example Code VB.NET**

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry
Dim legendOfMapEntry As String

mapCltn = VcGantt1.MapCollection
mapCltn.SelectMaps(VcMapType.vcFontMap)
map = mapCltn.MapByName("Map1")
mapEntry = map.FirstMapEntry
legendOfMapEntry = "1. activity"
```

**Example Code C#**

```
VcMapCollection mapCltn = vcGantt1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcFontMap);
VcMap map = mapCltn.MapByName("Map1");
VcMapEntry mapEntry = map.FirstMapEntry();
string 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	System.Int32	1/100 units

## 812 API Reference: VcMapEntry

### Example Code VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry
Dim millimeterOfMapEntry As Integer

mapCltn = VcGantt1.MapCollection
mapCltn.SelectMaps(VcMapType.vcMillimeterMap)
map = mapCltn.MapByName("Map1")
mapEntry = map.FirstMapEntry
millimeterOfMapEntry = 3
```

### Example Code C#

```
VcMapCollection mapCltn = vcGantt1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcMillimeterMap);
VcMap map = mapCltn.MapByName("Map1");
VcMapEntry mapEntry = map.FirstMapEntry();
int millimeterOfMapEntry = 3;
```

## Number

### Property of VcMapEntry



*For numeric maps:* This property lets you set or retrieve the numeric value of a map entry.

	Data Type	Explanation
Property value	System.Int32	Numeric value

## Pattern

### Property of VcMapEntry




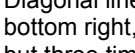

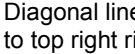
*For Pattern Maps (vcPatternMap):* this property lets you set or retrieve the pattern of a map entry.

	Data Type	Explanation
Property value	VcFillPattern  <b>Possible Values:</b> .vc05PercentPattern... .vc90PercentPattern 01 - 11  .vcAeroGlassPattern 44	Pattern type  Dots in foreground color on background color, the density of the foreground color 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 twice the line width
.vcDashedHorizontalPattern 2026	Dashed horizontal lines
.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
.vcHorizontalPattern 3	Horizontal lines
.vcLargeCheckerboardPattern 2044	Checkerboard pattern showing squares of twice the size of vcSmallCheckerBoardPattern
.vcLargeConfettiPattern 2029	Confetti pattern, large

## 814 API Reference: VcMapEntry

.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
.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
.vcVerticalPattern 2	Vertical lines

.vcVerticalTopLightedConvexPattern 42	Vertical color gradient from bright to dark 
.vcWavePattern 2031	Horizontal waves pattern 
.vcWeavePattern 2034	Interwoven stripes 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, showing the same spacing but three times the line width of vcB-DiagonalPattern 
.vcZigZagPattern 2030	Horizontal zig-zag lines 

**Example Code VB.NET**

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry
Dim pattern As VcFillPattern

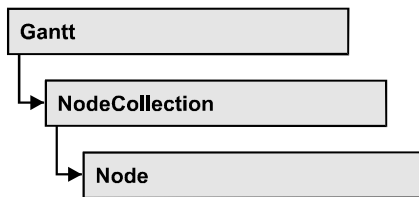
mapCltn = VcGantt1.MapCollection
mapCltn.SelectMaps(VcMapType.vcPatternMap)
map = mapCltn.MapByName("Map1")
mapEntry = map.FirstMapEntry
pattern = VcFillPattern.vcB-DiagonalPattern
```

**Example Code C#**

```
VcMapCollection mapCltn = vcGantt1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcPatternMap);
VcMap map = mapCltn.MapByName("Map1");
VcMapEntry mapEntry = map.FirstMapEntry();
VcFillPattern pattern = VcFillPattern.vcB-DiagonalPattern;
```



## 6.53 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
- Marked
- OutgoingLinks
- OutgoingLinks
- SuperGroup

### Methods

- DataRecord
- Delete
- GetPositionInView
- NodeRowInView
- OutlineIndent
- OutlineOutdent
- RelatedDataRecord
- SetPositionInView
- Update

## 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 an object 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	System.String	All data of the data set

#### Example Code VB.NET

```
Private Sub VcGanttASP1_VcNodeModifying(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcNodeModifyingEventArgs) Handles VcGanttASP1.VcNodeModifying
    Dim allDataOfNode As String
    e.ReturnStatus = VcReturnStatus.vcRetStatFalse

    allDataOfNode = e.Node.AllData
    MsgBox(allDataOfNode)
End Sub
```

#### Example Code C#

```
private void vcGanttASP1_VcNodeModifying(object sender,
NETRONIC.XGantt.VcNodeModifyingEventArgs e)
{
    e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
    string allDataOfNode = e.Node.AllData.ToString();
    MessageBox.Show(allDataOfNode);
}
```

### 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 **Update** method.

	Data Type	Explanation
Parameter: ⇒ index	System.Int16	Index of data field
Property value	System.Object	Content of the data field

## 818 API Reference: VcNode

### Example Code VB.NET

```
node.DataField(1) = "Node 1"
```

### Example Code C#

```
node.set_DataField(1, "Node 1");
```

## ID

### Read Only Property of VcNode

By this property you can retrieve the ID of a node.

	Data Type	Explanation
Property value	System.String	Node ID

### Example Code VB.NET

```
VcNode node = VcGanttASP1.NodeCollection.FirstNode()  
MsgBox (node.ID)
```

### Example Code C#

```
VcNode node = vcGanttASP1.NodeCollection.FirstNode();  
MessageBox.Show(node.ID)
```

## IncomingLinks

### Read Only Property of VcNode

This property lets you access all incoming links of a node.

	Data Type	Explanation
Property value	VcLinkCollection	Link collection

### Example Code VB.NET

```
Dim incomingLinks As VcLinkCollection  
Dim link As VcLink  
  
incomingLinks = node.IncomingLinks  
For Each link In incomingLinks  
    link.PredecessorNode.Marked = True  
Next
```

### Example Code C#

```
VcLinkCollection incomingLinks = node.IncomingLinks;  
  
foreach (VcLink link in incomingLinks)  
    link.PredecessorNode.Marked = true;
```

## Marked

Property of VcNode

This property lets you 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	System.Boolean	Node marked/not marked

### Example Code VB.NET

```
Dim nodeCltn As VcNodeCollection
Dim node As VcNode
Dim predecessor As VcNode
Dim linkCltn As VcLinkCollection
Dim link As VcLink

nodeCltn = VcGantt1.NodeCollection
nodeCltn.SelectNodes(VcSelectionType.vcAll)

For Each node In nodeCltn
    linkCltn = node.IncomingLinks
    For Each link In linkCltn
        predecessor = link.PredecessorNode
        predecessor.Marked = True
    Next
Next
```

### Example Code C#

```
VcNodeCollection nodeCltn = vcGantt1.NodeCollection;
nodeCltn.SelectNodes(VcSelectionType.vcAll);
VcNode predecessorNode;
VcLinkCollection linkCltn;
foreach (VcNode node in nodeCltn)
{
    linkCltn = node.IncomingLinks;
    foreach (VcLink link in linkCltn)
    {
        predecessorNode = link.PredecessorNode;
        predecessorNode.Marked = true;
    }
}
```

## OutgoingLinks

Read Only Property of VcNode

This property lets you access all links that leave a node.

	Data Type	Explanation
Property value	VcLinkCollection	Link collection

## 820 API Reference: VcNode

### Example Code VB.NET

```
Dim outgoingLinks As VcLinkCollection
Dim link As VcLink

outgoingLinks = node.OutgoingLinks
For Each link In outgoingLinks
    successorNode = link.SuccessorNodeMarked = True
Next
```

### Example Code C#

```
VcLinkCollection outgoingLinks = node.OutgoingLinks;
foreach (VcLink link in outgoingLinks)
    link.SuccessorNode.Marked = true;
```

## OutgoingLinks

Read Only Property of VcNode

This property lets you access all links that leave a node.

	Data Type	Explanation
Property value	VcLinkCollection	Link collection

## 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 VB.NET

```
Dim text As String

text = node.SuperGroup.Name
```

### Example Code C#

```
string text = node.SuperGroup.Name;
```

## 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

### Delete

Method of VcNode

This method lets you delete a node.

	Data Type	Explanation
Return value	System.Boolean	Node was/was not deleted successfully

#### Example Code VB.NET

```
node.Delete()
```

#### Example Code C#

```
node.Delete();
```

### GetPositionInView

Method of VcNode

This method lets you retrieve the position of a node in the visible area of the diagram.

	Data Type	Explanation
Parameter: viewReferencePoint	VcViewReferencePoint  <b>Possible Values:</b> .vcVRPBottomCenter 28 .vcVRPBottomLeft 27 .vcVRPBottomRight 29 .vcVRPCenterCenter 25 .vcVRPCenterLeft 24	Reference point (of the diagram)  bottom center bottom left bottom right center center center left

## 822 API Reference: VcNode

	.vcVRPCenterRight 26 .vcVRPTopCenter 22 .vcVRPTopLeft 21 .vcVRPTopRight 23	center right top center top left top right
nodeReferencePoint	VcNodeReferencePoint  <b>Possible Values:</b> .vcNRPBottomCenter 28 .vcNRPBottomLeft 27 .vcNRPBottomRight 29 .vcNRPCenterCenter 25 .vcNRPCenterLeft 24 .vcNRPCenterRight 26 .vcNRPTopCenter 22 .vcNRPTopLeft 21 .vcNRPTopRight 23	Node reference point  bottom center bottom left bottom right center center center left center right top center top left top right
↔ xOffset	System.Int32	X value of the offset (= distance of the node reference point and the reference point) (unit: pixels)
↔ yOffset	System.Int32	Y value of the offset (unit: pixels)
<b>Return value</b>	Void	

## 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
<b>Return value</b>	System.Boolean	Row is/is not in the visible section of the diagram

### Example Code VB.NET

```
Dim node As VcNode

node = VcGantt1.GetNodeByID(15)
If Not node.NodeRowInView Then
    VcGantt1.ScrollToNodeLine(node, VcVerticalAlignment.vcVerCenterAligned)
End If
```

### Example Code C#

```
VcNode node = vcGantt1.GetNodeByID(2);
if (node.NodeRowInView() == false)
    vcGantt1.ScrollToNodeLine(node, VcVerticalAlignment.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
<b>Return value</b>	System.Boolean	Method successful (True)/ not successful (False)

### Example Code VB.NET

```
Private Sub CommandOutlineIndent_Click()
    Dim anode As VcNode
    Set anode = getSelectedNode 'Function returns selected node
    MsgBox anode.OutlineIndent
    Set anode = Nothing
End Sub
```

### Example Code C#

```
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
<b>Return value</b>	System.Boolean	Method successful (True)/ not successful (False)



### Example Code VB.NET

```
Private Sub CommandOutlineOutdent_Click()
    Dim anode As VcNode
    Set anode = getSelectedNode 'Function returns selected node
    MsgBox anode.OutlineOutdent
    Set anode = Nothing
End Sub
```

### Example Code C#

```
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
<b>Parameter:</b> ⇒ index	System.Int16	Index of data field that holds the key
<b>Return value</b>	VcDataRecord	Related data record returned

## SetPositionInView

Method of VcNode

This method sets that the node will be displayed in a visible position of the diagram after scrolling. The position is specified by an offset vector (x,y) between a reference point in the node and a reference point in the diagram.

	Data Type	Explanation
<b>Parameter:</b> viewReferencePoint	VcViewReferencePoint  <b>Possible Values:</b> .vcVRPBottomCenter 28 .vcVRPBottomLeft 27 .vcVRPBottomRight 29 .vcVRPCenterCenter 25 .vcVRPCenterLeft 24 .vcVRPCenterRight 26 .vcVRPTopCenter 22 .vcVRPTopLeft 21	Reference point (of the diagram)  bottom center bottom left bottom right center center center left center right top center top left

	.vcVRPTopRight 23	top right
nodeReferencePoint	VcNodeReferencePoint <b>Possible Values:</b> .vcNRPBottomCenter 28 .vcNRPBottomLeft 27 .vcNRPBottomRight 29 .vcNRPCenterCenter 25 .vcNRPCenterLeft 24 .vcNRPCenterRight 26 .vcNRPTopCenter 22 .vcNRPTopLeft 21 .vcNRPTopRight 23	Node reference point bottom center bottom left bottom right center center center left center right top center top left top right
⇐ xOffset	System.Int32	X value of the offset (= distance of the node reference point and the reference point) (unit: pixels)
⇐ yOffset	System.Int32	Y value of the offset (unit: pixels)
<b>Return value</b>	Void	

**Example Code VB.NET**

```
' 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)
Dim node As VcNode
```

```
node.SetPositionInView(VcViewReferencePoint.vcVRPBottomRight,
VcNodeReferencePoint.vcNRPBottomRight, -10, -10)
```

**Example Code C#**

```
// 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)
VcNode node;
node.SetPositionInView(VcViewReferencePoint.vcVRPBottomRight,
VcNodeReferencePoint.vcNRPBottomRight, -10, -10);
```

## Update

**Method of VcNode**

If data fields of a node have been modified by the **DataField** property, the diagram needs to be updated by the **Update** method.

	Data Type	Explanation
<b>Return value</b>	System.Boolean	Node was/was not updated successfully

**Example Code VB.NET**

```
Dim nodeCltn As VcNodeCollection
Dim node As VcNode

nodeCltn = VcGanttASP1.NodeCollection
node = nodeCltn.FirstNode

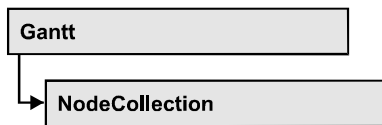
node.DataField(12) = "Group A"
node.Update()
```

## 826 API Reference: VcNode

### Example Code C#

```
VcNodeCollection nodeCltn = vcGanttASPl.NodeCollection;  
VcNode node = nodeCltn.FirstNode();  
node.set_DataField(12, "Group A");  
node.Update();
```

## 6.54 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 NodeCollection** or by the methods **First...** and **Next...**. The number of nodes in the collection object can be retrieved by the property **Count**.

### Properties

- Count

### Methods

- FirstNode
- GetEnumerator
- NextNode
- SelectNodes

---

## Properties

### Count

**Read Only Property of VcNodeCollection**

This property lets you retrieve the number of nodes in the NodeCollection object.

	Data Type	Explanation
Property value	System.Int32	Number of Nodes in the node collection

#### Example Code VB.NET

```

Dim nodeCltn As VcNodeCollection

nodeCltn = VcGanttASP1.NodeCollection
MessageBox("Number of nodes: " + nodeCltn.Count)
  
```

#### Example Code C#

```

VcNodeCollection nodeCltn = vcGanttASP1.NodeCollection;
MessageBox.Show("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 NodeCollection, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcNode	First Node

#### Example Code VB.NET

```
Dim nodeCltn As VcNodeCollection
Dim node As VcNode

nodeCltn = VcGanttASp1.NodeCollection
node = nodeCltn.FirstNode
```

#### Example Code C#

```
VcNodeCollection nodeCltn = vcGanttASp1.NodeCollection;
VcNode node = nodeCltn.FirstNode();
```

### GetEnumerator

Method of VcNodeCollection

This method returns an Enumerator object which supports the iteration by language specific elements. It is implied in the For...Each construct of Visual Basic and C#. This object allows to iterate over the node objects included.

	Data Type	Explanation
Return value	VcObject	Reference object

### 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
<b>Return value</b>	VcNode	Succeeding node

**Example Code VB.NET**

```
Dim nodeCltn As VcNodeCollection
Dim node As VcNode

nodeCltn = VcGanttASP1.NodeCollection
node = nodeCltn.FirstNode
While Not node Is Nothing
    node.Marked = False
    node = nodeCltn.NextNode
End While
```

**Example Code C#**

```
VcNodeCollection nodeCltn = vcGanttASP1.NodeCollection;
VcNode node = nodeCltn.FirstNode();
while (node != null)
{
    node.Marked = false;
    node = nodeCltn.NextNode;
}
```

**SelectNodes****Method of VcNodeCollection**

This method lets you specify the nodes to be collected by the NodeCollection object.

	Data Type	Explanation
<b>Parameter:</b> ⇒ selType	VcSelectionType  <b>Possible Values:</b> .vcAll 0 .vcAllLinksCausingCycles 7  .vcAllLinksInCycles 6	Nodes to be selected  All objects in the diagram will be selected 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. 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.
<b>Return value</b>	System.Int32	Number of nodes selected

**Example Code VB.NET**

```
Dim nodeCltn As VcNodeCollection
Dim node As VcNode

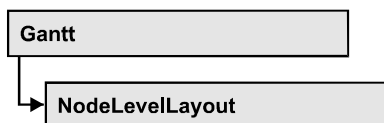
nodeCltn = VcGanttASP1.NodeCollection
nodeCltn.SelectNodes(VcSelectionType.vcSelected)
```

## 830 API Reference: VcNodeCollection

### Example Code C#

```
VcNodeCollection nodeCltn = vcGanttASPl.NodeCollection;  
nodeCltn.SelectNodes(VcSelectionType.vcSelected);
```

## 6.55 VcNodeLevelLayout



An object of the type `VcNodeLevelLayout` defines the sorting of nodes as well as the appearance of node rows.

### Properties

- `CalendarGridName`
- `CalendarGridsVisible`
- `RowBackgroundColorAsARGB`
- `RowBackgroundColorDataFieldIndex`
- `RowBackgroundColorMapName`
- `RowPattern`
- `RowPatternColorAsARGB`
- `RowPatternColorDataFieldIndex`
- `RowPatternColorMapName`
- `RowPatternDataFieldIndex`
- `RowPatternMapName`
- `SeparationLineColor`
- `SeparationLineInterval`
- `SeparationLinesVisible`
- `SeparationLinesVisibleAtTop`
- `SeparationLineThickness`
- `SeparationLineType`
- `SortDataFieldIndex`
- `SortOrder`

---

## Properties

### CalendarGridName

**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.



## 832 API Reference: VcNodeLevelLayout

	Data Type	Explanation
Property value	System.String	name of the calendar grid

### CalendarGridsVisible

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	System.Boolean	Workfree periods are/are not accentuated

### RowBackgroundColorAsARGB

Property of VcNodeLevelLayout

This property lets you set or retrieve the background color of the rows. The default color is white.

	Data Type	Explanation
--	-----------	-------------

### RowBackgroundColorDataFieldIndex

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
--	-----------	-------------

## RowBackgroundColorMapName

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

## 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	Pattern type

## 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	System.Drawing.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	System.Int32	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	System.String	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	System.Int32	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	System.String	Name of the pattern 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	System.Drawing.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

## SeparationLinesVisible

**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
Property value	System.Boolean	Separation lines are displayed/not displayed

## SeparationLinesVisibleAtTop

**Read Only Property of VcNodeLevelLayout**

This property lets you set or retrieve whether separation lines are to be displayed between activities.

This property also can be set in the **Nodes** section of the **Grouping** dialog.

	Data Type	Explanation
Property value	System.Boolean	Separation lines at top are displayed/not displayed

## SeparationLineThickness

**Read Only Property of VcNodeLevelLayout**

This property lets you set or retrieve the line thickness of a separation line between node 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 **Grouping** dialog, section **Groupwise**, field **Separation Line**.

	Data Type	Explanation
Property value	System.Int32	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 specify/enquire 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  <b>Possible Values:</b> .vcDashed 4 .vcDashedDotted 5 .vcDotted 3 .vcNone 1 .vcSolid 2	Type of separation lines of hierarchy levels  Line dashed Line dashed-dotted Line dotted No line type assigned Line solid

## SortDataFieldIndex

**Property of VcNodeLevelLayout**

This property lets you set/retrieve the data field index used for sorting the nodes of this VcGroupLevelLayout object

## 838 API Reference: VcNodeLevelLayout

	Data Type	Explanation
<b>Parameter:</b> ⇒ sortlevel	System.Int32	Sorting level
<b>Property value</b>	System.Int32	sorting field

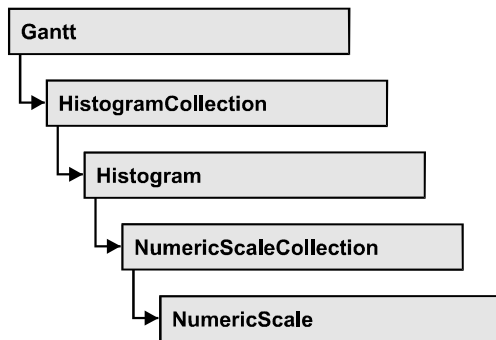
## SortOrder

### 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 activities are sorted by. This property also can be set in the **Grouping** dialog.

	Data Type	Explanation
<b>Parameter:</b> ⇒ sortLevel	System.Int32	Sorting level
<b>Property value</b>	SortOrderEnum	Ascending or descending order

## 6.56 VcNumericScale



An object of the type VcNumericScale is the scale of the vertical axis of a histogram.

### Properties

- BackgroundColor
- Font
- FontColor
- Histogram
- MajorTicks
- MajorTicksEx
- MinorTicks
- MinorTicksEx
- Name
- Pattern
- ThreeDEffect
- Title
- Unit
- UnitLabel
- UnitWidth

---

## Properties

### BackgroundColor

Property of VcNumericScale

This property lets you set or retrieve the background color of the numeric scale. You can also set this color in the **Edit Histogram** dialog.



## 840 API Reference: VcNumericScale

	Data Type	Explanation
Property value	System.Drawing.Color	RGB color values {0...255},{0...255},{0...255} <b>Default value:</b> RGB (192, 192, 192)

### Example Code VB.NET

```
Dim histogram As VcHistogram
Dim numericScale As VcNumericScale

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
numericScale = histogram.NumericScaleCollection.Active
numericScale.BackgroundColor = Color.Blue
```

### Example Code C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1");
VcNumericScale numericScale = histogram.NumericScaleCollection.Active;
numericScale.BackgroundColor = Color.LightSteelBlue;
```

## Font

### Property of VcNumericScale

This property lets you set or retrieve the font attributes of the numeric scale.

	Data Type	Explanation
Property value	Font	Font attributes of the numeric scale

### Example Code VB.NET

```
Dim histogram As VcHistogram
Dim numericScale As VcNumericScale
Dim newFont As Font

histogram = VcGantt1.HistogramCollection.FirstHistogram()
numericScale = histogram.NumericScaleCollection.FirstNumericScale()
newFont = New Font("Times New Roman", 14, FontStyle.Italic)
numericScale.Font = newFont
```

### Example Code C#

```
VcHistogram histogram = vcGantt1.HistogramCollection.FirstHistogram();
VcNumericScale numericScale =
histogram.NumericScaleCollection.FirstNumericScale()
Font newFont = new Font("Times New Roman", 14, FontStyle.Italic);
numericScale.Font = newFont;
```

## FontColor

### Property of VcNumericScale

This property lets you set or retrieve the font color of the numeric scale.

	Data Type	Explanation
<b>Property value</b>	System.Drawing.Color	RGB color values  {0...255},{0...255},{0...255} <b>Default value:</b> RGB (0,0,0)

**Example Code VB.NET**

```
Dim histogram As VcHistogram
Dim numericScale As VcNumericScale

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
numericScale = histogram.NumericScaleCollection.Active
numericScale.FontColor = Color.Blue
```

**Example Code C#**

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1");
VcNumericScale numericScale = histogram.NumericScaleCollection.Active;
numericScale.FontColor = Color.LightSteelBlue;
```

## Histogram

**Read Only Property of VcNumericScale**

This property lets you retrieve the histogram to which the numeric scale belongs.

	Data Type	Explanation
<b>Property value</b>	VcHistogram	Histogram object

## MajorTicks

**Property of VcNumericScale**

This property lets you set or retrieve after how many units a major tick is drawn that has an annotation. Also see **set/getMinorTick**. You can also set the number of the units in the **Edit Histogram** dialog.

	Data Type	Explanation
<b>Property value</b>	System.Int16	Number of units between two major ticks

### Example Code VB.NET

```
Dim histogram As VcHistogram
Dim numericScale As VcNumericScale

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
numericScale = histogram.NumericScaleCollection.Active
numericScale.MajorTicks = 4
```

### Example Code C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1");
VcNumericScale numericScale = histogram.NumericScaleCollection.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	System.Double	Number of units between two major ticks

### Example Code VB.NET

```
Dim histogram As VcHistogram
Dim numericScale As VcNumericScale

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
numericScale = histogram.NumericScaleCollection.Active
numericScale.MajorTicks = 4
```

### Example Code C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1");
VcNumericScale numericScale = histogram.NumericScaleCollection.Active;
numericScale.MajorTicks = 4;
```

## MinorTicks

Property of VcNumericScale

This property lets you set or retrieve after how many time units a minor tick without annotation is drawn. Also see **set/getMinorTick**. You can also set the number of the units in the **Edit Histogram** dialog.

	Data Type	Explanation
Property value	System.Int16	Number of units between two minor ticks

**Example Code VB.NET**

```
Dim histogram As VcHistogram
Dim numericScale As VcNumericScale

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
numericScale = histogram.NumericScaleCollection.Active
numericScale.MinorTicks = 2
```

**Example Code C#**

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1");
VcNumericScale numericScale = histogram.NumericScaleCollection.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	System.Double	Number of units between two minor ticks

**Example Code VB.NET**

```
Dim histogram As VcHistogram
Dim numericScale As VcNumericScale

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
numericScale = histogram.NumericScaleCollection.Active
numericScale.MinorTicks = 2
```

**Example Code C#**

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1");
VcNumericScale numericScale = histogram.NumericScaleCollection.Active;
numericScale.MinorTicks = 2;
```

## Name

**Read Only Property of VcNumericScale**

This property lets you retrieve the name of a numeric scale of an histogram. The name can be set in the **Edit Histogram** dialog.

## 844 API Reference: VcNumericScale

	Data Type	Explanation
Property value	System.String	Name of the numeric scale

### Example Code VB.NET

```
Dim histogram As VcHistogram
Dim numericScale As VcNumericScale

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
numericScale = histogram.NumericScaleCollection.Active
MsgBox("Active numeric Scale: " + numericScale.Name)
```






### Example Code C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1");
VcNumericScale numericScale = histogram.NumericScaleCollection.Active;
MessageBox.Show("Active numeric scale: " + numericScale.Name);
```

## Pattern

### Property of VcNumericScale

This property lets you set or retrieve the pattern of the numeric scale.

	Data Type	Explanation
Property value	VcFillPatternSingleColored	Pattern type
	<b>Possible Values:</b> .vcAeroGlassPattern 44  .vcSingleColoredNoPattern 1276 .vcSingleColoredVerticalBottomLightedConvexPattern 43  .vcSingleColoredVerticalConcavePattern 40  .vcSingleColoredVerticalConvexPattern 41  .vcSingleColoredVerticalTopLightedConvexPattern 42	Vertical color gradient in the color of the fill pattern   No fill pattern Vertical color gradient from bright to dark   Vertical color gradient from dark to bright to dark   Vertical color gradient from bright to dark to bright   Vertical color gradient from dark to bright 

## 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
<b>Property value</b>	System.Boolean	3D effect switched on (True)/switched off (False) <b>Default value:</b> False

### Example Code VB.NET

```
Dim histogram As VcHistogram
Dim numericScale As VcNumericScale

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
numericScale = histogram.NumericScaleCollection.Active
numericScale.ThreeDEffect = True
```

### Example Code C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1");
VcNumericScale numericScale = histogram.NumericScaleCollection.Active;
numericScale.ThreeDEffect = true;
```

## Title

### Property of VcNumericScale

This property lets you set or retrieve a title of 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
<b>Parameter:</b> ⇒ position	VcNumericAnnotationPosition	Position of the title in the numeric scale
<b>Property value</b>	System.String	Title of the numeric scale

### Example Code VB.NET

```
' Title positioned at 50% downward from top
numericScale.Title(VcNumericAnnotationPosition.vc50PercentFromTop) = "1350
Loops"
```

### Example Code C#

```
// Title positioned at 50% downward from top
numericScale.set_Title(VcNumericAnnotationPosition.vc50PercentFromTop, "1350
Loops");
```

## 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	System.Int32	unit

## UnitLabel

### Property of VcNumericScale

This property lets you set or retrieve the designation of the units of the numeric scale. This designation is displayed in the middle of the upper border of the numeric scale.

	Data Type	Explanation
Property value	System.String	Designation of the unit

### Example Code VB.NET

```
Dim numericScaleCltn As VcNumericScaleCollection
Dim activeNumericScale As VcNumericScale

numericScaleCltn =
VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1").NumericScaleCollecti
on
activeNumericScale = numericScaleCltn.Active
activeNumericScale.UnitLabel = "Hours"
```

### Example Code C#

```
VcNumericScaleCollection numericScaleCltn =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1").NumericScaleCollecti
on;
VcNumericScale 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	System.Int32	unit width (1/100 mm)

**Example Code VB.NET**

```
Dim numericScaleCltn As VcNumericScaleCollection
Dim activeNumericScale As VcNumericScale

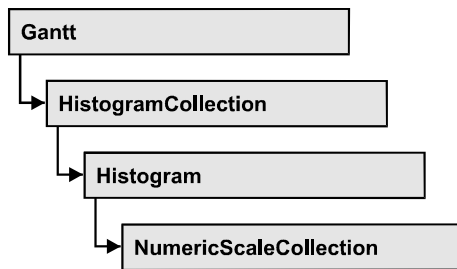
numericScaleCltn =
VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1").NumericScaleCollecti
on
activeNumericScale = numericScaleCltn.Active
activeNumericScale.UnitWidth = 200
```

**Example Code C#**

```
VcNumericScaleCollection numericScaleCltn =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1").NumericScaleCollecti
on;
VcNumericScale activeNumericScale = numericScaleCltn.Active;
activeNumericScale.UnitWidth = 200;
```



## 6.57 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 scale 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

- Active
- Count

### Methods

- FirstNumericScale
- NextNumericScale
- NumericScaleByIndex
- NumericScaleByName

---

## Properties

### Active

Property of `VcNumericScaleCollection`

This method lets you set or retrieve the active numeric scale of the histogram.

	Data Type	Explanation
Property value	VcNumericScale	Currently used numeric scale

**Example Code VB.NET**

```
Dim numericScaleCltn As VcNumericScaleCollection
Dim activeNumericScale As VcNumericScale

numericScaleCltn =
VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1").NumericScaleCollecti
on
activeNumericScale = numericScaleCltn.Active
```

**Example Code C#**

```
VcNumericScaleCollection numericScaleCltn =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1").NumericScaleCollecti
on;
VcNumericScale 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	System.Int32	Number of numeric scales

**Example Code VB.NET**

```
Dim numericScaleCltn As VcNumericScaleCollection
Dim numberOfNumericScales As Integer

numericScaleCltn =
VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1").NumericScaleCollecti
on
numberOfNumericScales = numericScaleCltn.Count
```

**Example Code C#**

```
VcNumericScaleCollection numericScaleCltn =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1").NumericScaleCollecti
on;
int numberOfNumericScale = 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
<b>Return value</b>	VcNumericScale	First numeric scale

### Example Code VB.NET

```
Dim numericScaleCltn As VcNumericScaleCollection
Dim numericScale As VcNumericScale

numericScaleCltn =
VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1").NumericScaleCollecti
on
numericScale = numericScaleCltn.FirstNumericScale
```

### Example Code C#

```
VcNumericScaleCollection numericScaleCltn =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1").NumericScaleCollecti
on;
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
<b>Return value</b>	VcNumericScale	Succeeding numeric scale

### Example Code VB.NET

```
Dim numericScaleCltn As VcNumericScaleCollection
Dim numericScale As VcNumericScale

numericScaleCltn =
VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1").NumericScaleCollecti
on
numericScale = numericScaleCltn.FirstNumericScale

While Not numericScale Is Nothing
    ListBox1.Items.Add(numericScale.Name)
    numericScale = numericScaleCltn.NextNumericScale
End While
```

**Example Code C#**

```
VcNumericScaleCollection numericScaleCltn =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1").NumericScaleCollecti
on;
numericScale = numericScaleCltn.FirstNumericScale();
while (numericScale != null)
{
    listBox1.Items.Add(numericScale.Name);
    numericScale = numericScaleCltn.NextNumericScale();
}
```

**NumericScaleByIndex****Method of VcNumericScaleCollection**

This method lets you access a numeric scale by its index. If a numeric scale does not exist at the index specified, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
<b>Parameter:</b> ⇒ index	System.Int16	Index of the numeric scale
<b>Return value</b>	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
<b>Parameter:</b> ⇒ numericScaleName	System.String	Name of the numeric scale
<b>Return value</b>	VcNumericScale	Numeric scale

**Example Code VB.NET**

```
Dim numericScaleCltn As VcNumericScaleCollection

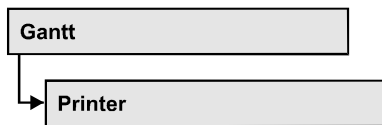
numericScaleCltn =
VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1").NumericScaleCollecti
on
numericScaleCltn.Active = numericScaleCltn.NumericScaleByName("STEP1")
```

## 852 API Reference: VcNumericScaleCollection

### Example Code C#

```
VcNumericScaleCollection numericScaleCltn =  
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1").NumericScaleCollecti  
on;  
numericScaleCltn.Active = numericScaleCltn.NumericScaleByName("STEP1");
```

## 6.58 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
- AbsoluteLeftMarginInCM
- AbsoluteRightMarginInCM
- AbsoluteTopMarginInCM
- Alignment
- Alignment
- CurrentHorizontalPagesCount
- CurrentZoomFactor
- CuttingMarks
- DefaultPrinterName
- DiagramEnabled
- DiagramEnabled
- DocumentName
- FitToPage
- FoldingMarksType
- MaxHorizontalPagesCount
- MaxVerticalPagesCount
- Orientation
- PageDescription
- PageDescriptionString
- PageFrame
- PageNumberMode
- PageNumbers
- PagePaddingEnabled
- PaperSize
- PrintDate

- PrinterName
- PrintPreviewWithFirstPage
- ReOptimizeNodesInGroupsEnabled
- ScalingMode
- TableColumnRanges
- TableTimeScaleOnAllPages
- 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	System.Double	Height of the bottom margin of the page

#### Example Code VB.NET

```
VcGanttASP1.Printer.BottomMargin = 2    ' 2 cm
VcGanttASP1.Printer.BottomMargin = 12   ' 12 cm
VcGanttASP1.Printer.BottomMargin = 100  ' 0,1 cm
VcGanttASP1.Printer.BottomMargin = 200  ' 0,2 cm
VcGanttASP1.Printer.BottomMargin = 1200 ' 1,2 cm
```

#### Example Code C#

```
vcGanttASP1.Printer.BottomMargin = 2;    // 2 cm
vcGanttASP1.Printer.BottomMargin = 12;   // 12 cm
vcGanttASP1.Printer.BottomMargin = 100;  // 0,1 cm
vcGanttASP1.Printer.BottomMargin = 200;  // 0,2 cm
vcGanttASP1.Printer.BottomMargin = 1200; // 1,2 cm
```

### 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	System.Double	Width of the left margin of the page

**Example Code VB.NET**

```
VcGanttASP1.Printer.LeftMargin = 2    ' 2 cm
VcGanttASP1.Printer.LeftMargin = 12   ' 12 cm
VcGanttASP1.Printer.LeftMargin = 100  ' 0,1 cm
VcGanttASP1.Printer.LeftMargin = 200  ' 0,2 cm
VcGanttASP1.Printer.LeftMargin = 1200 ' 1,2 cm
```

**Example Code C#**

```
vcGanttASP1.Printer.LeftMargin = 2;    // 2 cm
vcGanttASP1.Printer.LeftMargin = 12;   // 12 cm
vcGanttASP1.Printer.LeftMargin = 100;  // 0,1 cm
vcGanttASP1.Printer.LeftMargin = 200;  // 0,2 cm
vcGanttASP1.Printer.LeftMargin = 1200; // 1,2 cm
```

## 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	System.Double	Width of the right margin of the page

**Example Code VB.NET**

```
VcGanttASP1.Printer.RightMargin = 2    ' 2 cm
VcGanttASP1.Printer.RightMargin = 12   ' 12 cm
VcGanttASP1.Printer.RightMargin = 100  ' 0,1 cm
VcGanttASP1.Printer.RightMargin = 200  ' 0,2 cm
VcGanttASP1.Printer.RightMargin = 1200 ' 1,2 cm
```

**Example Code C#**

```
vcGanttASP1.Printer.RightMargin = 2;    // 2 cm
vcGanttASP1.Printer.RightMargin = 12;   // 12 cm
vcGanttASP1.Printer.RightMargin = 100;  // 0,1 cm
vcGanttASP1.Printer.RightMargin = 200;  // 0,2 cm
vcGanttASP1.Printer.RightMargin = 1200; // 1,2 cm
```

## 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	System.Double	Height of the top margin of the page

**Example Code VB.NET**

```
VcGanttASP1.Printer.TopMargin = 2      ' 2 cm
VcGanttASP1.Printer.TopMargin = 12     ' 12 cm
VcGanttASP1.Printer.TopMargin = 100    ' 0,1 cm
VcGanttASP1.Printer.TopMargin = 200    ' 0,2 cm
VcGanttASP1.Printer.TopMargin = 1200   ' 1,2 cm
```

**Example Code C#**

```
vcGanttASP1.Printer.TopMargin = 2;     // 2 cm
vcGanttASP1.Printer.TopMargin = 12;    // 12 cm
vcGanttASP1.Printer.TopMargin = 100;   // 0,1 cm
vcGanttASP1.Printer.TopMargin = 200;   // 0,2 cm
vcGanttASP1.Printer.TopMargin = 1200;  // 1,2 cm
```

## 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 **RepeatTitleAndLegend** property was set. In any other case the output will be centered.

	Data Type	Explanation
Property value	VcPrinterAlignment	Alignment of the output with its sheet <b>Default value:</b> vcPCenterCenter
	<b>Possible Values:</b>	
	.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 VB.NET**

```
VcGanttASP1.Printer.Alignment = VcPrinterAlignment.vcPTopLeft
```

**Example Code C#**

```
vcGanttASP1.Printer.Alignment = VcPrinterAlignment.vcPTopLeft;
```

## 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 **TableTimeScaleOnAllPages** property was set. In any other case the output will be centered.

	Data Type	Explanation
<b>Property value</b>	VcPrinterAlignment	Alignment of the output with its sheet <b>Default value:</b> vcPCenterCenter
	<b>Possible Values:</b>	
	.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 VB.NET

```
VcGantt1.Printer.Alignment = VcPrinterAlignment.vcPTopLeft
```

### Example Code C#

```
vcGantt1.Printer.Alignment = VcPrinterAlignment.vcPTopLeft;
```

## 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 **CurrentVerticalPagesCount** and **MaxHorizontalPagesCount**.

	Data Type	Explanation
<b>Property value</b>	System.Int32	Current number of pages counted in horizontal 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	System.Double	Current 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	System.Boolean	Cutting marks are (True) / are not (False) printed <b>Default value:</b> False

### Example Code VB.NET

```
VcGanttASP1.Printer.CuttingMarks = True
```

### Example Code C#

```
vcGanttASP1.Printer.CuttingMarks = true;
```

## 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	System.String	Name of current default printer

## DiagramEnabled

Property of VcPrinter

This property lets you specify whether the diagram (timescale and layers) shall be also printed or not.

	Data Type	Explanation
Property value	System.Boolean	Diagram is (True) / is not (False) printed <b>Default value:</b> True

### Example Code VB.NET

```
VcGantt1.Printer.DiagramEnabled = True
```

### Example Code C#

```
vcGantt1.Printer.DiagramEnabled = true;
```

## 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	System.Boolean	Diagram is (True) / is not (False) printed <b>Default value:</b> 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	System.String	Name of document <b>Default value:</b> " "

## FitToPage

Property of VcPrinter

This property lets you set or retrieve, whether (True) the diagram is to be printed to a set of pages defined by the properties **MaxHorizontalPagesCount** and **MaxVerticalPagesCount**, or whether (False) it is to be printed by the enlargement set by the **ZoomFactor** property.

	Data Type	Explanation
Property value	System.Boolean	Diagram is printed on a defined set of pages/is printed in a defined enlargement.

### Example Code VB.NET

```
VcGanttASP1.Printer.FitToPage = True
```

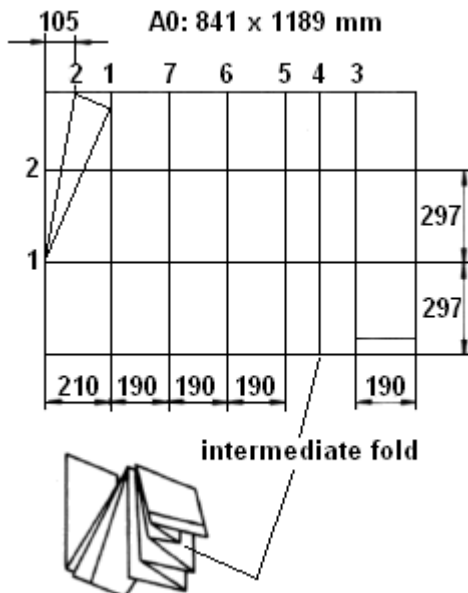
### Example Code C#

```
vcGanttASP1.Printer.FitToPage = true;
```

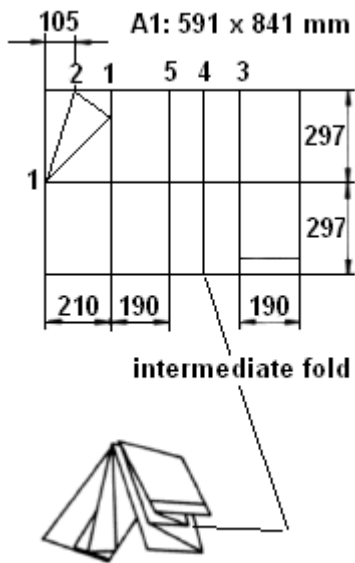
## FoldingMarksType

Property of VcPrinter

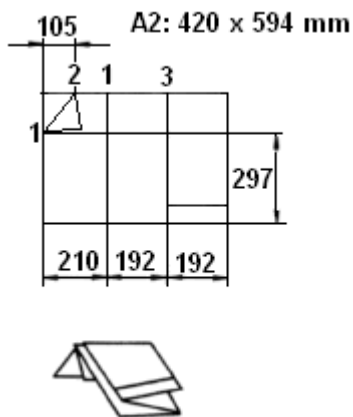
This property lets you set or retrieve folding marks according to DIN 824. The folding marks allow to fold paper sheets of the German DIN-A standard:



Folding of the DIN-A-0 format

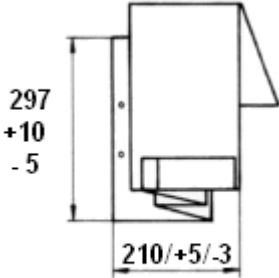
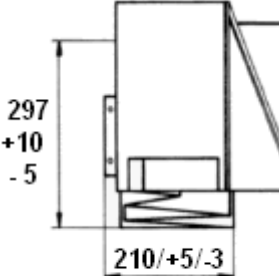
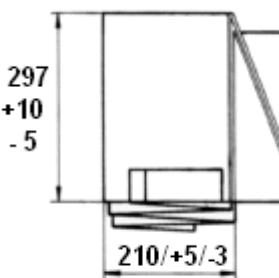


Folding of the DIN-A-1 format



Folding of the DIN-A-2 format

	Data Type	Explanation
Property value	VcFoldingMarksType	Folding marks <b>Default value:</b> vcFMTNone
	<b>Possible Values:</b>	

.vcFMTDIN824FormA 65	Folding marks according to DIN824-A: the drawing can be punched and filed directly to a folder.
	
	<b>DIN 824-A way of folding</b>
.vcFMTDIN824FormB 66	Folding marks according to DIN824-B: the chart can be punched and filed to a folder by a flexi filing fastener.
	
	<b>DIN 824-B way of folding</b>
.vcFMTDIN824FormC 67	Folding marks according to DIN824-C: the folded chart is not to be punched but to be put into a sheet protector.
	
	<b>DIN 824-C way of folding</b>
.vcFMTNone 0	No folding marks

## 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**.

	Data Type	Explanation
<b>Property value</b>	System.Int32	Maximum number of pages counted in horizontal direction <b>Default value:</b> 1

**Example Code VB.NET**

```
VcGanttASP1.Printer.MaxHorizontalPagesCount = 4
```

**Example Code C#**

```
vcGanttASP1.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**.

	Data Type	Explanation
<b>Property value</b>	System.Int32	Maximum number of pages counted in vertical direction <b>Default value:</b> 1

**Example Code VB.NET**

```
VcGanttASP1.Printer.MaxVerticalPagesCount = 4
```

**Example Code C#**

```
vcGanttASP1.Printer.MaxVerticalPagesCount = 4;
```

## Orientation

**Property of VcPrinter**

This property lets you set or retrieve the orientation of the output.

	Data Type	Explanation
<b>Property value</b>	VcOrientation	Orientation <b>Default value:</b> VcPortrait
	<b>Possible Values:</b> .vcLandscape 42 .vcPortrait 41	Printing orientation <b>landscape</b> Printing orientation <b>portrait</b>



### Example Code VB.NET

```
VcGanttASP1.Printer.Orientation = VcOrientation.vcLandscape
```

### Example Code C#

```
vcGanttASP1.Printer.Orientation = VcOrientation.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
<b>Property value</b>	System.Boolean	Page description is (True) / is not printed (False) <b>Default value:</b> False

### Example Code VB.NET

```
VcGanttASP1.Printer.PageDescription = True
```

### Example Code C#

```
vcGanttASP1.Printer.PageDescription = true;
```

## PageDescriptionString

### Property of VcPrinter

This property lets you set or retrieve a page description 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 below codes which will be replaced by the corresponding 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
<b>Property value</b>	System.String	Page description <b>Default value:</b> Empty string ""

**Example Code VB.NET**

```
VcGanttASP1.Printer.PageDescriptionString = "Gantt-Graphics"
```

**Example Code C#**

```
vcGanttASP1.Printer.PageDescriptionString = "Gantt-Graphics";
```

## 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 **TableTimeScaleOnAllPages** 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
<b>Property value</b>	System.Boolean	Page frame is (True) / is not (False) displayed <b>Default value:</b> True

**Example Code VB.NET**

```
VcGanttASP1.Printer.PageFrame = True
```

**Example Code C#**

```
vcGanttASP1.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
<b>Property value</b>	VcPageNumberMode	Mode of page numbering <b>Default value:</b> vcPRowColumn
	<b>Possible Values:</b> .vcPageNOfM 1597 .vcPRowColumn 1596	"Page N of M pages" "x.y" (row no./column no.).

**Example Code VB.NET**

```
Dim printer As VcPrinter
printer.Orientation = VcOrientation.vcLandscape
printer.PageNumberMode = VcPageNumberMode.vcPageNOfM
printer.PageNumbers = True
printer.FitToPage = False
```

**Example Code C#**

```
VcPrinter printer = vcGantt1.Printer;
printer.Orientation = VcOrientation.vcLandscape;
printer.PageNumberMode = VcPageNumberMode.vcPageNOfM;
printer.PageNumbers = true;
printer.FitToPage = false;
```

## 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
<b>Property value</b>	System.Boolean	Page numbers are (True) / are not (False) printed <b>Default value:</b> False

**Example Code VB.NET**

```
VcGanttASP1.Printer.PageNumbers = True
```

**Example Code C#**

```
vcGanttASP1.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	System.Boolean	Space between diagram and boxes for legend/title is (True) / is not (False) left <b>Default value:</b> True

**Example Code VB.NET**

```
VcGanttASP1.Printer.PagePaddingEnabled = True
```

**Example Code C#**

```
vcGanttASP1.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	VcPaperSize	Paper size
	<b>Possible Values:</b>	
	.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 VB.NET**

```
VcGanttASP1.Printer.PaperSize = VcPaperSize.vcDIN_A3
```

**Example Code C#**

```
vcGanttASP1.Printer.PaperSize = VcPaperSize.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	System.Boolean	Print date is/is not set

### Example Code VB.NET

```
VcGanttASP1.Printer.PrintDate = True
```

### Example Code C#

```
vcGanttASP1.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.

	Data Type	Explanation
Property value	System.String	Printer name

## PrintPreviewWithFirstPage

### 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	System.Boolean	At the start of the page preview: only first page of the diagram (True) / all pages of the diagram (False)

### Example Code VB.NET

```
Dim printer As VcPrinter  
printer.Orientation = VcOrientation.vcLandscape  
printer.PrintPreviewWithFirstPage = True  
printer.FitToPage = False
```

### Example Code C#

```
VcPrinter printer = vcGantt1.Printer;  
printer.Orientation = VcOrientation.vcLandscape;  
printer.PrintPreviewWithFirstPage = true;  
printer.FitToPage = false;
```

## 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	System.Boolean	With the <b>TimeScaleAdjustment</b> property switched on: optimized groups are (True)/are not (False) reoptimized for output or print preview <b>Default value:</b> False

### Example Code VB.NET

```
VcGanttASP1.Printer.ReOptimizeNodesInGroupsEnabled = True
```

### Example Code C#

```
vcGanttASP1.Printer.ReOptimizeNodesInGroupsEnabled = true;
```

## ScalingMode

Read Only 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 **MaxHorizontalPagesCount** 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 **vcFitToPageCount**, you can achieve at covering the pages evenly by the property **AdjustTimeScale**.

	Data Type	Explanation
Property value	VcScalingMode	<b>Default value:</b> Skalierungsmodus für den Ausdruck  <b>Possible Values:</b> .vcFitToPageCount 1 .vcZoomFactor 0 .vcZoomWithHorizontalFit 2
		Scaling mode "Fit to Page" Scaling mode: "Zoomfactor". Scaling mode "Combined Fit"

## 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	System.String	Number of table columns which are printed

### Example Code VB.NET

```
VcGanttASP1.Printer.TableColumnRanges = "1;5-7;3"
```

### Example Code C#

```
vcGanttASP1.TableColumnRanges = "1;5-7;3";
```

## TableTimeScaleOnAllPages

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	System.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)

### Example Code VB.NET

```
VcGantt1.Printer.TableTimeScaleOnAllPages = True
```

### Example Code C#

```
vcGantt1.Printer.TableTimeScaleOnAllPages = true;
```

## TimeScaleAdjustment

Property of VcPrinter

This property improves utilization of the printing pages:

- If the scaling type **fit to page** is selected: The zoom factor is calculated in a way that utilizes the selected number of pages in the dimension of height. The time scale will be downsized or enlarged to adapt to the selected number of pages in the dimension of width.
- If **scaling by zoom factor** is selected: The time scale will be downsized or enlarged so that the selected number of pages is used to full capacity in the dimension of width.

	Data Type	Explanation
<b>Parameter:</b> ↔ Value	System.Int.32	Adjustment of time scale
<b>Property value</b>	System.Boolean	Adjustment of time scale <b>Default value:</b> False

**Example Code VB.NET**

```
VcGanttASP1.Printer.TimeScaleAdjustment = True
```

**Example Code C#**

```
vcGanttASP1.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
<b>Property value</b>	System.Double	Zoom factor of the diagram

**Example Code VB.NET**

```
VcGanttASP1.Printer.ZoomFactor = 150
```

**Example Code C#**

```
vcGanttASP1.Printer.ZoomFactor = 150;
```



## 6.59 VcRect

Rect

An object of the type **VcRect** designates a rectangle object and is only available in VcInPlaceEditorShowing.

### 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	System.Int32	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	System.Int32	Height of the rectangle

## Left

Property of VcRect

This property returns/sets the left coordinate of the VcRect object.

	Data Type	Explanation
Property value	System.Int32	Position of the left border of the rectangle

### Example Code VB.NET

```

Private Sub VcGanttASP1_VcInPlaceEditorShowing(ByVal sender As Object, ByVal
e As NETRONIC.XGantt.VcInPlaceEditorShowingEventArgs) Handles
VcGanttASP1.VcInPlaceEditorShowing
    Dim node As VcNode
    node = e.EditObject
    If e.EditObjectType = VcObjectType.vcObjTypeNodeInTable Then
        e.ReturnStatus = VcReturnStatus.vcRetStatFalse
        Select Case e.FieldIndex
            Case 1 'Name
                TextBox1.Left = e.FldRectVisible.Left + VcGanttASP1.Left
                TextBox1.Top = e.FldRectVisible.Top + VcGanttASP1.Top
                TextBox1.Width = e.FldRectVisible.Width
                TextBox1.Height = e.FldRectVisible.Height
                TextBox1.Text = node.DataField(0)
                TextBox1.Visible = True
                TextBox1.Focus()
            Case 2, 3 'Start or End
                DateTimePicker1.Left = e.FldRectVisible.Left +
VcGanttASP1.Left
                DateTimePicker1.Top = e.FldRectVisible.Top + VcGanttASP1.Top
                DateTimePicker1.Value = node.DataField(0)
                DateTimePicker1.Visible = True
                DateTimePicker1.Focus()
            Case 13 'Employee
                ComboBox1.Left = e.FldRectVisible.Left + VcGanttASP1.Left
                ComboBox1.Top = e.FldRectVisible.Top + VcGanttASP1.Top
                ComboBox1.Width = e.FldRectVisible.Width
                ComboBox1.Height = e.FldRectVisible.Height
                ComboBox1.Text = node.DataField(0)
                ComboBox1.Visible = True
                ComboBox1.Focus()
        End Select
    End If
End Sub

```

**Example Code C#**

```

private void vcGanttASP1_VcInPlaceEditorShowing(object sender,
NETRONIC.XGantt.VcInPlaceEditorShowingEventArgs e)
{
    VcNode node = (VcNode)e.EditObject;
    if (e.EditObjectType == VcObjectType.vcObjTypeNodeInTable)
    {
        e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
        switch (e.FieldIndex)
        {
            case 1: //Name
                textBox1.Left = e.FldRectVisible.Left + vcGanttASP1.Left;
                textBox1.Top = e.FldRectVisible.Top + vcGanttASP1.Top;
                textBox1.Width = e.FldRectVisible.Width;
                textBox1.Height = e.FldRectVisible.Height;
                textBox1.Text = Convert.ToString(node.get_DataField(0));
                textBox1.Visible = true;
                textBox1.Focus();
                break;
            case 2: //Start or end
                dateTimePicker1.Left = e.FldRectVisible.Left + vcGanttASP1.Left;
                dateTimePicker1.Top = e.FldRectVisible.Top + vcGanttASP1.Top;
                dateTimePicker1.Value = Convert.ToDateTime(node.get_DataField(0));
                dateTimePicker1.Visible = true;
                dateTimePicker1.Focus();
                break;
            case 13: //Employee
                comboBox1.Left = e.FldRectVisible.Left + vcGanttASP1.Left;
                comboBox1.Top = e.FldRectVisible.Top + vcGanttASP1.Top;
                comboBox1.Width = e.FldRectVisible.Width;
                comboBox1.Height = e.FldRectVisible.Height;
                comboBox1.Text = Convert.ToString(node.get_DataField(0));
                comboBox1.Visible = true;
                comboBox1.Focus();
                break;
        }
    }
}

```

**Right****Property of VcRect**

This property returns/sets the right coordinate of the VcRect object.

	Data Type	Explanation
<b>Property value</b>	System.Int32	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	System.Int32	Position of the top border of the rectangle

**Example Code VB.NET**

```
DateTimePicker1.Top = e.FldRectVisible.Top + VcGanttASP1.Top
```

**Example Code C#**

```
dateTimePicker1.Top = e.FldRectVisible.Top + vcGanttASP1.Top;
```

## Width

**Read Only Property of VcRect**

This property returns the width of the VcRect object.

	Data Type	Explanation
Property value	System.Int32	Width of the rectangle

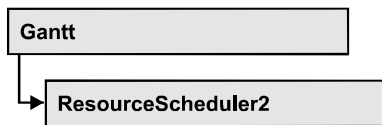
**Example Code VB.NET**

```
Text1.Width = fldRectVisible.Width
```

**Example Code C#**

```
textBox1.Width = e.FldRectVisible.Width;
```

## 6.60 VcResourceScheduler2



The ResourceScheduler2 is a substantial enhancement of ResourceScheduler1 (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 contain 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
- AssignmentOperationIDFieldIndex
- AssignmentResourceIDFieldIndex
- AssignmentResourceSelectionStrategyFieldIndex
- BaseTimeUnit
- BaseTimeUnitsPerStep
- DataRecordEventsEnabled
- DefaultOperationMaximumInterruptionTime
- DefaultResourceCalendarName
- FullUsageOfPlanningUnitsEnabled
- LinkDataTableName
- LinkDurationFieldIndex
- LinkPredecessorOperationIDFieldIndex
- LinkPredecessorTaskIDFieldIndex
- LinkSuccessorOperationIDFieldIndex
- LinkSuccessorTaskIDFieldIndex
- OperationDataTableName
- OperationLoadPerItemFieldIndex
- OperationMaximumInterruptionTimeFieldIndex
- OperationMinimumSupplementTimeFieldIndex
- OperationOverlapQuantityFieldIndex
- OperationPostLoadFieldIndex
- OperationPreparationLoadFieldIndex
- OperationResultEndDateFieldIndex
- OperationResultPostEndDateFieldIndex
- OperationResultPreparationStartDateFieldIndex
- OperationResultProcessingTimeFieldIndex
- OperationResultStartDateFieldIndex
- OperationResultStatusFieldIndex
- OperationRouteFieldIndex

- OperationSequenceNumberFieldIndex
- OperationStartLockDateFieldIndex
- OperationTaskIDFieldIndex
- OperationWorkInProgressFieldIndex
- 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
- WorkInProgressType
- WritingDebugFilesEnabled

## Methods

- DetermineIDOfFirstOperationByTaskID
- DetermineIDOfLastOperationByTaskID
- Process

---

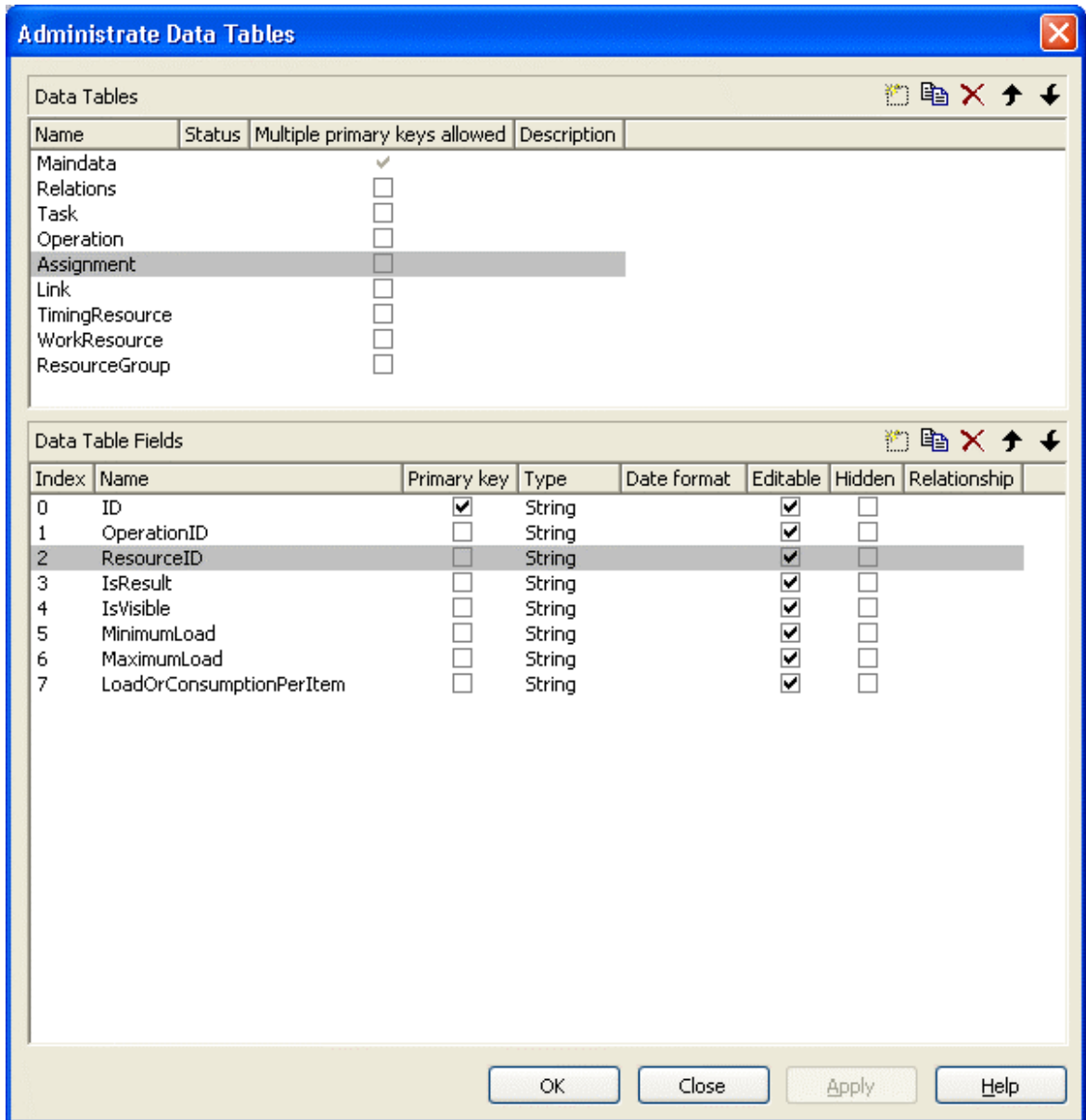
## Properties

### 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	System.String	Name of the assignment data table <b>Default value:</b> Empty string

**Example Code VB.NET**

```
VcGantt1.ResourceScheduler2.AssignmentDataTableName = "Assignment"
```

**Example Code C#**

```
vcGantt1.ResourceScheduler2.AssignmentDataTableName("Assignment");
```

## AssignmentIsResultFieldIndex

Property of VcResourceScheduler2

This property lets you set or retrieve the index of a data field in the assignment data table where VARCHART XGantt notes whether the corresponding data set was generated by itself. 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	System.Int32	<p>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.</p> <p>{-1...NumberOfFieldsInAssignmentDataTable -1}. By setting the index to -1, no data field of the assignment data table will be assigned to this property.</p> <p><b>Default value:</b> -1</p>

### Example Code VB.NET

```
VcGantt1.ResourceScheduler2.AssignmentIsResultFieldIndex = 3
```

### Example Code C#

```
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
<b>Property value</b>	System.Int32	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 operations table will be assigned to this property.  <b>Default value:</b> -1

### Example Code VB.NET

```
VcGantt1.ResourceScheduler2.AssignmentIsVisibleFieldIndex = 4
```

### Example Code C#

```
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 **TaskQuantityFieldIndex**). 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
<b>Property value</b>	System.Int32	Index of the data set in the assignment data table that is designated to hold the value.  {-1...NumberOfFieldsInOperationsDataTable -1}. By setting the index to -1, no data field of the operations table will be assigned to this property.  <b>Default value:</b> -1

### Example Code VB.NET

```
VcGantt1.ResourceScheduler2.AssignmentLoadOrConsumptionPerItemFieldIndex = 7
```

### Example Code C#

```
vcGantt1.ResourceScheduler2.AssignmentLoadOrConsumptionPerItemFieldIndex = 7;
```

## 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 equal 100.

Values between 1 and 99 in the data field will disable the properties **FullUsageOfPlanningUnitsEnabled** and **OperationMaximumInterruption-TimeFieldIndex**.

Also see **AssignmentMinimumLoadFieldIndex**.

	Data Type	Explanation
Property value	System.Int32	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. <b>Default value:</b> -1

### Example Code VB.NET

```
VcGantt1.ResourceScheduler2.AssignmentMaximumLoadFieldIndex = 6
```

### Example Code C#

```
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
<b>Property value</b>	System.Int32	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.  <b>Default value:</b> -1

#### Example Code VB.NET

```
VcGantt1.ResourceScheduler2.AssignmentMinimumLoadFieldIndex = 5
```

#### Example Code C#

```
vcGantt1.ResourceScheduler2.AssignmentMinimumLoadFieldIndex = 5;
```

## AssignmentOperationIDFieldIndex

Property of VcResourceScheduler2

This property lets you set or retrieve the index to a data field in the assignment data table which holds the ID of an operation. In the picture referring to **AssignmentDataTableName**, the field index for example is 1. This property needs to be set to a figure unequal to -1 before calling the method **Process**.

	Data Type	Explanation
<b>Property value</b>	System.Int32	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.  <b>Default value:</b> -1

#### Example Code VB.NET

```
VcGantt1.ResourceScheduler2.AssignmentOperationIDFieldIndex = 1
```

#### Example Code C#

```
vcGantt1.ResourceScheduler2.AssignmentOperationIDFieldIndex = 1;
```

## AssignmentResourceIDFieldIndex

Property of VcResourceScheduler2

This indexed property lets you set or retrieve the index of a data field in the assignment table that holds IDs of resources. In the picture referring to **AssignmentDataTableName**, 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
<b>Parameter:</b> ⇒ resourceTableIndex	System.Int16	Index of a resource table according to the assignments made by <b>ResourceDataTableName</b>  {0...24}
<b>Property value</b>	System.Int32	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.  <b>Default value:</b> -1

### Example Code VB.NET

```
VcGantt1.ResourceScheduler2.AssignmentResourceIDFieldIndex(0) = 2
```

### Example Code C#

```
vcGantt1.ResourceScheduler2.set_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 respective assignment to a resource group. If this field is empty at a resource or 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	System.Int32	<p>Index of the data field in the assignment data table that is designated to hold the data of the planning strategy.</p> <p>{-1...NumberOfFieldsInTaskDataTable -1}. By setting the index to -1, no data field of the task data table will be assigned to this property.</p> <p><b>Default value:</b> -1</p>

## BaseTimeUnit

Property of VcResourceScheduler2

This property lets you set or retrieve the basic time unit for resource scheduling, which may differ from the basic time unit set by **VcGantt.TimeUnit**. The values of the capacity, work load and stock curves refer to the base unit defined here.

	Data Type	Explanation
Property value	VcTimeUnit  <b>Possible Values:</b> .vcDay 5 .vcHour 6 .vcMinute 7 .vcSecond 8	<p>Time unit</p> <p><b>Default value:</b> Value, which was set during design time by vcGantt.TimeUnit. If no setting was made, the value is <b>vcDay</b>.</p> <p>Time unit <b>day</b>            Time unit <b>hour</b>            Time unit <b>minute</b>            Time unit <b>second</b></p>

**Example Code VB.NET**

```
VcResourceScheduler2.BaseTimeUnit = VcTimeUnit.vcMinute
VcResourceScheduler2.BaseTimeUnitsPerStep = 15
```

**Example Code C#**

```
vcGantt1.ResourceScheduler2.BaseTimeUnit = VcTimeUnit.vcMinute;
vcGantt1.ResourceScheduler2.BaseTimeUnitsPerStep = 15;
```

## BaseTimeUnitsPerStep

**Read Only Property of VcResourceScheduler2**

This property lets you set or retrieve 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 **VcResourceScheduler2.BaseTimeUnit**.

	Data Type	Explanation
<b>Property value</b>	System.Int16	Number of time units per step <b>Default value: 1</b>

**Example Code VB.NET**

```
VcResourceScheduler2.BaseTimeUnit = VcTimeUnit.vcMinute
VcResourceScheduler2.BaseTimeUnitsPerStep = 30
```

**Example Code C#**

```
vcGantt1.ResourceScheduler2.BaseTimeUnit = VcTimeUnit.vcMinute;
vcGantt1.ResourceScheduler2.BaseTimeUnitsPerStep = 30;
```

## DataRecordEventsEnabled

**Property of VcResourceScheduler2**

If this property is set to **true**, events will be triggered that indicate data modifications during the process method: DataRecordModifying, DataRecordModified, DataRecordCreating, DataRecordCreated, DataRecordDeleting and DataRecordDeleted.

	Data Type	Explanation
<b>Property value</b>	System.Boolean	<b>true</b> : events are triggered. <b>false</b> : events are not triggered. <b>Default value: false</b>



**Example Code VB.NET**

```
VcGantt1.ResourceScheduler2.DataRecordEventsEnabled = True
```

**Example Code C#**

```
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 value applies if the property **OperationMaximumInterruptionTimeFieldIndex** 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 is allowed.

This property will be disabled by setting the maximum load to less than 100% (see property **AssignmentMaximumLoadFieldIndex**).

	Data Type	Explanation
Property value	System.Int32	Number of base time units <b>Default value: 0</b>

**Example Code VB.NET**

```
VcGantt1.ResourceScheduler2.DefaultOperationMaximumInterruptionTime = 1
```

**Example Code C#**

```
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 **VcResourceScheduler2.ResourceCalendarNameFieldIndex** and **VcResourceScheduler2.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	System.String	Name of the calendar <b>Default value:</b> Empty String

**Example Code VB.NET**

```
VcGantt1.ResourceScheduler2.DefaultResourceCalendarName = ""
```

**Example Code C#**

```
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	System.Boolean	<b>true:</b> remaining capacities are used. <b>false:</b> remaining capacities are not used. <b>Default value:</b> true

**Example Code VB.NET**

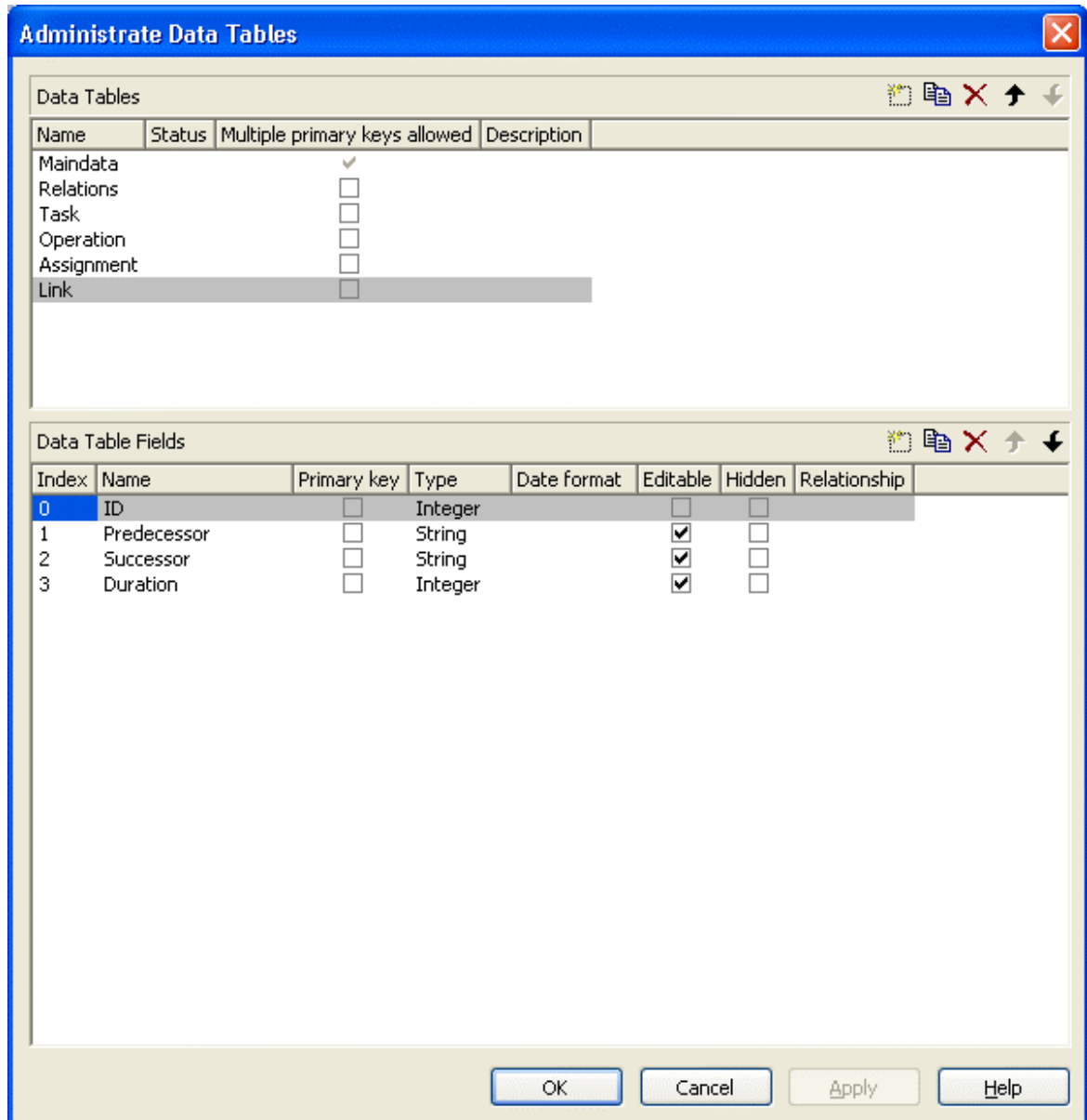
```
VcGantt1.ResourceScheduler2.FullUsageOfPlanningUnitsEnabled = True
```

**Example Code C#**

```
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	System.String	Name of the link data table <b>Default value:</b> Empty string

**Example Code VB.NET**

```
VcGantt1.ResourceScheduler2.NodeDataTableName = "Node"
```

**Example Code C#**

```
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
<b>Property value</b>	System.Int32	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.  <b>Default value:</b> -1

## LinkPredecessorOperationIDFieldIndex

Read Only 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
<b>Property value</b>	System.Int32	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.

### Example Code VB.NET

```
VcGantt1.ResourceScheduler2.LinkPredecessorOperationIDFieldIndex = 1
```

### Example Code C#

```
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 that holds 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	System.Int32	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.  <b>Default value:</b> -1

### Example Code VB.NET

```
VcGantt1.ResourceScheduler2.NodePredecessorTaskIDFieldIndex = 1
```

### Example Code C#

```
vcGantt1.ResourceScheduler2.LinkPredecessorTaskIDFieldIndex = 1;
```

## LinkSuccessorOperationIDFieldIndex

Read Only 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
<b>Property value</b>	System.Int32	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.

#### Example Code VB.NET

```
VcGantt1.ResourceScheduler2.LinkSuccessorOperationIDFieldIndex = 1
```

#### Example Code C#

```
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
<b>Property value</b>	System.Int32	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. <b>Default value:</b> -1

#### Example Code VB.NET

```
VcGantt1.ResourceScheduler2.NodeSuccessorTaskIDFieldIndex = 2
```

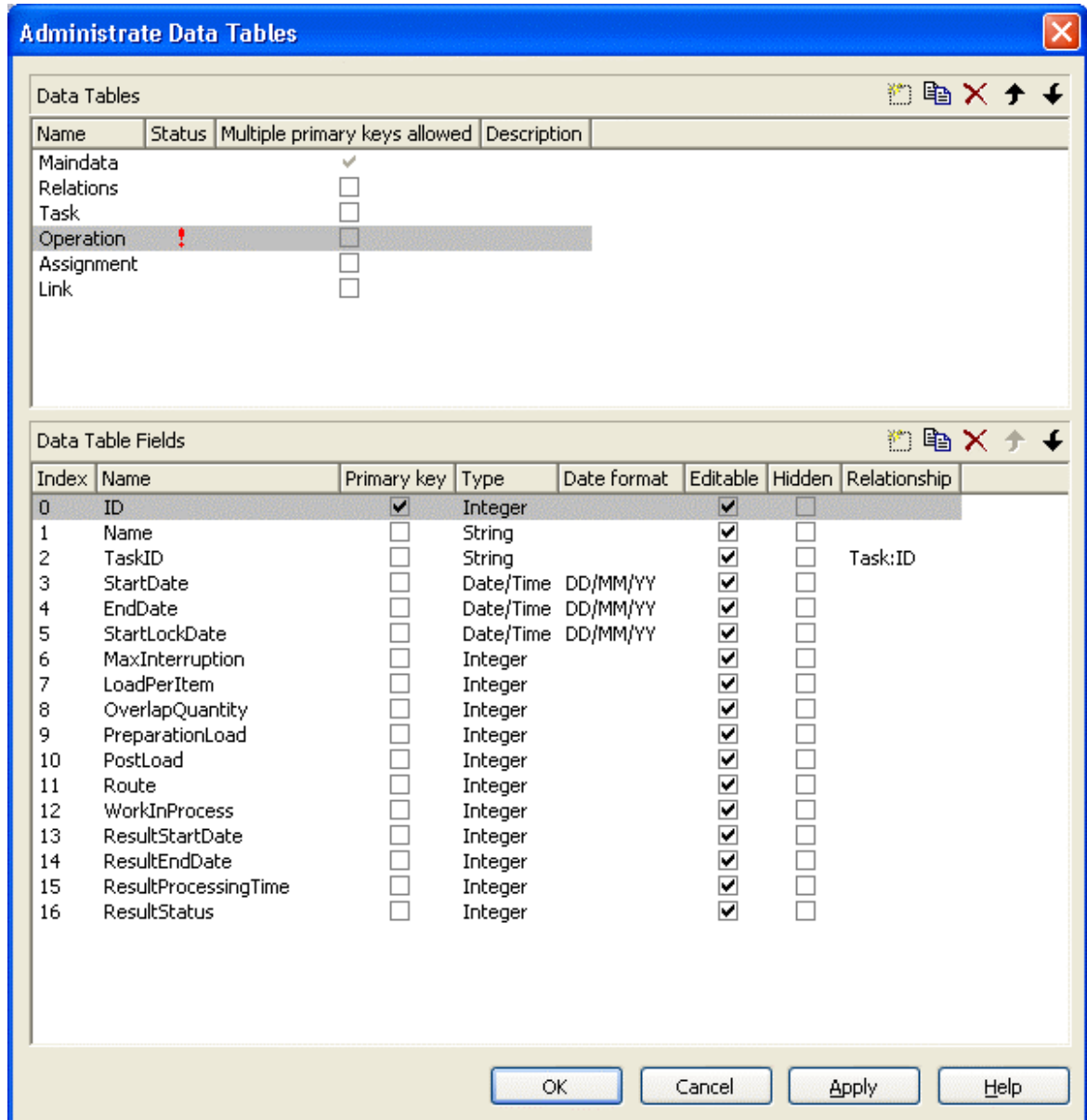
#### Example Code C#

```
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
<b>Property value</b>	System.String	Name of the operation data table <b>Default value:</b> Empty String

### Example Code VB.NET

```
VcGantt1.ResourceScheduler2.OperationDataTableName = "Operation"
```

**Example Code C#**

```
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
<b>Property value</b>	System.Int32	<p>Index of the data field in the operation data table that is designated to hold the values of the load.</p> <p>{-1...NumberOfFieldsInOperationsDataTable -1}. By setting the index to -1, no data field of the operation data table will be assigned to this property.</p> <p><b>Default value:</b> -1</p>

**Example Code VB.NET**

```
VcGantt1.ResourceScheduler2.OperationLoadPerItemFieldIndex = 10
```

**Example Code C#**

```
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 **DefaultOperationMaximumInterruptionTime** 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
Property value	System.Int32	<p>Index of the data field in the operation data table that is designated to hold the maximum interruption time.</p> <p>{-1...NumberOfFieldsInOperationDataTable -1}. By setting the index to -1, no data field of the operation data table will be assigned to this property.</p> <p><b>Default value:</b> -1</p>

#### Example Code VB.NET

```
VcGantt1.ResourceScheduler2.OperationMaximumInterruptionTimeFieldIndex = 9
```

#### Example Code C#

```
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 the designated field is a number that represents base time units (s. property **BaseTimeUnit**). In the picture referring to **OperationDataTableName**, the field index for example is 7.

Please also see **OperationMaximumSupplementLoadFieldIndex**, **OperationPreparationLoadFieldIndex** and **OperationPostLoadFieldIndex**.

	Data Type	Explanation
Property value	System.Int32	<p>Index of the data field in the operation data table that is designated to hold the minimum supplement time.</p> <p>{-1...NumberOfFieldsInOperationDataTable -1}. By setting the index to -1, no data field of the operation data table will be assigned to this property.</p> <p><b>Default value:</b> -1</p>

**Example Code VB.NET**

```
VcGantt1.ResourceScheduler2.OperationMinimumSupplementTimeFieldIndex = 7
```

**Example Code C#**

```
vcGantt1.ResourceScheduler2.OperationMinimumSupplementTimeFieldIndex = 7;
```

## OperationOverlapQuantityFieldIndex

Property of VcResourceScheduler2

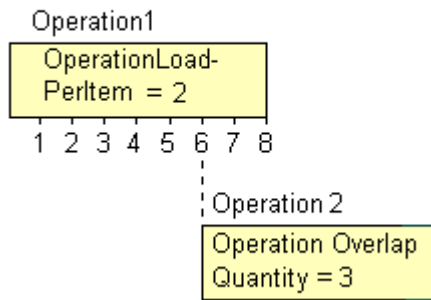
This property lets you set or retrieve the index of a data field in the operation data table that holds the 'overlap' quantity of an operation. Overlapping can only occur in tasks that were scheduled according to the strategy ASAP. This is the field to make succeeding resources overlap, which is useful 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.

## 898 API Reference: VcResourceScheduler2

Task Quantity = 4



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 **ResourceEfficiencyFieldIndex**; the same is valid for **AssignmentMaximumLoadFieldIndex**.

	Data Type	Explanation
Property value	System.Int32	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.  <b>Default value:</b> -1

### Example Code VB.NET

```
VcGantt1.ResourceScheduler2.OperationOverlapQuantityFieldIndex = 11
```

### Example Code C#

```
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 Please also see **OperationPreparationTimeFieldIndex**, **OperationMaximumSupplementTimeFieldIndex** and **OperationMinimumSupplementTimeFieldIndex**.

	Data Type	Explanation
Property value	System.Int32	Index of the data field in the operation data table that is designated to hold 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.  <b>Default value:</b> -1

### Example Code VB.NET

```
VcGantt1.ResourceScheduler2.OperationPostTimeFieldIndex = 13
```

### Example Code C#

```
vcGantt1.ResourceScheduler2.OperationPostTimeFieldIndex = 13;
```

## 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**, **OperationMaximumSupplementTimeFieldIndex** and **OperationMinimumSupplementTimeFieldIndex**.

	Data Type	Explanation
Property value	System.Int32	Index of the data field in the operation data table that is designated to hold 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.  <b>Default value:</b> -1

**Example Code VB.NET**

```
VcGantt1.ResourceScheduler2.OperationPreparationTimeFieldIndex = 12
```

**Example Code C#**

```
vcGantt1.ResourceScheduler2.OperationPreparationTimeFieldIndex = 12;
```

## 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**, **OperationResultProcessingTimeFieldIndex** and **OperationResultEndDateFieldIndex** need to be set to a value unequal to -1.

	Data Type	Explanation
Property value	System.Int32	Index of the data field in the operation data table that is designated to hold the values of the end date.  {-1...NumberOfFieldsInOperationsDataTable -1}. By setting the index to -1, no data field of the operation data table will be assigned to this property.  <b>Default value:</b> -1

**Example Code VB.NET**

```
VcGantt1.ResourceScheduler2.OperationResultEndDateFieldIndex = 17
```

**Example Code C#**

```
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
<b>Property value</b>	System.Int32	<p>Index of the data field in the operation data table that is designated to hold the end date of the post time.</p> <p>{-1...NumberOfFieldsInOperationDataTable -1}. By setting the index to -1, no data field of the operation data table will be assigned to this property.</p> <p><b>Default value:</b> -1</p>

### Example Code VB.NET

```
VcGantt1.ResourceScheduler2.OperationResultPostEndDateFieldIndex = 15
```

### Example Code C#

```
vcGantt1.ResourceScheduler2.OperationPostEndDateFieldIndex = 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
<b>Property value</b>	System.Int32	<p>Index of the data field in the operation data table that is designated to hold the start date of the preparation phase.</p> <p>{-1...NumberOfFieldsInOperationDataTable -1}. By setting the index to -1, no data field of the operation data table will be assigned to this property.</p> <p><b>Default value:</b> -1</p>

### Example Code VB.NET

```
VcGantt1.ResourceScheduler2.OperationResultPreparationStartDateFieldIndex = 10
```

### Example Code C#

```
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. In 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	System.Int32	<p>Index of the data field in the operation data table that is designated to hold the values of the processing time.</p> <p>{-1...NumberOfFieldsInOperationsDataTable -1}. By setting the index to -1, no data field of the operation data table will be assigned to this property.</p> <p><b>Default value:</b> -1</p>

### Example Code VB.NET

```
VcGantt1.ResourceScheduler2.OperationResultProcessingTimeFieldIndex = 18
```

### Example Code C#

```
vcGantt1.ResourceScheduler2.OperationResultProcessingTimeFieldIndex = 18;
```

## 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	System.Int32	Index of the data field in the operation data table that is designated to hold the values of the start date.  {-1...NumberOfFieldsInOperationsDataTable -1}. By setting the index to -1, no data field of the operations table will be assigned to this property.  <b>Default value:</b> -1

**Example Code VB.NET**

```
VcGantt1.ResourceScheduler2.OperationResultStartDateFieldIndex = 16
```

**Example Code C#**

```
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	System.Int32	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.  <b>Default value:</b> -1



### Example Code VB.NET

```
VcGantt1.ResourceScheduler2.OperationResultStatusFieldIndex = 19
```

### Example Code C#

```
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 is 14.

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	System.Int32	<p>Index of the data field in the operation data table that is designated to hold the name of the route.</p> <p>{-1...NumberOfFieldsInOperationDataTable -1}. By setting the index to -1, no data field of the operation data table will be assigned to this property.</p> <p><b>Default value:</b> -1</p>

### Example Code VB.NET

```
VcGantt1.ResourceScheduler2.OperationRouteFieldIndex = 14
```

### Example Code C#

```
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 **OperationDataTableName**, the field index for example is 6.

	Data Type	Explanation
<b>Property value</b>	System.Int32	<p>Index of the data field in the operation data table that is designated to hold the sequence values.</p> <p>{-1...NumberOfFieldsInOperationDataTable -1}. By setting the index to -1, no data field of the operation data table will be assigned to this property.</p> <p><b>Default value:</b> -1</p>

**Example Code VB.NET**

```
VcGantt1.ResourceScheduler2.OperationSequenceNumberFieldIndex = 6
```

**Example Code C#**

```
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 **OperationWorkInProgressFieldIndex**).

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
<b>Property value</b>	System.Int32	<p>Index of the data field in the operation data table that is designated to hold the lock date.</p> <p>{-1...NumberOfFieldsInOperationDataTable -1}. By setting the index to -1, no data field of the operation data table will be assigned to this property.</p> <p><b>Default value:</b> -1</p>

### Example Code VB.NET

```
VcGantt1.ResourceScheduler2.OperationStartLockDateFieldIndex = 5
```

### Example Code C#

```
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. In 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 **OperationSequenceNumberDataFieldIndex**.

	Data Type	Explanation
Property value	System.Int32	<p>Index of the data field in the operation data table that is designated to hold the task ID.</p> <p>{-1...NumberOfFieldsInOperationDataTable -1}. By setting the index to -1, no data field of the operation data table will be assigned to this property.</p> <p><b>Default value:</b> -1</p>

### Example Code VB.NET

```
VcGantt1.ResourceScheduler2.OperationTaskIDFieldIndex = 2
```

### Example Code C#

```
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 percent, i.e. a value between 0 and 100. 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	System.Int32	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.  <b>Default value:</b> -1

#### Example Code VB.NET

```
VcGantt1.ResourceScheduler2.OperationWorkInProgressFieldIndex = 15
```

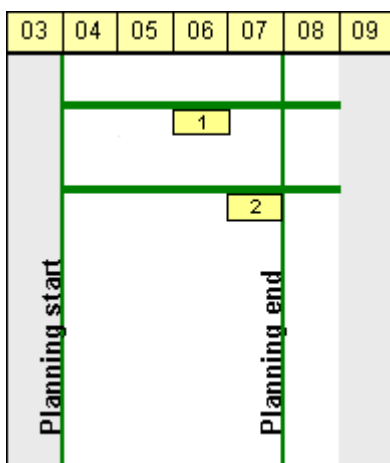
#### Example Code C#

```
vcGantt1.ResourceScheduler2.OperationWorkInProgressFieldIndex = 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	System.DateTime	End date of the scheduling period  <b>Default value:</b> DateTime.MinValue

### Example Code VB.NET

```
VcGantt1.ResourceScheduler2.PlanningEndDate = VcGantt1.TimeScaleEnd
```

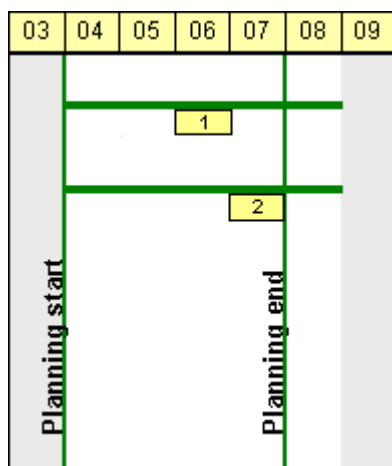
### Example Code C#

```
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	System.DateTime	Start date of the scheduling period <b>Default value:</b> DateTime.MinValue

### Example Code VB.NET

```
VcGantt1.ResourceScheduler2.PlanningStartDate = VcGantt1.TimeScaleStart
```

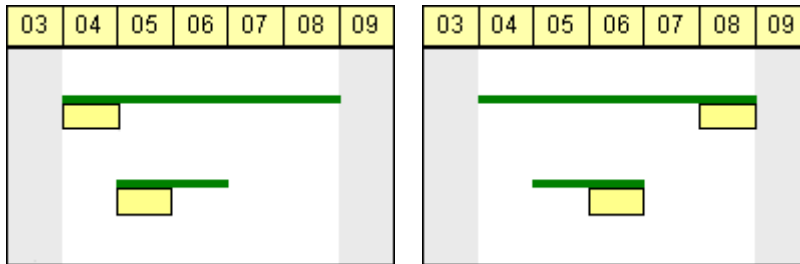
### Example Code C#

```
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 left end of the available period of completion, while JIT tasks tend to appear at its right 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	VcResourceSchedulingPlanningStrategy	Planning strategy <b>Default value:</b> vcResSchedPSASAP
	<b>Possible Values:</b> .vcResSchedPSASAP -1 .vcResSchedPSJIT 0	As soon as possible Just in time

### Example Code VB.NET

```
VcGantt1.ResourceScheduler2.PlanningStrategy =
VcResourceSchedulingPlanningStrategy.vcResSchedPSASAP
```

### Example Code C#

```
vcGantt1.ResourceScheduler2.PlanningStrategy =
VcResourceSchedulingPlanningStrategy.vcResSchedPSASAP;
```

## ResourceCalendarNameFieldIndex

Property of VcResourceScheduler2

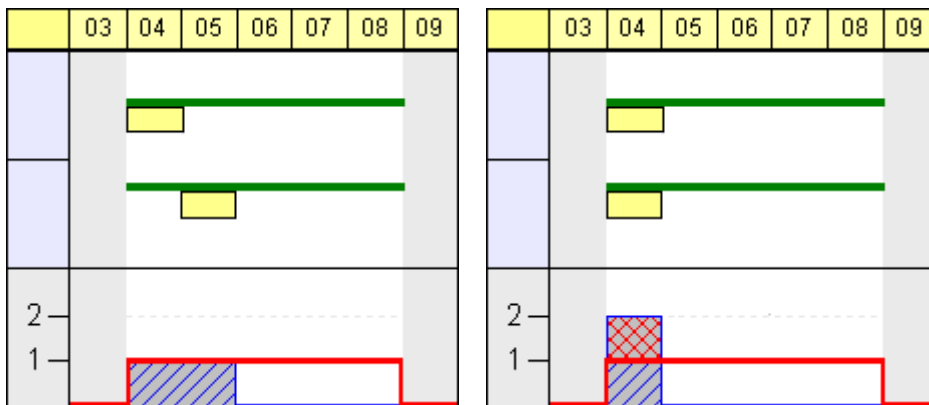
The index passed as the property value specifies a data field in the resource data table that defines the name of a calendar for a resource of the type **TimingResource** or **WorkResource**. If the field of the resource is empty, if it contains an invalid name or if this property is set to -1, as a substitute the name of the resource will be used for the calendar name, as set by the property **ResourceNameFieldIndex**.

	Data Type	Explanation
<b>Property value</b>	System.Int32	Index of the data field in the resource data table that defines a calendar name for the resource of the type <b>TimingResource</b> or <b>WorkResource</b> . <b>Default value:</b> -1

## ResourceCapacityType

Property of VcResourceScheduler2

This property sets 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
<b>Parameter:</b> ⇒ resourceTableIndex	System.Int16	Index of the resource data table. {0...24}
<b>Property value</b>	VcResourceCapacityType	Capacity types <b>Default value:</b> vcResSchedCTFinite

<b>Possible Values:</b>	
.vcResSchedCTFinite -1	Finite capacities
.vcResSchedCTInfinite 0	Infinite capacities

**Example Code VB.NET**

```
VcGantt1.ResourceScheduler2.ResourceCapacityType =
VcResourceSchedulingCapacityType.vcResSchedCTFinite
```

**Example Code C#**

```
vcGantt1.ResourceScheduler2.ResourceCapacityType =
VcResourceSchedulingCapacityType.vcResSchedCTFinite;
```

## ResourceCapacityTypeFieldIndex

Property of VcResourceScheduler2

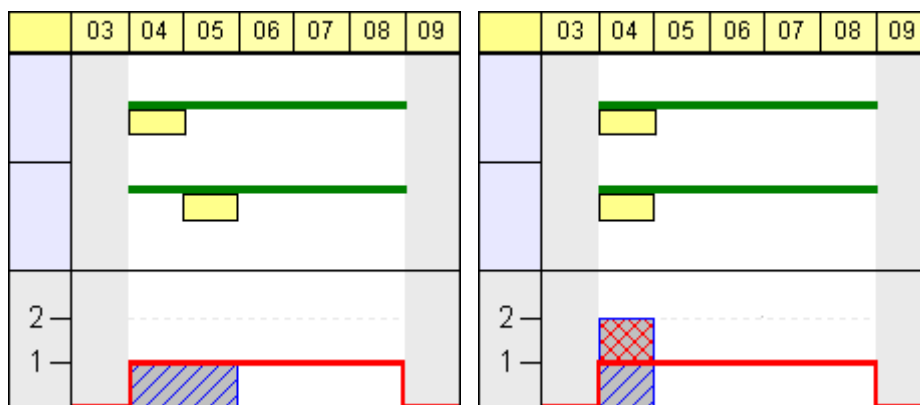
This property lets you set or retrieve the index of a field in a resource data table that holds the capacity type of a single timing resource. In the picture referring to **ResourceDataTableName**, the field index for example is 2.

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.



## 912 API Reference: VcResourceScheduler2

If a resource belongs to more than one group, it has to have the same capacity type in all groups.

	Data Type	Explanation
<b>Parameter:</b> ⇒ resourceTableIndex	System.Int16	Index of the resource data table  {0...24}
<b>Property value</b>	System.Int32	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. <b>Default value:</b> -1

### Example Code VB.NET

```
VcGantt1.ResourceScheduler2.ResourceCapacityTypeFieldIndex(0) = 1
```

### Example Code C#

```
vcGantt1.ResourceScheduler2.set_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 resource 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 truly 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
<b>Parameter:</b> ⇨ index	System.Int16	Index of the resource table  {0...24}
<b>Property value</b>	System.Int32	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.  <b>Default value:</b> -1

**Example Code VB.NET**

```
VcGantt1.ResourceScheduler2.ResourceConstraintTypeFieldIndex(0) = 1
```

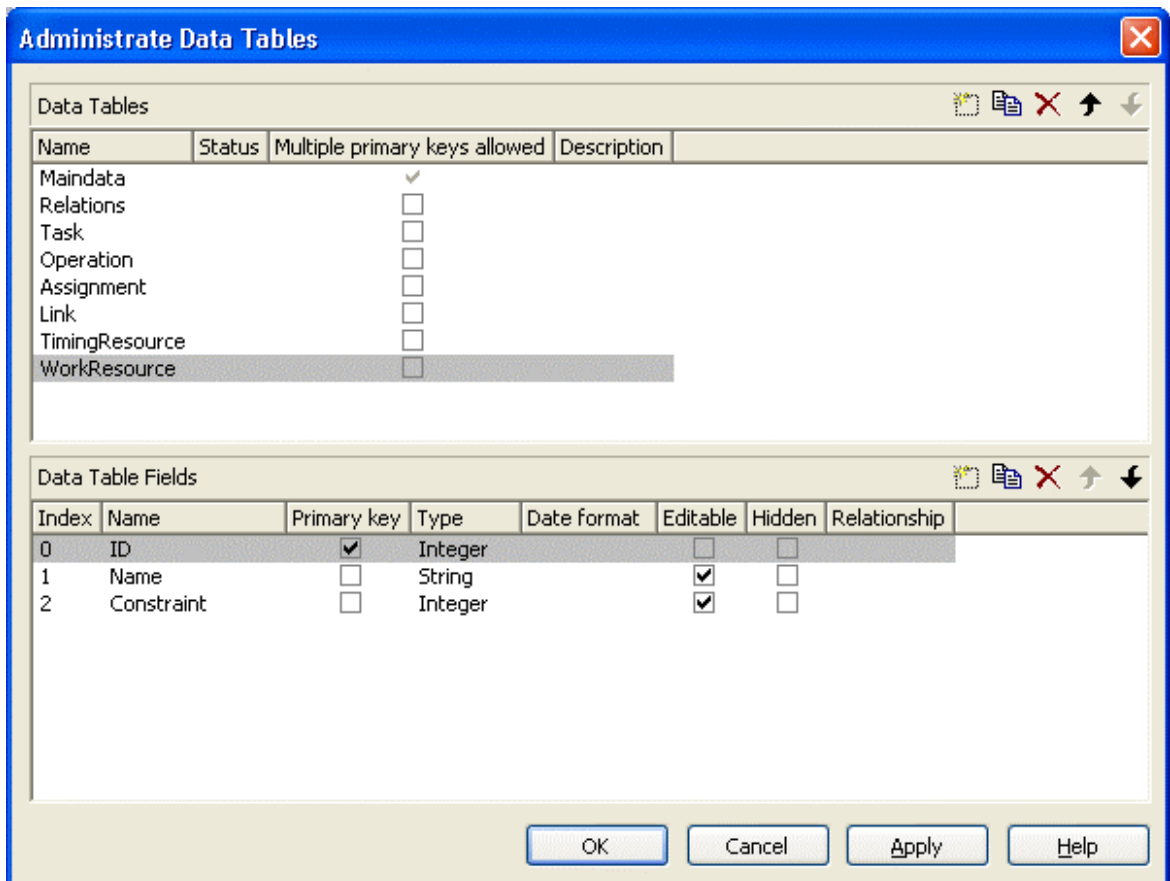
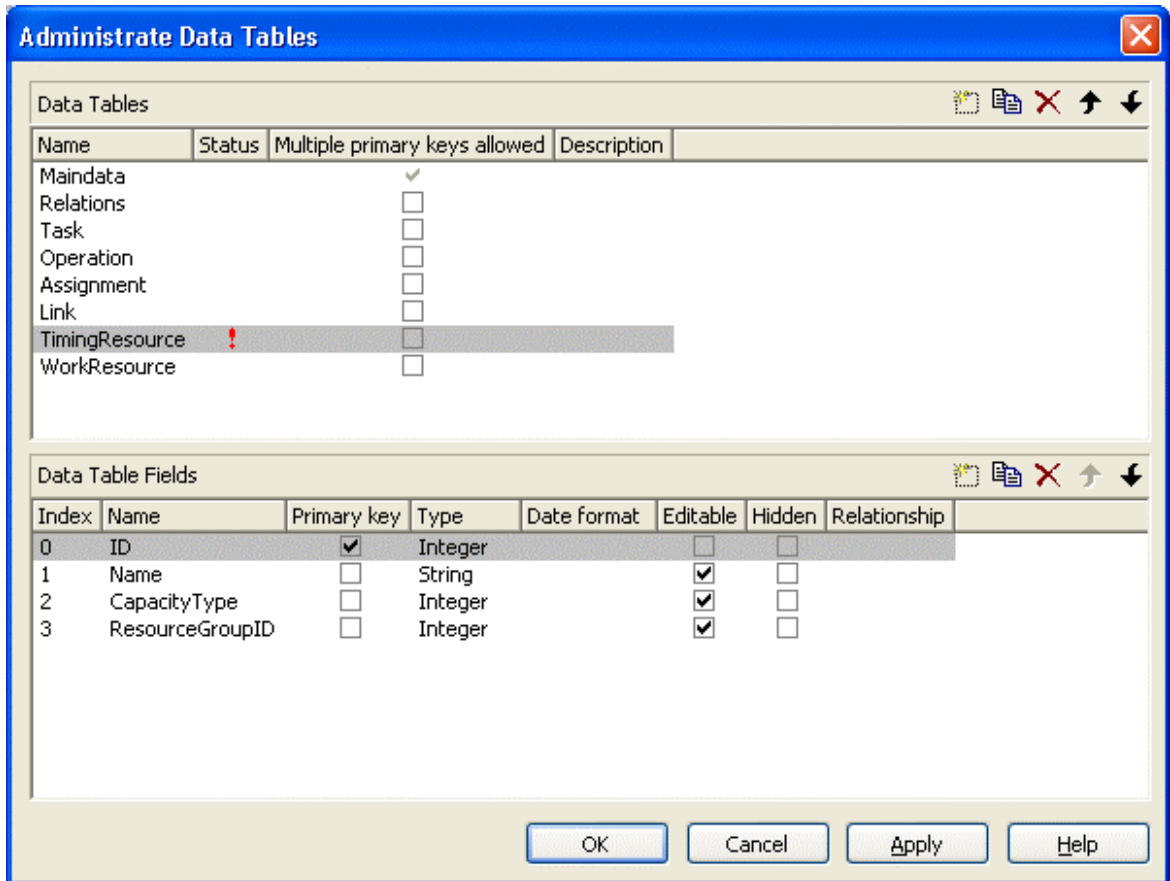
**Example Code C#**

```
vcGantt1.ResourceScheduler2.set_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**.



	Data Type	Explanation
<b>Parameter:</b> ⇨ resourceTableIndex	System.Int16	Index of the resource table  {0...24}
<b>Property value</b>	System.String	Name of the data table  <b>Default value:</b> Empty string

**Example Code VB.NET**

```
VcGantt1.ResourceScheduler2.ResourceDataTableName(1) = "Timing Resource"
```

**Example Code C#**

```
vcGantt1.ResourceScheduler2.set_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 whereas 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.

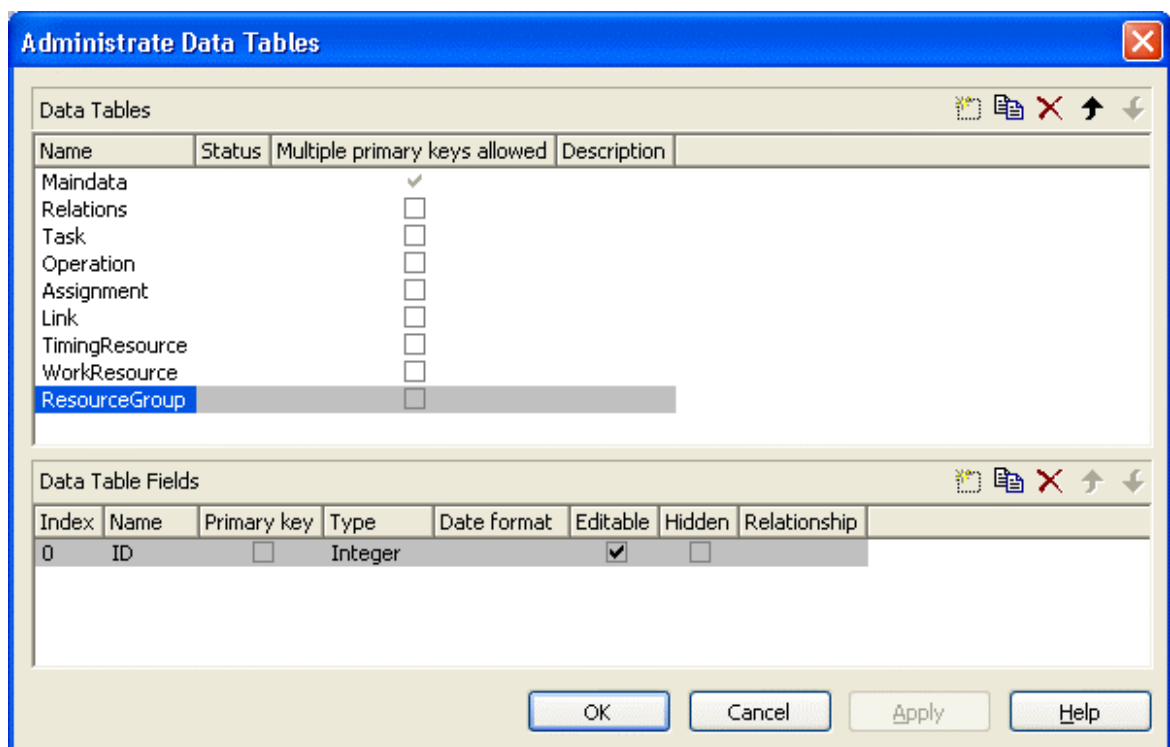
Being a percentage, the values of efficiency in general range between 1 and 100. Values of > 1,000 automatically will be put back to 1,000. The efficiency should NOT be set to a value as high as to reduce the occupation of a resource below 1.

	Data Type	Explanation
<b>Property value</b>	SystemInt.32	Index of the data field in the resource data table that indicates the efficiency in per cent of a resource of the type <b>Timing Resource</b> .

## 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. It 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
<b>Parameter:</b> ⇒ ResourceGroupTableIndex	System.Int16	Index of the resource group data table. {0...24}
<b>Property value</b>	System.String	Name of the resource group data table

### Example Code VB.NET

```
VcGantt1.ResourceScheduler2.ResourceGroupDataTableName(1) = "Printer Resource"
```

### Example Code C#

```
vcGantt1.ResourceScheduler2.set_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
<b>Parameter:</b> ⇨ resourceTableIndex	System.Int16	Index of the resource table.  {0...24}
<b>Property value</b>	System.Int32	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.  <b>Default value:</b> -1

### Example Code VB.NET

```
VcGantt1.ResourceScheduler2.ResourceGroupIDFieldIndex(0) = 1
```

### Example Code C#

```
vcGantt1.ResourceScheduler2.set_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 needs to be specified in different data records 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 **ResourceDataTableName**.

	Data Type	Explanation
<b>Parameter:</b> ⇒ resourceTableIndex	System.Int16	Index the resource table.  {0...24}
<b>Property value</b>	System.Int32	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.  <b>Default value:</b> -1

**Example Code VB.NET**

```
VcGantt1.ResourceScheduler2.ResourceNameFieldIndex(0) = 1
```

**Example Code C#**

```
vcGantt1.ResourceScheduler2.set_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 (see property **ResourceNameFieldIndex**) 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 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
<b>Property value</b>	System.String	Character string that contains the prefix <b>Default value:</b> "Load_"

**Example Code VB.NET**

```
VcGantt1.ResourceScheduler2.ResourceResultLoadCurveNamePrefix = "LoadCurve_"
```

**Example Code C#**

```
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 available 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
<b>Property value</b>	System.String	Character string that contains the prefix <b>Default value:</b> "Stock_"

**Example Code VB.NET**

```
VcGantt1.ResourceScheduler2.ResourceResultStockCurveNamePrefix = 1
```

**Example Code C#**

```
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
<b>Property value</b>	VcResourceSchedulingResourceSelectionStrategy	Selection types
		<b>Default value:</b> VcResSchedRSSequential
	<b>Possible Values:</b>	
	.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 satisfactory.
	.vcResSchedRSHighestEfficiency 2	The resource most efficient when the scheduling is performed will be selected (makes sense only if the property <b>ResourceEfficiency-FieldIndex</b> is used) if its available capacity permits.
	.vcResSchedRSLeastLoaded 0	The resource least loaded when the scheduling is performed will be selected, if its available 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 VB.NET**

```
VcGantt1.ResourceScheduler2.ResourceSelectionStrategy =
VcResourceSchedulingResourceSelectionStrategy.vcResSchedRSLeastLoaded
```

**Example Code C#**

```
vcGantt1.ResourceScheduler2.ResourceSelectionStrategy =
VcResourceSchedulingResourceSelectionStrategy.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 **Resource-ResultStockCurvePrefix**).

	Data Type	Explanation
<b>Parameter:</b> ⇨ resourceTableIndex	System.Int16	Index of the resource data table  {0...24}.
<b>Property value</b>	VcResourceSchedulingResourceType  <b>Possible Values:</b> .vcMaterial 1 .vcTiming -1 .vcWork 0	Type of the resource data table <b>Default value:</b> vcTiming  The resource type is "material". The resource type is "timing". The resource type is "work".

**Example Code VB.NET**

```
VcGantt1.ResourceScheduler2.ResourceType(0) =
VcResourceSchedulingResourceType.vcResSchedTiming
```

**Example Code C#**

```
vcGantt1.ResourceScheduler2.set_ResourceType(0,
VcResourceSchedulingResourceType.vcResSchedTiming);
```

## ResultProcessingStepCount

Property of VcResourceScheduler2

This property provides the number of scheduled tasks in the chart after a scheduling procedure.

	Data Type	Explanation
<b>Property value</b>	System.Int32	Number of tasks  <b>Default value:</b> 0

**Example Code VB.NET**

```
Dim i As Integer
i = VcGantt1.ResourceScheduler2.ResultProcessingStepCount()
```

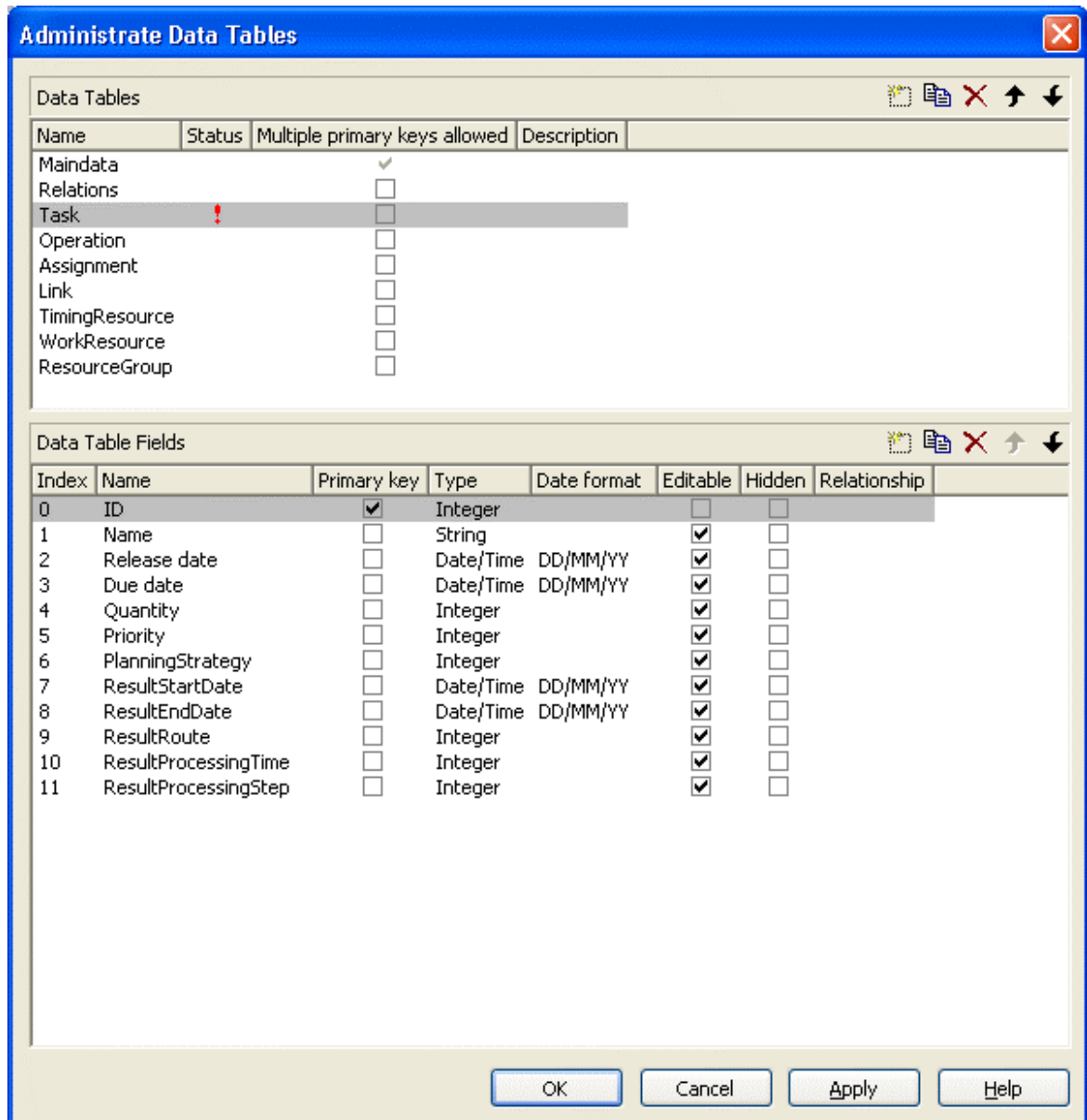
**Example Code C#**

```
int 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	System.String	Name of the task data table <b>Default value:</b> Empty string

### Example Code VB.NET

```
VcGantt1.ResourceScheduler2.TaskDataTableName = "Task"
```

### Example Code C#

```
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 **ToleranceTimeOnASAPDueDates**. 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 **TaskPriorityFieldIndex**).

	Data Type	Explanation
Property value	System.Int32	<p>Index of the data field in the task data table that is designated to hold the due date.</p> <p>{-1...NumberOfFieldsInTaskDataTable -1}. By setting the index to -1, no data field of the task data table will be assigned to this property.</p> <p><b>Default value:</b> -1</p>

### Example Code VB.NET

```
VcGantt1.ResourceScheduler2.TaskDueDateFieldIndex = 3
```

### Example Code C#

```
vcGantt1.ResourceScheduler2.TaskDueDateFieldIndex = 3;
```

## TaskPlanningStrategyFieldIndex

Property of VcResourceScheduler2

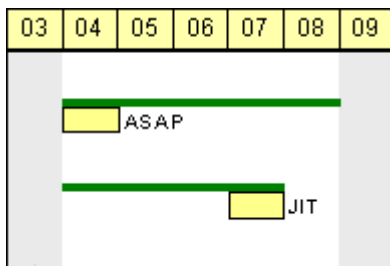
The index specifies a data field which holds an individual planning strategy for a task.

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	System.Int32	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. <b>Default value:</b> -1

#### Example Code VB.NET

```
VcGantt1.ResourceScheduler2.TaskPlanningStrategyFieldIndex = 6
```

#### Example Code C#

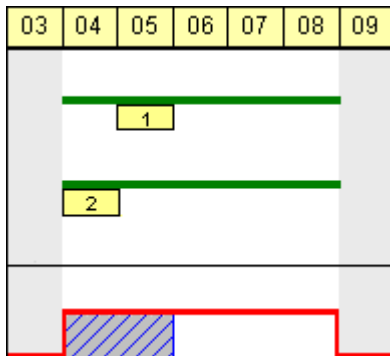
```
vcGantt1.ResourceScheduler2.TaskPlanningStrategyFieldIndex = 6;
```

## 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	System.Int32	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.  <b>Default value:</b> -1

### Example Code VB.NET

```
VcGantt1.ResourceScheduler2.TaskPriorityFieldIndex = 5
```

### Example Code C#

```
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
<b>Property value</b>	System.Int32	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.  <b>Default value:</b> -1

#### Example Code VB.NET

```
VcGantt1.ResourceScheduler2.TaskQuantityFieldIndex = 4
```

#### Example Code C#

```
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	System.Int32	<p>Index of the data field in the task data table that is designated to hold the release date.</p> <p>{-1...NumberOfFieldsInTaskDataTable -1}. By setting the index to -1, no data field of the task data table will be assigned to this property.</p> <p><b>Default value:</b> -1</p>

### Example Code VB.NET

```
VcGantt1.ResourceScheduler2.TaskReleaseDateFieldIndex = 2
```

### Example Code C#

```
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	System.Int32	<p>Index of the data field in the task data table that is designated to hold the end date.</p> <p>{-1...NumberOfFieldsInTaskDataTable -1}. By setting the index to -1, no data field of the task data table will be assigned to this property.</p> <p><b>Default value:</b> -1</p>

### Example Code VB.NET

```
VcGantt1.ResourceScheduler2.TaskResultEndDateFieldIndex = 8
```

### Example Code C#

```
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 an operation. If the post time is 0, the date is identical to the value in the data field which is referred to by the property **TaskResultEndDateFieldIndex**.

	Data Type	Explanation
<b>Property value</b>	System.Int32	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.  <b>Default value:</b> -1

**Example Code VB.NET**

```
VcGantt1.ResourceScheduler2.OperationResultPostEndDateFieldIndex = 15
```

**Example Code C#**

```
vcGantt1.ResourceScheduler2.OperationPostEndDateFieldIndex = 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, the date is identical to the value in the data field which is referred to by the property **TaskResultStartDateFieldIndex**.

	Data Type	Explanation
<b>Property value</b>	System.Int32	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.  <b>Default value:</b> -1

**Example Code VB.NET**

```
VcGantt1.ResourceScheduler2.OperationResultPreparationStartDateFieldIndex = 10
```

**Example Code C#**

```
vcGantt1.ResourceScheduler2.OperationResultPreparationStartDateFieldIndex = 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
<b>Property value</b>	System.Int32	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.  <b>Default value:</b> -1

**Example Code VB.NET**

```
VcGantt1.ResourceScheduler2.TaskResultProcessingStepFieldIndex = 11
```

**Example Code C#**

```
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
<b>Property value</b>	System.Int32	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.  <b>Default value:</b> -1

**Example Code VB.NET**

```
VcGantt1.ResourceScheduler2.TaskResultProcessingTimeFieldIndex = 10
```

**Example Code C#**

```
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	System.Int32	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. <b>Default value:</b> -1

### Example Code VB.NET

```
VcGantt1.ResourceScheduler2.TaskResultRouteFieldIndex = 9
```

### Example Code C#

```
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	System.Int32	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. <b>Default value:</b> -1

### Example Code VB.NET

```
VcGantt1.ResourceScheduler2.TaskResultStartDateFieldIndex = 7
```

### Example Code C#

```
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	System.Int32	Number of base time units {>=0} <b>Default value:</b> 0

### Example Code VB.NET

```
VcGantt1.ResourceScheduler2.ToleranceTimeOnASAPDueDates = 1
```

### Example Code C#

```
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	System.Int32	Number of base time units {>=0} <b>Default value:</b> 0

**Example Code VB.NET**

```
VcGantt1.ResourceScheduler2.ToleranceTimeOnJITReleaseDates = 1
```

**Example Code C#**

```
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	System.Int32	Number of base time units {>=0} <b>Default value:</b> 0

**Example Code VB.NET**

```
VcGantt1.ResourceScheduler2.ToleranceTimeOnStartLockDates = 1
```

**Example Code C#**

```
vcGantt1.ResourceScheduler2.ToleranceTimeOnStartLockDates = 1;
```

## WorkInProgressType

**Property of VcResourceScheduler2**

This property sets the unit to specify the degree of completion (please see **OperationWorkInProgressFieldIndex**).

	Data Type	Explanation
Property value	VcResourceSchedulingWorkInProgressType	Unit of the degree of completion <b>Default value:</b> vcResSchedWIPPercentage
	<b>Possible Values:</b>	

## 934 API Reference: VcResourceScheduler2

.vcResSchedWIPCompleted	0
.vcResSchedWIPPercentage	-1
.vcResSchedWIPRemaining	1

Unit: quantity already completed
Unit: percentage (0...100)
Unit: quantity to be completed

### Example Code VB.NET

```
VcGantt1.ResourceScheduler2.WorkInProgressType =  
VcResourceSchedulingWorkInProgressType.vcResSchedWIPCompleted
```

### Example Code C#

```
vcGantt1.ResourceScheduler2.WorkInProgressType =  
VcResourceSchedulingWorkInProgressType.vcResSchedWIPCompleted;
```

## WritingDebugFilesEnabled

Property of VcResourceScheduler2

If this property is set to **true**, a debug file named **OPS\_debug.txt** will be stored to the current directory, which may be useful for error analysis.

	Data Type	Explanation
Property value	System.Boolean	<b>true</b> : debug files can be written into the current directory. <b>false</b> : debug files cannot be written into the current directory. <b>Default value</b> : false

### Example Code VB.NET

```
VcGantt1.ResourceScheduler2.WritingDebugFilesEnabled = True
```

### Example Code C#

```
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
<b>Parameter:</b> ⇒ taskID	System.String	ID of a task of the corresponding data table which was set by the VcResourceScheduler2 property <b>TaskDataTableName</b> .
<b>Return value</b>	System.String	ID of the first operation of the corresponding data table which was set by the VcResourceScheduler2 property <b>OperationDataTableName</b> .

## 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
<b>Parameter:</b> taskID	System.String	ID of a task of the corresponding data table which was set by the VcResourceScheduler2 property <b>TaskDataTableName</b> .
<b>Return value</b>	System.String	ID of the last operation of the corresponding data table which was set by the VcResourceScheduler2 property <b>OperationDataTableName</b> .

## Process

Method of VcResourceScheduler2

This method starts the scheduling procedure after the desired properties were set. For messages on the progress please also see **ResourceScheduling-Progressing**. **OnResourceSchedulingProgress**. Beside, warnings are put out by **ResourceSchedulingWarning**.



## 936 API Reference: VcResourceScheduler2

	Data Type	Explanation
<b>Return value</b>	System.Boolean	<b>true:</b> No error occurred during the scheduling procedure. <b>false:</b> An error occurred or the scheduling procedure was abandoned.  If the settings allow, error codes may have been stored for each job by the data field addressed by the property <b>OperationResultStatusFieldIndex</b> .

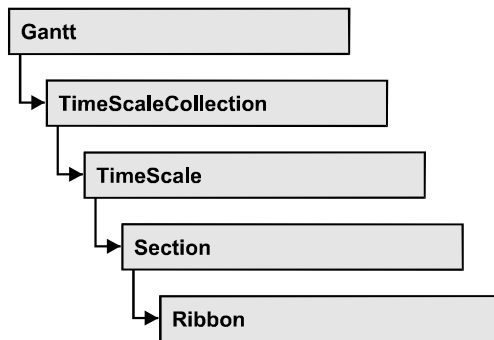
### Example Code VB.NET

```
VcGantt1.ResourceScheduler2.Process()
```

### Example Code C#

```
vcGantt1.ResourceScheduler2.Process();
```

## 6.61 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

- BackgroundColor
- CalendarName
- DateOutputFormat
- Font
- FontColor
- MajorTicks
- MinorTicks
- Pattern
- Position
- ReferenceDate
- TextAlignment
- TickColor
- TickPosition
- Type
- UnitSeparation
- UseReferenceDate

## Properties

### BackgroundColor

Property of VcRibbon

This property lets you set or retrieve the background color of the ribbon.

	Data Type	Explanation
Property value	System.Drawing.Color	RGB color values {0...255},{0...255},{0...255}

#### Example Code VB.NET

```
Dim timeScale As VcTimeScale
Dim ribbon As VcRibbon

timeScale = VcGantt1.TimeScaleCollection.Active
ribbon = timeScale.Ribbon(0, 0)
ribbon.BackgroundColor = Color.Blue
```

#### Example Code C#

```
VcTimeScale timeScale = vcGantt1.TimeScaleCollection.Active;
VcRibbon ribbon = timeScale.get_Ribbon(0,0);
ribbon.BackgroundColor = Color.LightSteelBlue;
```

### CalendarName

Property of VcRibbon

This property lets you set or retrieve the calendar name.

	Data Type	Explanation
Property value	System.String	Calendar name

### 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 **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
- 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	System.String	Date format {DMYhms:;}

**Example Code VB.NET**

```
Dim timeScale As VcTimeScale
Dim ribbon As VcRibbon

timeScale = VcGantt1.TimeScaleCollection.Active
ribbon = timeScale.Ribbon(0, 0)
ribbon.DateOutputFormat = "DD.MMM.YYYY hh:mm:ss"
```

**Example Code C#**

```
VcTimeScale timeScale = vcGantt1.TimeScaleCollection.Active;
VcRibbon ribbon = timeScale.get_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	Font	Font attributes of the ribbon

**Example Code VB.NET**

```
Dim timeScale As VcTimeScale
Dim ribbon As VcRibbon
Dim newFont As Font

timeScale = VcGantt1.TimeScaleCollection.Active
ribbon = timeScale.Ribbon(0, 0)
newFont = New Font("Times New Roman", 14, FontStyle.Italic)
ribbon.Font = newFont
```

**Example Code C#**

```
VcTimeScale timeScale = vcGantt1.TimeScaleCollection.Active;
VcRibbon ribbon = timeScale.get_Ribbon(0,0);
Font newFont = new Font("Times New Roman", 14, FontStyle.Italic);
ribbon.Font = newFont;
```

## FontColor

**Property of VcRibbon**

This property lets you set or retrieve the font color of the ribbon.

	Data Type	Explanation
<b>Property value</b>	System.Drawing.Color	RGB color values {0...255},{0...255},{0...255}

**Example Code VB.NET**

```
Dim timeScale As VcTimeScale
Dim ribbon As VcRibbon

timeScale = VcGantt1.TimeScaleCollection.Active
ribbon = timeScale.Ribbon(0, 0)
ribbon.FontColor = Color.Blue
```

**Example Code C#**

```
VcTimeScale timeScale = vcGantt1.TimeScaleCollection.Active;
VcRibbon ribbon = timeScale.get_Ribbon(0,0);
ribbon.FontColor = Color.LightSteelBlue;
```

## 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	System.Int16	Number of units between two major ticks

### Example Code VB.NET

```
Dim timeScale As VcTimeScale
Dim ribbon As VcRibbon

timeScale = VcGantt1.TimeScaleCollection.Active
ribbon = timeScale.Ribbon(0, 0)
ribbon.MajorTicks = 7
```

### Example Code C#

```
VcTimeScale timeScale = vcGantt1.TimeScaleCollection.Active;
VcRibbon ribbon = timeScale.get_Ribbon(0,0);
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	System.Int16	Number of units between two minor ticks

### Example Code VB.NET

```
Dim timeScale As VcTimeScale
Dim ribbon As VcRibbon

timeScale = VcGantt1.TimeScaleCollection.Active
ribbon = timeScale.Ribbon(0, 0)
ribbon.MinorTicks = 1
```






### Example Code C#

```
VcTimeScale timeScale = vcGantt1.TimeScaleCollection.Active;
VcRibbon ribbon = timeScale.get_Ribbon(0,0);
ribbon.MinorTicks = 1;
```

## Pattern

### Property of VcRibbon

This property lets you set or retrieve the pattern of the ribbon background.

	Data Type	Explanation
<b>Property value</b>	VcFillPatternSingleColored	Pattern type
	<b>Possible Values:</b> .vcAeroGlassPattern 44	Vertical color gradient in the color of the fill pattern 
	.vcSingleColoredNoPattern 1276	No fill pattern
	.vcSingleColoredVerticalBottomLightedConvexPattern 43	Vertical color gradient from bright to dark 
	.vcSingleColoredVerticalConcavePattern 40	Vertical color gradient from dark to bright to dark 
	.vcSingleColoredVerticalConvexPattern 41	Vertical color gradient from bright to dark to bright 
.vcSingleColoredVerticalTopLightedConvexPattern 42	Vertical color gradient from dark to bright 	

## Position

**Property of VcRibbon**

This property lets you set or retrieve the position of the ribbon.

	Data Type	Explanation
<b>Property value</b>	VcRibbonPosition	Ribbon position
	<b>Possible Values:</b> .vcRPBottom 2 .vcRPNone 0 .vcRPTop 1	bottom none top

## ReferenceDate

**Property of VcRibbon**

This property lets you set or retrieve the reference date.



	Data Type	Explanation
Property value	System.DateTime	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	VcHorizontalRibbonTextAlignment	Positioned above the tick, centered between two ticks, left aligned, right aligned
	<b>Possible Values:</b> .vcRTAtTickAligned 1039 .vcRTHorCenterAligned -1 .vcRTLeftAligned -3 .vcRTRightAligned -2	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 VB.NET

```
Dim timeScale As VcTimeScale
Dim ribbon As VcRibbon

timeScale = VcGantt1.TimeScaleCollection.Active
ribbon = timeScale.Ribbon(0, 0)
ribbon.TextAlignment = VcHorizontalRibbonTextAlignment.vcRTLeftAligned
```

### Example Code C#

```
VcTimeScale timeScale = vcGantt1.TimeScaleCollection.Active;
VcRibbon ribbon = timeScale.get_Ribbon(0,0);
ribbon.TextAlignment = VcHorizontalRibbonTextAlignment.vcRTLeftAligned;
```

## TickColor

Property of VcRibbon

This property lets you set or retrieve the color of ticks.

	Data Type	Explanation
Property value	System.Drawing.Color RGB ({0...255},{0...255},{0...255})	RGB color values ({0...255},{0...255},{0...255}) <b>Default value:</b> 0,0,0

## TickPosition

Property of VcRibbon

This property lets you set or retrieve the tick position.

Property value	Data Type	Explanation
	VcRibbonTickPosition	Tick position
	<b>Possible Values:</b> .vcTPAbove 1044 .vcTPBelow 1045	above below

## Type

Property of VcRibbon

This property lets you set or retrieve the ribbon type. The types available are listed below.

Property value	Data Type	Explanation
	VcRibbonType	Ribbon type
	<b>Possible Values:</b> .vcDayRibbon 5 .vcHourRibbon 6 .vcMinuteRibbon 7 .vcMonthRibbon 3 .vcQuarterRibbon 10 .vcSecondRibbon 9 .vcShiftRibbon 8 .vcWeekRibbon 4 .vcYearRibbon 1	Ribbon showing days' 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 VB.NET

```
Dim timeScale As VcTimeScale
Dim ribbon As VcRibbon

timeScale = VcGantt1.TimeScaleCollection.Active
ribbon = timeScale.Ribbon(0, 0)
ribbon.Type = VcRibbonType.vcWeekRibbon
```

### Example Code C#

```
VcTimeScale timeScale = vcGantt1.TimeScaleCollection.Active;
VcRibbon ribbon = timeScale.get_Ribbon(0,0);
ribbon.Type = VcRibbonType.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
<b>Property value</b>	VcUnitSeparation  <b>Possible Values:</b> .vcUSFullLine 4 .vcUSNone 1 .vcUSTick 1035	Appearance of the major tick  Units separated by full lines Units not separated Units separated by ticks

### Example Code VB.NET

```
Dim timeScale As VcTimeScale
Dim ribbon As VcRibbon

timeScale = VcGantt1.TimeScaleCollection.Active
ribbon = timeScale.Ribbon(0, 0)
ribbon.UnitSeparation = VcUnitSeparation.vcUSTick
```

### Example Code C#

```
VcTimeScale timeScale = vcGantt1.TimeScaleCollection.Active;
VcRibbon ribbon = timeScale.get_Ribbon(0,0);
ribbon.UnitSeparation = VcUnitSeparation.vcUSTick;
```

## UseReferenceDate

Property of VcRibbon

This property lets you set or retrieve whether the ribbon uses a reference date.

	Data Type	Explanation
<b>Property value</b>	System.Boolean	The ribbon uses (True) / does not use (False) reference date

## 6.62 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 actual end date of the activity. This is only possible as long as no data has been loaded.

	Data Type	Explanation
Property value	System.Int32	Index of the data field which holds the actual end date

### ActualStartDateDataFieldIndex

Property of VcScheduler

With this property you can set/retrieve the index of the data field which contains the actual start date of the activity. This is only possible as long as no data has been loaded.

	Data Type	Explanation
Property value	System.Int32	Index of the data field which holds the currently 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	System.Boolean	Automatic time scheduling is switched on (true) or off (false) <b>Default value:</b> false

## 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	SystemInt.32	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 earliest possible end date of the activity. This is only possible as long as no data has been loaded.

	Data Type	Explanation
Property value	SystemInt.32	Index of the data field which holds the earliest possible end date of an activity

## EarlyStartDateDataFieldIndex

Property of VcScheduler

With this property you can set/retrieve the index of the data field which contains the earliest possible start date of the activity. This is only possible as long as no data has been loaded.

	Data Type	Explanation
Property value	SystemInt.32	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	System.DateTime	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	System.Int32	Index of the data field which holds the desired late 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	System.Int32	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 project. This is only possible as long as no data has been loaded.

	Data Type	Explanation
Property value	System.Int32	Index of the data field which holds the latest possible end date of an activity

## 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 project.activity. This is only possible as long as no data has been loaded.

	Data Type	Explanation
Property value	SystemInt.32	Index of the data field which holds the latest possible start date of an activity

## 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	SystemInt.32	Index of the data field which holds the minimum time space between a predecessor and a successor

## ScheduledProjectEndDate

Read Only Property of VcScheduler

This property returns the data **Early end** of a project after having calculated the project dates by **VcScheduler.ScheduleProject** if the end date was set before.

This property can also be set on the **General** property page.

	Data Type	Explanation
Property value	SystemInt.32	Index of the data field which holds the calculated end date of the project



## ScheduledProjectStartDate

**Read Only Property of VcScheduler**

This property returns the **Late start** 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
<b>Property value</b>	System.Int32	Index of the data field which holds the calculated 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
<b>Property value</b>	System.Boolean	Scheduling of nodes only with predecessors is switched on/off

## 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
<b>Property value</b>	System.DateTime	Desired start date for automatic scheduling

## StartDateNotEarlierThanDataFieldIndex

**Property of VcScheduler**

With this property you can set/retrieve the index of the data field which contains the desired earliest start date of the activity.

	Data Type	Explanation
Property value	System.Int.32	Index of the data field which holds the desired early start date

## TotalFloatDataFieldIndex

Property of VcScheduler

With this property you can set/retrieve the index of the data field which contains the calculated total float of the activity.

	Data Type	Explanation
Property value	System.Int.32	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 **EarlyStartDateDataFieldIndex**, **LateStartDateDataFieldIndex**, **EarlyEndDateDataFieldIndex**, **LateEndDateDataFieldIndex**, **FreeFloatDataFieldIndex** and **TotalFloatDataFieldIndex**.

	Data Type	Explanation
<b>Parameter:</b>		
⇒ startDate	System.DateTime	Desired start date
⇒ endDate	System.DateTime	Desired end date

## 954 API Reference: VcScheduler

---

<b>Return value</b>	System.Boolean	The project data were successfully calculated (true) / were not calculated (False)
---------------------	----------------	--

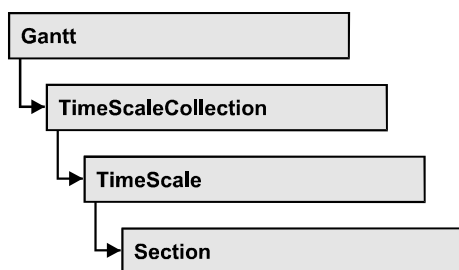
### Example Code VB.NET

```
VcScheduler.ScheduleProject(2.5.2012, 1.10.2012)
```

### Example Code C#

```
vcScheduler.ScheduleProject(2.5.2012, 1.10.2012)
```

## 6.63 VcSection



An object of the type VcSection represents a section of the time scale.

### Properties

- CalendarGrid
- DateLineGrid
- NonWorkIntervalsCollapsed
- Ribbon
- StartDate
- TimeUnit
- UnitWidth

---

## Properties

### CalendarGrid

**Read Only Property of VcSection**

This property lets you retrieve one of the calendar grids used in the section.

The property is an Indexed Property, which in C# is addressed by the method `get_CalendarGrid (gridIndex)`.

	Data Type	Explanation
<b>Parameter:</b> ⇒ gridIndex	System.Int16	Index of the calendar grid
<b>Property value</b>	VcCalendarGrid	CalendarGrid object

## 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.

The property DateLineGrid is an Indexed Property, which in C# is addressed by the method `get_DateLineGrid (gridIndex)`.

	Data Type	Explanation
<b>Parameter:</b> ⇒ gridIndex	System.Int16	Index of the date line grid
<b>Property value</b>	VcDateLineGrid	DateLine object

### Example Code VB.NET

```
Dim timescale As VcTimeScale
Dim section As VcSection
Dim dateLineGrid As VcDateLineGrid

timescale = VcGantt1.TimeScaleCollection.Active
section = timescale.Section(0)
dateLineGrid = section.DateLineGrid(0)
```

### Example Code C#

```
VcTimeScale timeScale = vcGantt1.TimeScaleCollection.Active;
VcSection section = timeScale.get_Section(0);
VcDateLineGrid dateLineGrid = section.get_DateLineGrid(0);
```

## NonWorkIntervalsCollapsed

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**.

	Data Type	Explanation
<b>Property value</b>	System.Boolean	Workfree periods are/are not collapsed

**Example Code VB.NET**

```
Dim timeScaleCltn As VcTimeScaleCollection
Dim timeScale As VcTimeScale
Dim section As VcSection

timeScaleCltn = VcGantt1.TimeScaleCollection
timeScale = timeScaleCltn.Active
section = timeScale.Section(1)
section.NonWorkIntervalsCollapsed = True
```

**Example Code C#**

```
VcTimeScaleCollection timeScaleCltn = vcGantt1.TimeScaleCollection;
VcTimeScale timeScale = timeScaleCltn.Active;
VcSection section = timeScale.get_Section(1);
section.NonWorkIntervalsCollapsed = true;
```

## Ribbon

**Property of VcSection**

This property lets you access the ribbons of a section.

The property Ribbon is an Indexed Property, which in C# is addressed by the methods `sset_Ribbon (ribbonIndex, pvn)` and `get_Ribbon (ribbonIndex)`.

	Data Type	Explanation
<b>Parameter:</b> ⇒ ribbonIndex	System.Int16	Index of the ribbon
<b>Property value</b>	VcRibbon	Ribbon object

**Example Code VB.NET**

```
Dim timescale As VcTimeScale
Dim section As VcSection
Dim ribbon As VcRibbon

timescale = VcGantt1.TimeScaleCollection.Active
section = timescale.Section(0)
ribbon = section.Ribbon(0)
```

**Example Code C#**

```
VcTimeScale timeScale = vcGantt1.TimeScaleCollection.Active;
VcSection section = timeScale.get_Section(0);
VcRibbon ribbon = section.get_Ribbon(0);
```

## StartDate

**Property of VcSection**

This property lets you set or retrieve the start date of a time scale section. The start date of the first section (Section 0) is automatically set by the

project start. It cannot be set here, but can merely be retrieved. Besides, a start date beyond the time scale must not be set.

	Data Type	Explanation
Property value	System.DateTime	Start date of the time scale section

**Example Code VB.NET**

```
Dim timescale As VcTimeScale
Dim section As VcSection

timescale = VcGantt1.TimeScaleCollection.Active
section = timescale.Section(0)
section.StartDate = "21.06.14"
```

**Example Code C#**

```
VcTimeScale timeScale = vcGantt1.TimeScaleCollection.Active;
VcSection section = timeScale.get_Section(0);
section.StartDate = Convert.ToDateTime("21.06.14");
```

## TimeUnit

Property of VcSection

This property lets you set or retrieve the time unit that a section is based on.

	Data Type	Explanation
Property value	VcTimeUnit  <b>Possible Values:</b> .vcDay 5 .vcHour 6 .vcMinute 7 .vcSecond 8	Time unit of the section  Time unit <b>day</b> Time unit <b>hour</b> Time unit <b>minute</b> Time unit <b>second</b>

**Example Code VB.NET**

```
Dim timescale As VcTimeScale
Dim section As VcSection

timescale = VcGantt1.TimeScaleCollection.Active
section = timescale.Section(0)
section.TimeUnit = VcTimeUnit.vcHour
```

**Example Code C#**

```
VcTimeScale timeScale = vcGantt1.TimeScaleCollection.Active;
VcSection section = timeScale.get_Section(0);
section.TimeUnit = VcTimeUnit.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	System.Int32	unit width (1/100 mm)

### Example Code VB.NET

```
Dim timescale As VcTimeScale
Dim section As VcSection

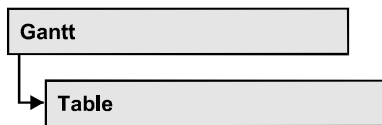
timescale = VcGantt1.TimeScaleCollection.Active
section = timescale.Section(0)
section.UnitWidth = 660
```

### Example Code C#

```
VcTimeScale timeScale = vcGantt1.TimeScaleCollection.Active;
VcSection section = timeScale.get_Section(0);
section.UnitWidth = 660;
```



## 6.64 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
- Position
- TableFormatCollection
- Visible

### Methods

- 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.

The property ColumnTitle is an Indexed Property, which in C# is addressed by the methods set\_ColumnTitle (colNumber, pvn) and get\_ColumnTitle (colNumber).

**Note:** The index starts at 1.

	Data Type	Explanation
<b>Parameter:</b> ⇒ colNumber	System.Int16	Number of table column
<b>Property value</b>	System.String	Column title

**Example Code VB.NET**

```
Dim table As VcTable

table = VcGantt1.LeftTable
table.ColumnTitle(2) = "ID"
```

**Example Code C#**

```
VcTable table = vcGantt1.LeftTable;
table.set_ColumnTitle(2, "ID");
```

## ColumnWidth

**Property of VcTable**

This property lets you specify the width of each table column (in units of 1/100 mm). This property also can be set in the **Edit Table** dialog.

The property ColumnWidth is an Indexed Property, which in C# is addressed by the methods set\_ColumnWidth (colNumber, pvn) and get\_ColumnWidth (colNumber).

	Data Type	Explanation
<b>Parameter:</b> ⇒ colNumber	System.Int16	Number of table column
<b>Property value</b>	System.Int32	Column width

**Example Code VB.NET**

```
Dim table As VcTable

table = VcGantt1.LeftTable
table.ColumnWidth(1) = 1500
```

**Example Code C#**

```
VcTable table = vcGantt1.LeftTable;
table.set_ColumnWidth(1, 1500);
```

## 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
<b>Property value</b>	VcTablePosition  <b>Possible Values:</b> .vcLeftTable 0	Position of the table  Table on the left of the diagram

	.vcRightTable 1	Table on the right of the diagram
--	-----------------	-----------------------------------

### Example Code VB.NET

```
Dim table As VcTable

table = VcGantt1.LeftTable
MsgBox(table.Position)
```

### Example Code C#

```
VcTable table = vcGantt1.LeftTable;
MessageBox.Show(table.Position.ToString());
```

## TableFormatCollection

**Read Only Property of VcTable**

This property lets you access the TableFormatCollection object that contains all table formats available.

	Data Type	Explanation
<b>Property value</b>	VcTableFormatCollection	TableFormatCollection object

### Example Code VB.NET

```
Dim table As VcTable
Dim formatCltn As VcTableFormatCollection

table = VcGantt1.LeftTable
formatCltn = table.TableFormatCollection
```

### Example Code C#

```
VcTable table = vcGantt1.LeftTable;
VcTableFormatCollection formatCltn = table.TableFormatCollection;
```

## Visible

**Property of VcTable**

This property lets you set or retrieve whether the table is visible or not.

	Data Type	Explanation
<b>Property value</b>	System.Boolean	Table visible/invisible

### Example Code VB.NET

```
Dim table As VcTable

table = VcGantt1.LeftTable
table.Visible = True
```

**Example Code C#**

```
VcTable table = vcGantt1.LeftTable;  
table.Visible = true;
```

---

## Methods

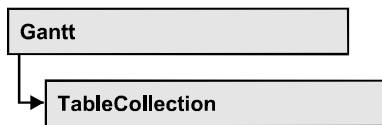
### OptimizeColumnWidth

**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
<b>Parameter:</b> ⇒ columnNo	System.Int16	Column number
<b>Return value</b>	Void	

## 6.65 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 TableCollection** 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.

### Methods

- `TableByIndex`

---

## Methods

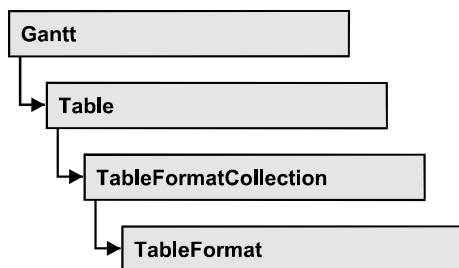
### TableByIndex

**Method of VcTableCollection**

This method lets you access a table by its index. If a table does not exist at the index specified, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
<b>Parameter:</b>		
⇒ index	System.Int16	Index of the table
<b>Return value</b>	VcTable	Table object returned

## 6.66 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

- `CollapseColumn`
- `FieldsSeparatedByLines`
- `FilterName`
- `FormatField`
- `FormatFieldCount`
- `IndentColumn`
- `IndentWidth`
- `Name`

- SeparationLineColor
- ThreeDEffect

## Methods

- GetEnumerator

## Properties

### 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	System.Int16	Display of +/- in column switched on

#### Example Code VB.NET

```
' Display of +/- in the fifth column
VcGantt1.TableCollection.Active.TableFormatCollection.FormatByName("Hierarchy").
CollapseColumn = 5
VcGantt1.TableCollection.Active.TableFormatCollection.FormatByName("HierarchyCol
lapsed").CollapseColumn = 5
```

#### Example Code C#

```
// Display of +/- in the fifth column
vcGantt1.TableCollection.Active.TableFormatCollection.FormatByName("Hierarchy").
CollapseColumn = 5;
vcGantt1.TableCollection.Active.TableFormatCollection.FormatByName("HierarchyCol
lapsed").CollapseColumn = 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	System.Boolean	Table fields are separated by lines (True)/ are not separated by lines (False).

**Example Code VB.NET**

```
Dim format As VcTableFormat

format = VcGantt1.LeftTable.TableFormatCollection.FormatByName("StandardList")
format.FieldsSeparatedByLines = True
```

**Example Code C#**

```
VcTableFormat format =
vcGantt1.LeftTable.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	System.String	Name of the filter

**Example Code VB.NET**

```
Dim format As VcTableFormat

format = VcGantt1.LeftTable.TableFormatCollection.FormatByName("ListFormat2")
format.FilterName = "Code1NotA"
```

**Example Code C#**

```
VcTableFormat format =
vcGantt1.LeftTable.TableFormatCollection.FormatByName("ListFormat2")
format.FilterName = "Code1NotA";
```

## FormatField

**Read Only Property of VcTableFormat**

This property gives access to a VcTableFormatField object by its 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.

The property FormatField is an Indexed Property, which in C# is addressed by the method get\_FormatField (index).



	Data Type	Explanation
<b>Parameter:</b> index	System.Int16	Index of the table format field 0 ... FormatFieldCount-1
<b>Property value</b>	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
<b>Property value</b>	System.Int16	Number of table columns

### Example Code VB.NET

```
Dim format As VcTableFormat
Dim numberOfColumns As Integer

format = VcGantt1.LeftTable.TableFormatCollection.FormatByName("StandardList")
numberOfColumns = format.FormatFieldCount
```

### Example Code C#

```
VcTableFormat format =
vcGantt1.LeftTable.TableFormatCollection.FormatByName("StandardList");
int numberOfColumns = format.FormatFieldCount;
```

## IndentColumn

**Property of VcTableFormat**

This property lets you specify the number of the column which shall be indented.

	Data Type	Explanation
<b>Property value</b>	System.Int16	Number of indented column

### Example Code VB.NET

```
' Second column is indented
VcGantt1.TableCollection.Active.TableFormatCollection.FormatByName("StandardList")
.IndentColumn = 2
```

**Example Code C#**

```
// Second column is indented
vcGantt1.TableCollection.Active.TableFormatCollection.FormatByName("StandardList").IndentColumn = 2;
```

**IndentWidth**

Property of VcTableFormat

Specify the measure by which the column shall be indented in mm

	Data Type	Explanation
Property value	System.Int32	Measure of indentation

**Example Code VB.NET**

```
' Second column is indented by 100 mm
VcGantt1.TableCollection.Active.TableFormatCollection.FormatByName("StandardList").IndentColumn = 2
VcGantt1.TableCollection.Active.TableFormatCollection.FormatByName("StandardList").IndentWidth = 100
```

**Example Code C#**

```
// Second column is indented by 100 mm
vcGantt1.TableCollection.Active.TableFormatCollection.FormatByName("StandardList").IndentColumn = 2;
vcGantt1.TableCollection.Active.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	System.String	Table format name

**Example Code VB.NET**

```
Dim format As VcTableFormat
Dim formatName As String

format = VcGantt1.LeftTable.TableFormatCollection.FirstFormat
formatName = format.Name
```

**Example Code C#**

```
VcTableFormat format = vcGantt1.LeftTable.TableFormatCollection.FirstFormat();
string 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} <b>Default value:</b> RGB(0,0,0)

### Example Code VB.NET

```
VcTableFormat.SeparationLineColor = RGB(255, 204, 204)
```

### Example Code C#

```
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	System.Boolean	3D effect switched on (True)/switched off (False)

### Example Code VB.NET

```
Dim format As VcTableFormat
```

```
format = VcGantt1.LeftTable.TableFormatCollection.FormatByName("StandardList")
format.ThreeDEffect = True
```

### Example Code C#

```
VcTableFormat format =
vcGantt1.LeftTable.TableFormatCollection.FormatByName("StandardList");
format.ThreeDEffect = true;
```

## Methods

### GetEnumerator

Method of VcTableFormat

This method returns an Enumerator object which supports the iteration by language specific elements. It is implied in the For...Each construct of Visual Basic and C#. This object allows to iterate over the table format fields included.

	Data Type	Explanation
Return value	VcObject	Reference object

#### Example Code VB.NET

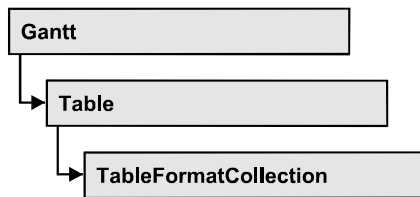
```
Dim format As VcTableFormat
Dim formatField As VcTableFormatField

For Each formatField In format
    Debug.Write(formatField.Index)
Next
```

#### Example Code C#

```
private void Form1_Load(object sender, System.EventArgs e)
{
    VcTableFormat format;
    foreach (VcTableFormatField formatField in format)
        Console.Writ(formatField.Index);
}
```

## 6.67 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

- `Count`

### Methods

- `FirstFormat`
- `FormatByIndex`
- `FormatByName`
- `GetEnumerator`
- `NextFormat`

---

## Properties

### 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	System.Int32	Number of table formats

**Example Code VB.NET**

```
Dim formatCltn As VcTableFormatCollection
Dim numberOfFormats As Integer

formatCltn = VcGantt1.LeftTable.TableFormatCollection
numberOfFormats = formatCltn.Count
```

**Example Code C#**

```
VcTableFormatCollection formatCltn = vcGantt1.LeftTable.TableFormatCollection;
int 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
<b>Return value</b>	VcTableFormat	First table format

**Example Code VB.NET**

```
Dim format As VcTableFormat

format = VcGantt1.LeftTable.TableFormatCollection.FirstFormat
```

**Example Code C#**

```
VcTableFormat format = vcGantt1.LeftTable.TableFormatCollection.FirstFormat();
```

### FormatByIndex

**Method of VcTableFormatCollection**

This method lets you access a table format by its index. If a table format does not exist at the index specified, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
<b>Parameter:</b> ⇒ index	System.Int16	Index of the table format

<b>Return value</b>	VcTableFormat	Table format 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
<b>Parameter:</b> ⇒ formatName	System.String	Name of the table format
<b>Return value</b>	VcTableFormat	Table format

### Example Code VB.NET

```
Dim format As VcTableFormat

format = vcGantt1.LeftTable.TableFormatCollection.FormatByName("StandardList")
```

### Example Code C#

```
VcTableFormat format =
vcGantt1.LeftTable.TableFormatCollection.FormatByName("StandardList");
```

## GetEnumerator

### Method of VcTableFormatCollection

This method returns an Enumerator object which supports the iteration by language specific elements. It is implied in the For...Each construct of Visual Basic and C#. This object allows to iterate over the table formats included.

	Data Type	Explanation
<b>Return value</b>	VcObject	Reference object

### Example Code VB.NET

```
Dim format As VcTableFormat

For Each format In vcGantt1.LeftTable.TableFormatCollection
    Debug.Write(format.Name)
Next
```

### Example Code C#

```
foreach (VcTableFormat format in vcGantt1.LeftTable.TableFormatCollection)
    Console.Write(format.Name);
```

## 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 VB.NET

```
Dim formatCltn As VcTableFormatCollection
Dim format As VcTableFormat

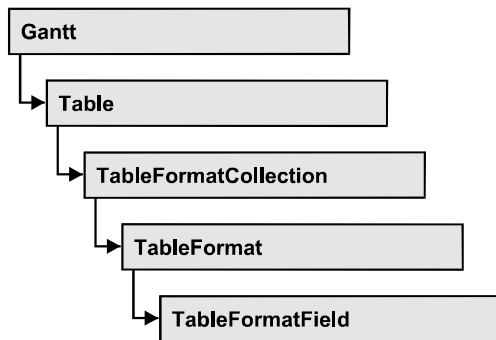
formatCltn = VcGantt1.LeftTable.TableFormatCollection
format = formatCltn.FirstFormat
While Not format Is Nothing
    ListBox1.Items.Add(format.Name)
    format = formatCltn.NextFormat
End While
```

### Example Code C#

```
VcTableFormatCollection formatCltn = vcGantt1.LeftTable.TableFormatCollection;
VcTableFormat format = formatCltn.FirstFormat();
while (format != null)
{
    listBox1.Items.Add(format.Name);
    format = formatCltn.NextFormat();
}
```



## 6.68 VcTableFormatField



An object of the type `VcTableFormatField` represents a field of a `VcTableFormat`-Object. A table format field does not have a name as many other objects, but it has an index that defines its position in the table format.

### Properties

- Alignment
- BackgroundColor
- BackgroundColorDataFieldIndex
- BackgroundColorMapName
- BottomMargin
- ConstantText
- FormatName
- GraphicsFileName
- GraphicsFileNameDataFieldIndex
- GraphicsFileNameMapName
- GraphicsHeight
- Index
- LeftMargin
- MaximumTextLineCount
- MinimumTextLineCount
- MultiState
- Pattern
- RightMargin
- TextAndGraphicsCombined
- 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	VcFormatFieldAlignment	Alignment of the field content
	<b>Possible Values:</b> .vcFFABottom 28 .vcFFABottomLeft 27 .vcFFABottomRight 29 .vcFFACenter 25 .vcFFALeft 24 .vcFFARight 26 .vcFFATop 22 .vcFFATopLeft 21 .vcFFATopRight 23	Bottom Bottom left Bottom right Center Left Right Top Top left Top right

### BackgroundColor

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.

If the table format field shall have the color of the table format, select the value **-1**.

If in the property **BackgroundColorMapName** a map is specified, the map will set the background color in dependence on the data.

	Data Type	Explanation
Property value	System.Drawing.Color	RGB color values {0...255},{0...255},{0...255} Default value: -1

## BackgroundColorDataFieldIndex

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 **BackgroundColorMapName**. If you set this property to **-1**, no map will be used, then the Text Font Color will be used without any mapping.

	Data Type	Explanation
Property value	System.Int16	Data field index

## BackgroundColorMapName

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 **BackgroundColorDataFieldIndex**, then the background color is controlled by the map. If no data field entry applies, the background color that is specified in the property **BackgroundColor** will be used.

	Data Type	Explanation
Property value	System.String	Name of the color map

## BottomMargin

Property of VcTableFormatField

This property lets you set or retrieve the width of the bottom margin of the table format field.

	Data Type	Explanation
Property value	System.Int16	Width (in mm) of the bottom margin of the table format field  0...9

## 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 **TextDataFieldIndex** was set to **-1**.

	Data Type	Explanation
Property value	System.String	Constant text

## FormatName

### Read Only Property of VcTableFormatField

This property lets you retrieve the name of the table format to which this field belongs.

	Data Type	Explanation
Property value	System.String	Name of the table format

## 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)
- \*.EMF, with EMF+ included

- \*.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, 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	System.String	Name of the graphics file

## 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 in property **GraphicsFileName** 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	System.Int16	Index of the data field

## 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 **GraphicsFileNameDataFieldIndex**, 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	System.String	Name of the graphics map

## GraphicsHeight

Property of VcTableFormatField

This property lets you set or retrieve for the type **vcFFTGraphics** the height of the graphics in the table format field.

	Data Type	Explanation
Property value	System.Int16	Height of the graphics in mm 0 ... 99

## Index

Read Only Property of VcTableFormatField

This property lets you retrieve the index of the table format field in the associated table format.

	Data Type	Explanation
Property value	System.Int16	Index of the table format field

## LeftMargin

Property of VcTableFormatField

This property lets you set or retrieve the width of the left margin of the table format field.

	Data Type	Explanation
Property value	System.Int16	Width (in mm) of the left margin of the table format field 0...9

## 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	System.Int16	Maximum number of lines 0...9

## 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. Also see the property **MaximumTextLineCount**. 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	System.Int16	Minimum number of lines 0...9

## 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	System.Boolean	Multi-state field (True) / no multi-state field (False)

## Pattern

### 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	VcFieldFillPattern	Pattern type
	<b>Possible Values:</b> .vcAeroGlassPattern 44  .vcFieldNoPattern 1276 .vcFieldVerticalBottomLightedConvexPattern 43  .vcFieldVerticalConcavePattern 40  .vcFieldVerticalConvexPattern 41  .vcFieldVerticalTopLightedConvexPattern 42	Vertical color gradient in the color of the fill pattern   No fill pattern Vertical color gradient from bright to dark   Vertical color gradient from dark to bright to dark   Vertical color gradient from bright to dark to bright   Vertical color gradient from dark to bright 



## RightMargin

Property of VcTableFormatField

This property lets you set or retrieve the width of the right margin of the table format field.

	Data Type	Explanation
Property value	System.Int16	Width (in mm) of the right margin of the table format field 0...9

## TextAndGraphicsCombined

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	System.Boolean	Combi field (True)/ no combi field (False)

## TextDataFieldIndex

Property of VcTableFormatField

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. This property only works if the type of the data field is **vcFFTText**. If the value of the index equals **-1**, the content of the property **ConstantText** will be returned instead.

	Data Type	Explanation
Property value	System.Int16	Index of the data field

## 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 in the property **TextFontMapName** a map was set, the map will control the text font color in dependence of the data.

	Data Type	Explanation
Property value	Font	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 **TextFontMapName**, the map will control the text font color in dependence of the data.

	Data Type	Explanation
Property value	System.Drawing.Color	Font color of the table format

## 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	System.Int16	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	System.String	Name of the font color map

## TextFontDataFieldIndex

Property of VcTableFormatField

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	System.Int16	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	System.String	Name of the font map

## TopMargin

Property of VcTableFormatField

This property lets you set or retrieve the width of the top margin of the table format field.

	Data Type	Explanation
Property value	System.Int16	Width (in mm) of the top margin of the table format field 0...9

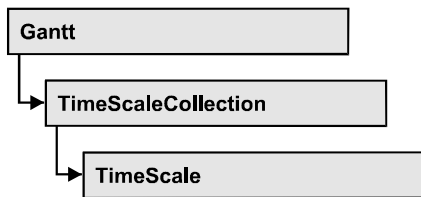
## Type

Property of VcTableFormatField

This property lets you set or retrieve the type of the table format field.

	Data Type	Explanation
<b>Property value</b>	VcFormatFieldType  <b>Possible Values:</b> .vcFFTGraphics 64 .vcFFTText 36	Type of the table format field  Graphics Text

## 6.69 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
- CalendarGridsVisible
- DateGridsVisible
- Font
- FontColor
- Name
- Ribbon
- Section
- ThreeDEffect

---

## Properties

### BackgroundColor

Property of VcTimeScale

This property lets you set or retrieve the background color of the time scale.

	Data Type	Explanation
Property value	System.Drawing.Color	RGB color values {0...255},{0...255},{0...255}

**Example Code VB.NET**

```
Dim timescale As VcTimeScale

timescale = VcGantt1.TimeScaleCollection.Active
timescale.BackgroundColor = Color.Blue
```

**Example Code C#**

```
VcTimeScale timeScale = vcGantt1.TimeScaleCollection.Active;
timeScale.BackgroundColor = Color.LightSteelBlue;
```

## CalendarGridsVisible

**Property of VcTimeScale**

This property lets you set or retrieve whether workfree periods will be marked by gray shadings. This property also can be set in the **Specify Time Scale/ Edit time scale section** dialog.

	Data Type	Explanation
Property value	System.Boolean	Workfree periods are/are not displayed in gray.

**Example Code VB.NET**

```
Dim timescale As VcTimeScale

timescale = VcGantt1.TimeScaleCollection.Active
timescale.CalendarGridsVisible = True
```

**Example Code C#**

```
VcTimeScale timeScale = vcGantt1.TimeScaleCollection.Active;
timeScale.CalendarGridsVisible = true;
```

## DateGridsVisible

**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	System.Boolean	Date grids are/are not displayed.

**Example Code VB.NET**

```
Dim timescale As VcTimeScale

timescale = VcGantt1.TimeScaleCollection.Active
timescale.DateGridsVisible = True
```

### Example Code C#

```
VcTimeScale timeScale = vcGantt1.TimeScaleCollection.Active;  
timeScale.DateGridsVisible = true;
```

## Font

### Property of VcTimeScale

This property lets you set or retrieve all font attributes of the timescale.

	Data Type	Explanation
Property value	Font	Font attributes of the timescale

### Example Code VB.NET

```
Dim newFont As Font  
newFont = VcGantt1.TimeScaleCollection.Active.Font  
MsgBox(newFont.ToString())
```

### Example Code C#

```
Font newFont = vcGantt1.TimeScaleCollection.Active.Font;  
MessageBox.Show(newFont.ToString());
```

## FontColor

### Property of VcTimeScale

This property lets you set or retrieve the font color of the time scale.

	Data Type	Explanation
Property value	System.Drawing.Color	RGB color values ({0...255},{0...255},{0...255})

### Example Code VB.NET

```
Dim timescale As VcTimeScale  
  
timescale = VcGantt1.TimeScaleCollection.Active  
timescale.FontColor = Color.Blue
```

### Example Code C#

```
VcTimeScale timeScale = vcGantt1.TimeScaleCollection.Active;  
timeScale.FontColor = Color.LightSteelBlue;
```

## Name

### Property of VcTimeScale

This property lets you set or retrieve the name of the time scale.

	Data Type	Explanation
Property value	System.String	Name

**Example Code VB.NET**

```
Dim timescale As VcTimeScale

timescale = VcGantt1.TimeScaleCollection.Active
MsgBox("Active timescale: " + timescale.Name)
```

**Example Code C#**

```
VcTimeScale timescale = vcGantt1.TimeScaleCollection.Active;
MessageBox.Show("Active timescale: " + timescale.Name);
```

## Ribbon

**Read Only Property of VcTimeScale**

This property gives access to the ribbons of a time scale.

The property Ribbon is an Indexed Property, which in C# can be addressed by the method `get_Ribbon(ribbonIndex, sectionIndex)`.

	Data Type	Explanation
<b>Parameter:</b>		
⇒ sectionIndex	System.Int16	Index of the time scale section
⇒ ribbonIndex	System.Int16	Index of the ribbon
<b>Property value</b>	VcRibbon	Ribbon object

**Example Code VB.NET**

```
Dim timescale As VcTimeScale
Dim ribbon As VcRibbon

timescale = VcGantt1.TimeScaleCollection.Active
ribbon = timescale.Ribbon(0, 0)
```

**Example Code C#**

```
VcTimeScale timescale = vcGantt1.TimeScaleCollection.Active;
VcRibbon ribbon = timescale.get_Ribbon(0,0);
```

## Section

**Read Only Property of VcTimeScale**

This property gives access to the sections of a timescale.



## 992 API Reference: VcTimeScale

The property `Section` is an Indexed Property, which in C# is addressed by the method `get_Section` (`sectionIndex`).

	Data Type	Explanation
<b>Parameter:</b> ⇒ <code>sectionIndex</code>	System.Int16	Index of the section
<b>Property value</b>	VcSection	Section object

### Example Code VB.NET

```
Dim timescale As VcTimeScale
Dim section As VcSection

timescale = VcGantt1.TimeScaleCollection.Active
section = timescale.Section(0)
```

### Example Code C#

```
VcTimeScale timeScale = vcGantt1.TimeScaleCollection.Active;
VcSection section = timeScale.get_Section(0);
```

## 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
<b>Property value</b>	System.Boolean	3D effect switched on (True)/switched off (False)

### Example Code VB.NET

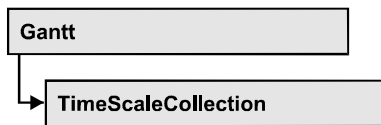
```
Dim timescale As VcTimeScale

timescale = VcGantt1.TimeScaleCollection.Active
timeScale.ThreeDEffect = False
```

### Example Code C#

```
VcTimeScale timeScale = vcGantt1.TimeScaleCollection.Active;
timeScale.ThreeDEffect = false;
```

## 6.70 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

- Count

### Methods

- FirstTimeScale
- GetEnumerator
- NextTimeScale
- TimeScaleByIndex
- TimeScaleByName

---

## Properties

### 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	System.Int32	Number of time scales

### Example Code VB.NET

```

Dim numberOfTimeScales As Integer
numberOfTimeScales = VcGantt1.TimeScaleCollection.Count
  
```

**Example Code C#**

```
int 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 VB.NET**

```
Dim timeScaleCltn As VcTimeScaleCollection
Dim timeScale As VcTimeScale

timeScaleCltn = VcGantt1.TimeScaleCollection
timeScale = timeScaleCltn.FirstTimeScale
```

**Example Code C#**

```
VcTimeScaleCollection timeScaleCltn = vcGantt1.TimeScaleCollection;
VcTimeScale timeScale = timeScaleCltn.FirstTimeScale();
```

### GetEnumerator

**Method of VcTimeScaleCollection**

This method returns an Enumerator object which supports the iteration by language specific elements. It is implied in the For...Each construct of Visual Basic and C#. This object allows to iterate over the time scale objects included.

	Data Type	Explanation
Return value	VcObject	Reference object

## 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
<b>Return value</b>	VcTimeScale	Succeeding time scale

### Example Code VB.NET

```
Dim timeScaleCltn As VcTimeScaleCollection
Dim timeScale As VcTimeScale

timeScaleCltn = VcGantt1.TimeScaleCollection
timeScale = timeScaleCltn.FirstTimeScale
While Not timeScale Is Nothing
    ListBox1.Items.Add(timeScale.Name)
    timeScale = timeScaleCltn.NextTimeScale
End While
```

### Example Code C#

```
VcTimeScaleCollection timeScaleCltn = vcGantt1.TimeScaleCollection;
VcTimeScale timeScale = timeScaleCltn.FirstTimeScale();
while (timeScale != null)
{
    listBox1.Items.Add(timeScale.Name);
    timeScale = timeScaleCltn.NextTimeScale();
}
```

## TimeScaleByIndex

Method of VcTimeScaleCollection

This method lets you access a time scale by its index. If a time scale does not exist at the index specified, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
<b>Parameter:</b> ⇒ index	System.Int16	Index of the time scale
<b>Return value</b>	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
<b>Parameter:</b>		
⇒ timeScaleName	System.String	Name of the time scale
<b>Return value</b>	VcTimeScale	Time scale

### Example Code VB.NET

```
Dim timeScaleCltn As VcTimeScaleCollection

timeScaleCltn = VcGantt1.TimeScaleCollection
timeScaleCltn.Active = timeScaleCltn.TimeScaleByName("Days")
```

### Example Code C#

```
VcTimeScaleCollection timeScaleCltn = vcGantt1.TimeScaleCollection;
timeScaleCltn.Active = timeScaleCltn.TimeScaleByName("Days");
```

---



---

## 7 Index

### A

**AbsoluteBottomMarginInCM**

Property of

VcPrinter 854

**AbsoluteLeftMarginInCM**

Property of

VcPrinter 854

**AbsoluteRightMarginInCM**

Property of

VcPrinter 855

**AbsoluteTopMarginInCM**

Property of

VcPrinter 855

**Active**

Property of

VcCalendarCollection 358

VcHistogramCollection 681

VcNumericScaleCollection 848

**ActiveNodeFilter**

Property of

VcGantt 510

**Activities**

hierarchical arrangement 511

**Activity 116**

saving and reloading order 287

**ActualEndDateDataFieldIndex**

Property of

VcScheduler 948

**ActualStartDateDataFieldIndex**

Property of

VcScheduler 948

**Add**

Method of

VcBoxCollection 322

VcBoxFormatCollection 334

VcCalendarCollection 360

VcCurveCollection 412

VcDataRecordCollection 438

VcDataTableCollection 449

VcDataTableFieldCollection 462

VcFilterCollection 496

VcGroupLevelLayoutCollection  
670

VcIntervalCollection 700

VcLayerCollection 735

VcLineFormatCollection 754

VcLinkAppearanceCollection 783

VcMapCollection 800

**AddBySpecification**

Method of

VcBoxCollection 322

VcBoxFormatCollection 334

VcCalendarCollection 360

VcCurveCollection 413

VcFilterCollection 497

VcGroupLevelLayoutCollection  
670

VcIntervalCollection 700

VcLayerCollection 735

VcLineFormatCollection 754

VcLinkAppearanceCollection 784

VcMapCollection 800

**AddDuration**

Method of

VcCalendar 352

**Addend**

Property of

VcCurve 380

**AddSubCondition**

Method of  
VcFilter 492

**AdjustToReferenceDate**

Property of  
VcDateLineGrid 481

**Administrate calendar profiles**

dialog 241

**AJAX 63**

**Ajax extensions 150**

**AjaxExtensionsEnabled**

Property of  
VcGantt 511

**Alignment**

Property of  
VcBoxFormatField 340  
VcLayerFormatField 743  
VcLineFormatField 759  
VcPrinter 856, 857  
VcTableFormatField 977

**AllData**

Property of  
VcDataRecord 431  
VcLink 767  
VcNode 817

**AllNodesInOneRow**

Property of  
VcGroupLevelLayout 655

**AlwaysCurrentDate**

Property of  
VcDateLine 467

**AnnotationAtBottom**

Property of  
VcDateLineGrid 481

**AnnotationAtCenter**

Property of  
VcDateLineGrid 481

**AnnotationAtTop**

Property of  
VcDateLineGrid 482

**Arrangement**

Property of  
VcGantt 511

**ASP .NET AJAX 298**

**ASP.NET**

interactions 35

**ASPX 293**

**AssignmentDataTableName**

Property of  
VcResourceScheduler2 879

**AssignmentIsResultFieldIndex**

Property of  
VcResourceScheduler2 881

**AssignmentIsVisibleFieldIndex**

Property of  
VcResourceScheduler2 881

**AssignmentLoadOrConsumptionPerItemFieldIndex**

Property of  
VcResourceScheduler2 882

**AssignmentMaximumLoadFieldIndex**

Property of  
VcResourceScheduler2 883

**AssignmentMinimumLoadFieldIndex**

Property of  
VcResourceScheduler2 883

**AssignmentOperationIDFieldIndex**

Property of  
VcResourceScheduler2 884

**AssignmentResourceIDFieldIndex**

Property of  
VcResourceScheduler2 885

**AssignmentResourceSelectionStrategyFieldIndex**

Property of  
 VcResourceScheduler2 885

**AutomaticSchedulingEnabled**  
 Property of  
 VcScheduler 948

**Autoschedule 167**

## B

**Background color**  
 selected row 160

**BackgroundColor**  
 Property of  
 VcBoxFormatField 341  
 VcCalendarGrid 366  
 VcInterval 687  
 VcLayer 705  
 VcLineFormatField 760  
 VcNumericScale 839  
 VcRibbon 938  
 VcTableFormatField 977  
 VcTimeScale 988

**BackgroundColorDataFieldIndex**  
 Property of  
 VcCalendarGrid 366  
 VcLayer 706  
 VcLineFormatField 760  
 VcTableFormatField 978

**BackgroundColorMapName**  
 Property of  
 VcCalendarGrid 367  
 VcLayer 707  
 VcLineFormatField 761  
 VcTableFormatField 978

**Bars**  
 moving into visible area 285

**BaseTimeUnit**  
 Property of

VcResourceScheduler2 886

**BaseTimeUnitsPerStep**  
 Property of  
 VcResourceScheduler2 887

**BorderArea**  
 Property of  
 VcGantt 512  
 see also  
 VcBorderArea 301

**BorderBox**  
 Method of  
 VcBorderArea 301  
 see also  
 VcBorderBox 303

**Bottom**  
 Property of  
 VcRect 872

**BottomMargin**  
 Property of  
 VcTableFormatField 978

**Box 65**  
 allow multiple marking 149  
 by index 323  
 marking 315  
 see also  
 VcBox 311

**Box format**  
 by index 336

**Box format field**  
 alignment 340  
 background color 341  
 fill pattern 763, 983  
 font color 347  
 font type 346  
 height of graphics 342  
 index 343  
 maximum number of lines 343



## **1000** Index

minimum number of lines 344  
minimum width 345  
ofrmat name 342  
pattern 345  
type 347

### **BoxByIndex**

Method of  
VcBoxCollection 323

### **BoxByName**

Method of  
VcBoxCollection 324

### **BoxCollection**

Property of  
VcGantt 512  
see also  
VcBoxCollection 321

### **Boxes**

convert pixel to offset 320

### **BoxFormat**

see also  
VcBoxFormat 328

### **BoxFormatCollection**

Property of  
VcGantt 513  
see also  
VcBoxFormatCollection 333

### **BoxFormatField**

see also  
VcBoxFormatField 340

## **C**

### **CalcDuration**

Method of  
VcCalendar 352

### **CalculateCurrentWidth**

Method of  
VcLayer 733

### **CalculateLineCount**

Method of  
VcLayerFormatField 749

### **Calendar 98**

by index 361  
name 376  
number of seconds of a workday 351  
see also  
VcCalendar 349

### **Calendar grid 255**

background color 367  
background color 366  
line color 367, 368  
line color map 368  
line type 368  
name 369

### **Calendar Grids**

administrate 203

### **Calendar profile**

order 378  
type 377

### **CalendarByIndex**

Method of  
VcCalendarCollection 361

### **CalendarByName**

Method of  
VcCalendarCollection 361

### **CalendarCollection**

Property of  
VcGantt 513  
see also  
VcCalendarCollection 358

### **CalendarGrid**

Property of  
VcSection 955  
see also  
VcCalendarGrid 365

**CalendarGridCollection**

Property of  
VcGantt 513

**CalendarGridName**

Property of  
VcGroupLevelLayout 655  
VcNodeLevelLayout 831

**CalendarGridsVisible**

Property of  
VcGroupLevelLayout 656  
VcHistogram 674  
VcNodeLevelLayout 832  
VcTimeScale 989

**CalendarGridsWithChildGroups**

Property of  
VcGroupLevelLayout 656

**CalendarName**

Property of  
VcCalendarGrid 367  
VcHistogram 675  
VcRibbon 938

**CalendarNameDataFieldIndex**

Property of  
VcGroupLevelLayout 656

**CalendarProfile**

see also  
VcCalendarProfile 376

**CalendarProfileCollection**

Property of  
VcCalendar 350  
VcGantt 514

**CalendarProfileName**

Property of  
VcInterval 688

**Clear**

Method of  
VcCurve 402

**CollapseColumn**

Property of  
VcTableFormat 966

**Collapsed**

Property of  
VcGroupLevelLayout 656

**Color**

Property of  
VcMapEntry 806

**ColumnTitle**

Property of  
VcTable 960

**ColumnWidth**

Property of  
VcTable 961

**ComparisonValueAsString**

Property of  
VcFilterSubCondition 501

**CompletionDataFieldIndex**

Property of  
VcLayer 709

**Configuration 62, 149****ConnectionOperator**

Property of  
VcFilterSubCondition 502

**ConsiderFilterEntries**

Property of  
VcMap 793

**ConstantText**

Property of  
VcLayerFormatField 744  
VcLineFormatField 761  
VcTableFormatField 979

**ConvertDistance**

Method of  
VcGantt 554

**Copy**

- Method of
  - VcBoxCollection 324
  - VcBoxFormatCollection 335
  - VcCalendarCollection 361
  - VcCurveCollection 413
  - VcDataTableCollection 449
  - VcDataTableFieldCollection 462
  - VcFilterCollection 497
  - VcGroupLevelLayoutCollection 671
  - VcIntervalCollection 701
  - VcLayerCollection 736
  - VcLineFormatCollection 755
  - VcLinkAppearanceCollection 784
  - VcMapCollection 801
- CopyFormatField**
  - Method of
    - VcBoxFormat 331
    - VcLayerFormat 741
    - VcLineFormat 752
- CopySubCondition**
  - Method of
    - VcFilter 493
- Count**
  - Property of
    - VcBoxCollection 321
    - VcBoxFormatCollection 333
    - VcCalendarCollection 359
    - VcCurveCollection 411
    - VcDataDefinitionTable 425
    - VcDataRecordCollection 437
    - VcDataTableCollection 448
    - VcDataTableFieldCollection 461
    - VcDateLineCollection 475
    - VcFilterCollection 495
    - VcGroupCollection 650
    - VcGroupLevelLayoutCollection 669
    - VcHistogramCollection 682
    - VcIntervalCollection 700
    - VcLayerCollection 734
    - VcLineFormatCollection 753
    - VcLinkAppearanceCollection 782
    - VcLinkCollection 789
    - VcMap 794
    - VcMapCollection 799
    - VcNodeCollection 827
    - VcNumericScaleCollection 849
    - VcTableFormatCollection 972
    - VcTimeScaleCollection 993
- CreateDataDefinitionField**
  - Method of
    - VcDataDefinitionTable 426
- CreateEntry**
  - Method of
    - VcMap 796
- CurrentHorizontalPagesCount**
  - Property of
    - VcPrinter 857
- CurrentZoomFactor**
  - Property of
    - VcPrinter 858
- Curve**
  - by index 414
  - see also
    - VcCurve 379
- CurveByIndex**
  - Method of
    - VcCurveCollection 414
- CurveByName**
  - Method of
    - VcCurveCollection 414
- CurveCollection**

Property of  
     VcHistogram 675  
 see also  
     VcCurveCollection 411

**CuttingMarks**

Property of  
     VcPrinter 858

**D****Data**

loading files 283

**Data binding 23****Data field**

for tooltip text 155

**Data record**

add to collection 438  
 all data 431  
**by ID** 440  
 data field 432  
 deleting 434  
 depending data record not found 594  
 enumerator object 441  
 ID 434  
 iteration, initial value 440  
 iteration, subsequent values 442  
 name of associated table 433  
 number in collection 437  
 related data record 435  
 remove from collection 443  
 unique ID 442  
 update 444  
 updating 436

**Data table**

add to collection 449  
 by index 450  
 by name 451  
 copy within collection 450

data record collection 445  
 data table field collection 446  
 description 446  
 enumerator object 429, 452  
 Extended data tables 519  
 Iteration, primary value 451  
 Iteration, subsequent values 452  
 name 555  
 name 447  
 number in collection 448  
 update 453

**Data table field**

add to collection 462  
 associated date table 454  
 by index 463  
 by name 464  
 copying 462  
 data type 460  
 date format 455  
 editable 456  
 enumerator object 465  
 index 556  
 index 457  
 iteration, initial value 464  
 iteration, subsequent values 465  
 name 555  
 name 457  
 number in collection 461  
 primary key 458  
 related field index 458

**Data tables**

extended 149

**Data Tables 69****DataDefinition**

Property of  
     VcGantt 514  
 see also

- VcDataDefinition 418
- DataDefinitionField**
  - see also
    - VcDataDefinitionField 420
- DataDefinitionFieldByIndex**
  - Method of
    - VcDataDefinitionTable 427
- DataDefinitionFieldByName**
  - Method of
    - VcDataDefinitionTable 427
- DataDefinitionTable**
  - Property of
    - VcDataDefinition 418
    - VcFilter 490
  - see also
    - VcDataDefinitionTable 425
- DataField**
  - Property of
    - VcDataRecord 432
    - VcGroup 640
    - VcLink 768
    - VcNode 817
- DataFieldIndex**
  - Property of
    - VcFilterSubCondition 502
- DataFieldValue**
  - Property of
    - VcMapEntry 807
- DataRecord**
  - Method of
    - VcGroup 647
    - VcLink 770
    - VcNode 821
  - see also
    - VcDataRecord 431
- DataRecordByID**
  - Method of
    - VcDataRecordCollection 439
- DataRecordCollection**
  - Property of
    - VcDataTable 445
  - see also
    - VcDataRecordCollection 437
- DataRecordEventsEnabled**
  - Property of
    - VcResourceScheduler2 887
- DataTable**
  - see also
    - VcDataTable 445
- DataTableByIndex**
  - Method of
    - VcDataTableCollection 450
- DataTableByName**
  - Method of
    - VcDataTableCollection 451
- DataTableCollection**
  - Property of
    - VcGantt 515
  - see also
    - VcDataTableCollection 448
- DataTableField**
  - see also
    - VcDataTableField 454
- DataTableFieldByIndex**
  - Method of
    - VcDataTableFieldCollection 463
- DataTableFieldByName**
  - Method of
    - VcDataTableFieldCollection 464
- DataTableFieldCollection**
  - Property of
    - VcDataTable 446
  - see also
    - VcDataTableFieldCollection 461

**DataTableName**

Property of

VcDataRecord 433

VcDataTableField 454

**Date**

Property of

VcDateLine 468

**Date line 76**

by index 476

DateLineCollection 475

**Date line grid**

line color 483

line color map 483, 718

reference date 488

**Date lines**

order 474

**Date output format 147****DateFormat**

Property of

VcDataDefinitionField 420

VcDataTableField 455

**DateGridsVisible**

Property of

VcTimeScale 989

**DateLine**

see also

VcDateLine 467

**DateLineByIndex**

Method of

VcDateLineCollection 476

**DateLineByName**

Method of

VcDateLineCollection 476

**DateLineCollection**

Property of

VcGantt 515

see also

VcDateLineCollection 475

**DateLineGrid**

Property of

VcSection 956

see also

VcDateLineGrid 480

**DateLineGridName**

Property of

VcGroupLevelLayout 657

**DateLineGridsVisible**

Property of

VcGroupLevelLayout 657

**DateLineGridsWithChildGroups**

Property of

VcGroupLevelLayout 657

**DateOutputFormat**

Property of

VcGantt 515

VcLineFormatField 761

VcRibbon 938

**DatesWithHourAndMinute**

Property of

VcFilter 490

**DayInEndMonth**

Property of

VcInterval 688

**DayInStartMonth**

Property of

VcInterval 688

**DefaultOperationMaximumInterruptionTime**

Property of

VcResourceScheduler2 888

**DefaultPrinterName**

Property of

VcPrinter 858

**DefaultResourceCalendarName**

Property of  
VcResourceScheduler2 888

**Delete**

Method of  
VcDataRecord 434  
VcGroup 647  
VcHistogramCollection 682  
VcLink 770  
VcNode 821

**DeleteEntry**

Method of  
VcMap 797

**DeleteLinkRecord**

Method of  
VcGantt 554

**DeleteNodeRecord**

Method of  
VcGantt 555

**DeletePoint**

Method of  
VcCurve 402

**Description**

Property of  
VcDataTable 446

**DetectDataTableFieldName**

Method of  
VcGantt 555

**DetectDataTableName**

Method of  
VcGantt 555

**DetectFieldIndex**

Method of  
VcGantt 556

**DetermineIDOfFirstOperationByTaskID**

Method of  
VcResourceScheduler2 934

**DetermineIDOfLastOperationByTaskID**

Method of  
VcResourceScheduler2 935

**Diagram**

background color 518  
export 61  
second background color for  
alternating lines 517

**Diagram background color 160**

**DiagramAlternatingRowBackgroundColor**

Property of  
VcGantt 517

**DiagramBackgroundColor**

Property of  
VcGantt 517

**DiagramEnabled**

Property of  
VcPrinter 859

**DiagramHistogramHeightRatio**

Property of  
VcGantt 518

**Dialog box**

Administrate Box Formats 217  
Administrate Boxes 213  
Administrate Data Tables 168  
Administrate Filters 184  
Administrate Histograms 257  
Administrate Line formats 190, 192  
Administrate Maps 208  
Configure Mapping 212  
Edit Box Format 219  
Edit Boxes 216  
Edit Date Line 268  
Edit Filter 186  
Edit Histogram 259  
Edit Layer 175

- Edit Layer Format 180
  - Edit Map 210
  - Edit Table 228
  - Edit Table Format 230
  - Edit Time Scale Section 251
  - grouping 195
  - Licensing 275
  - Line Attributes 235
  - Pattern 236
  - Select Curve Data Source 263
  - Select Ribbon Type 264
  - Specification of Texts, Graphics and Legend 270
  - Specify Bar Appearance 171
  - Specify Calendars 237
  - Specify Date Lines 266
  - Specify Table 226
  - Specify Time Scale 248
  - DocumentName**
    - Property of
      - VcPrinter 859
  - Double output format 148**
  - DST 77**
  - Duration**
    - Property of
      - VcInterval 689
  - DurationDataFieldIndex**
    - Property of
      - VcLayer 709
      - VcScheduler 949
- E**
- EarlyEndDateDataFieldIndex**
    - Property of
      - VcScheduler 949
  - EarlyStartDateDataFieldIndex**
    - Property of
      - VcScheduler 949
  - Editable**
    - Property of
      - VcDataDefinitionField 421
      - VcDataTableField 456
  - Enabled**
    - Property of
      - VcGantt 518
  - End date**
    - calculate 29
  - EndDataFieldIndex**
    - Property of
      - VcLayer 709
  - EndDateForAutomaticScheduling**
    - Property of
      - VcGantt 519
      - VcScheduler 949
  - EndDateNotLaterThanDataFieldIndex**
    - Property of
      - VcScheduler 950
  - EndTime**
    - Property of
      - VcInterval 689
  - EndLoading**
    - Method of
      - VcGantt 557
  - EndMonth**
    - Property of
      - VcInterval 689
  - EndTime**
    - Property of
      - VcInterval 690
  - EndWeekday**
    - Property of
      - VcInterval 690
  - Error messages 279, 284**
  - Event argument objects**



- VcBoxClickingEventArgs 584
  - VcComponentScrolledEventArgs 586
  - VcComponentScrollingEventArgs 589
  - VcCurveClickingEventArgs 592
  - VcDataRecordModifiedEventArgs 592
  - VcDataRecordModifyingEventArgs 593
  - VcDataRecordNotFoundEventArgs 594
  - VcDiagramClickingEventArgs 598
  - VcDiagramHorizontalScrolledEventArgs 595
  - VcDiagramHorizontalScrollingEventArgs 596
  - VcErrorOccurringEventArgs 599
  - VcFieldSelectingEventArgs 600
  - VcGroupClickingEventArgs 601
  - VcGroupModifiedEventArgs 602
  - VcGroupModifyingEventArgs 603
  - VcGroupsMarkedEventArgs 604
  - VcHistogramClickingEventArgs 606
  - VcHistogramsHeightChangedEventArgs 607
  - VcHistogramsHeightChangingEventArgs 608
  - VcLinksClickingEventArgs 608
  - VcNodeClickingEventArgs 610
  - VcNodeModifiedEventArgs 610
  - VcNodeModifiedExEventArgs 611
  - VcNodeModifyingEventArgs 613
  - VcNodesMarkedEventArgs 614
  - VcNodesMarkingEventArgs 605, 615
  - VcNumericScaleClickingEventArgs 616
  - VcObjectDrawingEventArgs 617
  - VcObjectDrawnEventArgs 618
  - VcResourceSchedulingProgressingEventArgs 619
  - VcTableCaptionClickingEventArgs 622
  - VcTableColumnWidthChangingEventArgs 623
  - VcTableWidthChangingEventArgs 624
  - VcTextEntrySupplyingEventArgs 625
  - VcTimeScaleClickingEventArgs 635
  - VcToolTipTextSupplyingEventArgs 637
- Events 79**
- VcBoxLeftClicking
    - VcGantt 584
  - VcComponentScrolled
    - VcGantt 585
  - VcComponentScrolling
    - VcGantt 588
  - VcCurveLeftClicking
    - VcGantt 591
  - VcDataRecordModified
    - VcGantt 592
  - VcDataRecordModifying
    - VcGantt 593
  - VcDataRecordNotFound
    - VcGantt 594
  - VcDiagramHorizontalScrolled
    - VcGantt 594
  - VcDiagramHorizontalScrolling
    - VcGantt 596
  - VcDiagramLeftClicking
    - VcGantt 598
  - VcErrorOccurring
    - VcGantt 599
  - VcFieldSelecting
    - VcGantt 600
  - VcGroupLeftClicking
    - VcGantt 601
  - VcGroupModified

VcGantt 601  
 VcGroupModifying  
     VcGantt 602  
 VcGroupsMarked  
     VcGantt 604  
 VcGroupsMarking  
     VcGantt 605  
 VcHistogramLeftClicking  
     VcGantt 606  
 VcHistogramsHeightChanged  
     VcGantt 607  
 VcHistogramsHeightChanging  
     VcGantt 607  
 VcLinksLeftClicking  
     VcGantt 608  
 VcNodeLeftClicking  
     VcGantt 609  
 VcNodeModified  
     VcGantt 610  
 VcNodeModifiedEx  
     VcGantt 611  
 VcNodeModifying  
     VcGantt 612  
 VcNodesMarked  
     VcGantt 614  
 VcNodesMarking  
     VcGantt 614  
 VcNumericScaleLeftClicking  
     VcGantt 615  
 VcObjectDrawing  
     VcGantt 616  
 VcObjectDrawn  
     VcGantt 618  
 VcResourceSchedulingProgressing  
     VcGantt 619  
 VcResourceSchedulingWarning  
     VcGantt 620

VcTableCaptionLeftClicking  
     VcGantt 622  
 VcTableColumnWidthChanging  
     VcGantt 623  
 VcTableWidthChanging  
     VcGantt 624  
 VcTextEntrySupplying  
     VcGantt 625  
 VcTimeScaleLeftClicking  
     VcGantt 635  
 VcTimeScaleSectionRescaled  
     VcGantt 636  
 VcToolTipTextSupplying  
     VcGantt 636

**ExportGraphicsToFileEx**

Method of  
     VcGantt 557

**ExtendedDataTablesEnabled**

Property of  
     VcGantt 519

## F

**FieldsSeparatedByLines**

Property of  
     VcBoxFormat 328  
     VcTableFormat 966

**FieldText**

Property of  
     VcBox 311

**File not found 295****FilePath**

Property of  
     VcGantt 520

**FillReference1BackgroundColor**

Property of  
     VcCurve 381

**FillReference1Name**

- Property of
  - VcCurve 381
- FillReference1Pattern**
  - Property of
    - VcCurve 382
- FillReference1PatternColor**
  - Property of
    - VcCurve 385
- FillReference2Color**
  - Property of
    - VcCurve 386
- FillReference2Name**
  - Property of
    - VcCurve 387
- FillReference2Pattern**
  - Property of
    - VcCurve 387
- FillReference2PatternColor**
  - Property of
    - VcCurve 390
- Filter 80**
  - by index 498
  - comparison value 187
  - see also
    - VcFilter 489
  - using 42
- FilterByIndex**
  - Method of
    - VcFilterCollection 498
- FilterByName**
  - Method of
    - VcFilterCollection 498
- FilterCollection**
  - Property of
    - VcGantt 520
  - see also
    - VcFilterCollection 495
- FilterName**
  - Property of
    - VcCurve 391
    - VcFilterSubCondition 503
    - VcLayer 710
    - VcLinkAppearance 773
    - VcTableFormat 967
- Filters**
  - administration 184
  - editing 186
- FilterSubCondition**
  - see also
    - VcFilterSubCondition 501
- FirstBox**
  - Method of
    - VcBoxCollection 325
- FirstCalendar**
  - Method of
    - VcCalendarCollection 362
- FirstCurve**
  - Method of
    - VcCurveCollection 415
- FirstDataDefinitionField**
  - Method of
    - VcDataDefinitionTable 428
- FirstDataRecord**
  - Method of
    - VcDataRecordCollection 440
- FirstDataTable**
  - Method of
    - VcDataTableCollection 451
- FirstDataTableField**
  - Method of
    - VcDataTableFieldCollection 464
- FirstDateLine**
  - Method of
    - VcDateLineCollection 477

**FirstFilter**

Method of  
VcFilterCollection 498

**FirstFormat**

Method of  
VcBoxFormatCollection 335  
VcLineFormatCollection 755  
VcTableFormatCollection 973

**FirstGroup**

Method of  
VcGroupCollection 651

**FirstGroupLevelLayout**

Method of  
VcGroupLevelLayoutCollection  
671

**FirstHistogram**

Method of  
VcHistogramCollection 683

**FirstInterval**

Method of  
VcIntervalCollection 701

**FirstLayer**

Method of  
VcLayerCollection 736

**FirstLink**

Method of  
VcLinkCollection 790

**FirstLinkAppearance**

Method of  
VcLinkAppearanceCollection 784

**FirstMap**

Method of  
VcMapCollection 801

**FirstMapEntry**

Method of  
VcMap 797

**FirstNode**

Method of  
VcNodeCollection 828

**FirstNumericScale**

Method of  
VcNumericScaleCollection 849

**FirstTimeScale**

Method of  
VcTimeScaleCollection 994

**FitChartIntoView**

Method of  
VcGantt 559

**FitHistogramsIntoView**

Method of  
VcGantt 560

**FitRangeIntoView**

Method of  
VcGantt 560  
VcHistogram 677

**FitToPage**

Property of  
VcPrinter 860

**FoldingMarksType**

Property of  
VcPrinter 860

**Font**

Property of  
VcNumericScale 840  
VcRibbon 940  
VcTimeScale 990

**FontAntiAliasingEnabled**

Property of  
VcGantt 521

**FontBody**

Property of  
VcMapEntry 808

**FontColor**

Property of

VcNumericScale 840

VcRibbon 941

VcTimeScale 990

### FontName

Property of

VcMapEntry 808

### Fonts 291

anti-aliasing 521

### FontSize

Property of

VcMapEntry 809

### Format

Property of

VcLayer 710

### FormatByIndex

Method of

VcBoxFormatCollection 336

VcLineFormatCollection 756

VcTableFormatCollection 973

### FormatByName

Method of

VcBoxFormatCollection 336

VcLineFormatCollection 756

VcTableFormatCollection 974

### FormatField

Property of

VcBoxFormat 329

VcLayerFormat 740

VcLineFormat 750

VcTableFormat 967

### FormatFieldCount

Property of

VcBoxFormat 329

VcLayerFormat 741

VcLineFormat 751

VcTableFormat 968

### FormatName

Property of

VcBox 312

VcBoxFormatField 342

VcDateLineGrid 482

VcLayerFormatField 744

VcLineFormatField 763

VcTableFormatField 979

### FreeFloatDataFieldIndex

Property of

VcScheduler 950

### FullUsageOfPlanningUnitsEnabled

Property of

VcResourceScheduler2 889

## G

### German Version 18

### GetCurrentViewDates

Method of

VcGantt 561

### GetCurrentYValues

Method of

VcHistogram 677

### GetDate

Method of

VcGantt 561

### GetEndOfPreviousWorktime

Method of

VcCalendar 353

### GetEnumerator

Method of

VcBoxCollection 325

VcBoxFormat 331

VcBoxFormatCollection 337

VcCalendarCollection 362

VcCurveCollection 415

VcDataDefinitionTable 429

VcDataRecordCollection 441

- VcDataTableCollection 452
- VcDataTableFieldCollection 465
- VcDateLineCollection 478
- VcFilter 493
- VcFilterCollection 499
- VcFilterSubCondition 505
- VcGroupCollection 651
- VcGroupLevelLayoutCollection 671
- VcHistogramCollection 683
- VcLayerCollection 737
- VcLayerFormat 741
- VcLinkAppearanceCollection 785
- VcLinkCollection 790
- VcMapCollection 802
- VcNodeCollection 828
- VcTableFormat 971
- VcTableFormatCollection 974
- VcTimeScaleCollection 994
- Property of
  - VcMap 794
- GetFirstOverload**
  - Method of
    - VcCurve 403
- GetFirstOverloadEx**
  - Method of
    - VcCurve 404
- GetLicenseInformation**
  - Method of
    - VcGantt 562
- GetLinkByID**
  - Method of
    - VcGantt 562
- GetLinkByNodeIDs**
  - Method of
    - VcGantt 563
- GetNewUniqueID**
  - Method of
    - VcDataRecordCollection 442
- GetNextIntervalBorder**
  - Method of
    - VcCalendar 354
- GetNextOverload**
  - Method of
    - VcCurve 405
- GetNextOverloadEx**
  - Method of
    - VcCurve 406
- GetNodeByID**
  - Method of
    - VcGantt 563
- GetPositionInView**
  - Method of
    - VcNode 821
- GetPreviousIntervalBorder**
  - Method of
    - VcCalendar 354
- GetStartOfInterval**
  - Method of
    - VcCalendar 355
- GetStartOfNextWorktime**
  - Method of
    - VcCalendar 355
- GetValues**
  - Method of
    - VcCurve 407
- GetValuesEx**
  - Method of
    - VcCurve 408
- GetViewComponentSize**
  - Method of
    - VcGantt 564
- GetXOffset**
  - Method of

VcBox 319

**Graphic**

Export 61

**Graphics Format 82**

**GraphicsFileName**

Property of

VcBoundingBox 303

VcLayer 710

VcMapEntry 809

VcTableFormatField 979

**GraphicsFileNameDataFieldIndex**

Property of

VcLayer 712

VcTableFormatField 980

**GraphicsFileNameMapName**

Property of

VcLayer 714

VcTableFormatField 981

**GraphicsHeight**

Property of

VcBoxFormatField 342

VcTableFormatField 981

**Group**

collapsed 656

ID 641

name 659

see also

VcGroup 639

show date line grids 657

visible 668

**Group level layout**

calendar grid name 655, 657, 658

calendar grids for sub groups 656

date line grids for sub groups 657

**Group levels**

show group nodes 658

**Group title row**

background color 661

fill pattern 662

**GroupByName**

Method of

VcGroupCollection 652

**GroupCollection**

Property of

VcGantt 521

see also

VcGroupCollection 650

**GroupDataFieldIndex**

Property of

VcGroupLevelLayout 658

**Groupi level layout**

grouping level 658

**Grouping 86, 197, 200**

calendar 198

calendar grid 656, 674

collapsing/expanding allowed 200, 658

initially collapsed 200

making overlapping activities in a group visible 286

Separation line color 664

separation line color map name 664

separation lines 199

sorting order 197, 198, 667, 838

**Grouping level**

line thickness 666, 837

line type 837

name of nodes' calendar grid 831

separation line color 664, 835

**Grouping levels**

data field index for color maps of nodes 832, 834

data field index for maps of nodes 834

name of color maps of nodes 833, 834

name of maps of nodes 835

pattern color of nodes 833

row pattern of nodes 833

show calendar grids of nodes 832

show separation lines 665, 836

show separation lines at top 665

specify row background color of nodes 832

**GroupingDataFieldIndex**

Property of  
VcGantt 521

**GroupingLevel**

Property of  
VcGroup 641

**GroupingModificationsAllowed**

Property of  
VcGantt 522

**GroupLevelLayout**

see also  
VcGroupLevelLayout 654

**GroupLevelLayoutByIndex**

Method of  
VcGroupLevelLayoutCollection 672

**GroupLevelLayoutByName**

Method of  
VcGroupLevelLayoutCollection 672

**GroupLevelLayoutCollection**

see also  
VcGroupLevelLayoutCollection 669

**GroupNodes**

Method of  
VcGantt 565

**GroupNodesVisible**

Property of  
VcGroupLevelLayout 658

**GroupOptimizationOnInteractionsEnabled**

Property of  
VcGantt 523

**Groups**

optimization on interactions 151

**GroupSortingDataFieldIndex**

Property of  
VcGantt 523

**GroupSortingOrder**

Property of  
VcGantt 524

## H

**Height**

Property of  
VcLayer 715  
VcRect 872

**Height ratio**

diagram area/histogram 159

**HeightDataFieldIndex**

Property of  
VcLayer 715

**Hidden**

Property of  
VcDataDefinitionField 422  
VcDataTableField 456

**Hierarchical Order 90**

**Hierarchy 196**

**HierarchyDataFieldIndex**

Property of  
VcGantt 524

**Histogram 92, 159**

all histograms in a window 560

assign calendar 675



creating 45  
curve collection 675  
curves 260  
high-low curve values 677  
matching numeric scale 677  
maximum value of the numeric scale 679  
minimum value of the numeric scale 679  
modification of diagram/histogram height ratio 607, 608  
name 675  
order 678  
Property of  
    VcCurve 392  
    VcNumericScale 841  
scale collection 676  
see also  
    VcHistogram 674  
separation line color 525  
visible 676

**HistogramByIndex**

Method of  
    VcHistogramCollection 684

**HistogramByName**

Method of  
    VcHistogramCollection 684

**HistogramCollection**

Property of  
    VcGantt 525  
see also  
    VcHistogramCollection 681

**HistogramSeparationLineColor**

Property of  
    VcGantt 525

**HorAlignment**

Property of  
    VcDateLineGrid 482

**HorizontalOffset**

Property of  
    VcLayer 716

**ID**

Property of  
    VcDataRecord 434  
    VcGroup 641  
    VcLink 769  
    VcNode 818

**IdentifyField**

Method of  
    VcGantt 566

**IdentifyLayerAt**

Method of  
    VcGantt 567

**IdentifyObjectAt**

Method of  
    VcGantt 568

**IIS**

versions 292

**ImportConfiguration**

Method of  
    VcGantt 570

**IncomingLinks**

Property of  
    VcNode 818

**IndentColumn**

Property of  
    VcTableFormat 968

**IndentWidth**

Property of  
    VcTableFormat 969

**Index**

Property of  
    VcBoxFormatField 343

- VcDataDefinitionField 422
- VcDataTableField 457
- VcFilterSubCondition 503
- VcLayerFormatField 744
- VcLineFormatField 763
- VcTableFormatField 981
- InitialRowCount**
  - Property of
    - VcGantt 526
- In-place editing**
  - nodes in table 149
- InsertLinkRecord**
  - Method of
    - VcGantt 571
- InsertNodeRecord**
  - Method of
    - VcGantt 571
- Installation 13**
- Interactions**
  - table and diagram area 35
- Internet 61, 141, 276**
- Interval**
  - Add 700
  - annotation of the time ribbon 697
  - background color 688
  - by index 702
  - calendar profile 688
  - copy 701
  - day of end month 688
  - day of start month 688
  - duration 689
  - end date and time 689
  - end month 689
  - end time 690
  - end weekday 690
  - first weekday 696
  - number 700
  - pattern 692
  - pattern color 695
  - remove 703
  - retrieving an interval by its name 702
  - see also
    - VcInterval 686
  - start date and time 695
  - start month 696
  - start time 696
  - time unit 697
  - type 697
  - usage of graphical attributes 375, 698
- IntervalByIndex**
  - Method of
    - VcIntervalCollection 702
- IntervalByName**
  - Method of
    - VcIntervalCollection 702
- IntervalCollection**
  - Property of
    - VcCalendar 350
    - VcCalendarProfile 376
  - see also
    - VcIntervalCollection 699
- Intervall**
  - erstes Intervall 701
  - nächstes Intervall 702
- Intervall collection**
  - update 703
- Intervalle**
  - line thickness 691
- Intervals**
  - line color 691
  - Specify 239, 242, 243, 244, 246
- IsValid**
  - Method of

VcFilter 494  
VcFilterSubCondition 505

**IsWorktime**

Method of  
VcCalendar 356

**J**

**Jittering 294**

**K**

**KeepingNodesTogetherDataFieldIndex**

Property of  
VcGantt 526

**L**

**LabelSizeDependence**

Property of  
VcLayer 716

**Language 107**

**LateEndDateDataFieldIndex**

Property of  
VcScheduler 950

**LateStartDateDataFieldIndex**

Property of  
VcScheduler 951

**Layer 99**

3D effect 178  
by index 737  
duration 176  
end date field 176  
height 177  
layer shape 175  
order 733  
see also  
VcLayer 704  
start date field 176

using 39  
visible in legend 173

**LayerByIndex**

Method of  
VcLayerCollection 737

**LayerByName**

Method of  
VcLayerCollection 737

**LayerCollection**

Property of  
VcGantt 526  
see also  
VcLayerCollection 734

**LayerFormat**

see also  
VcLayerFormat 740

**LayerFormatField**

see also  
VcLayerFormatField 743

**LayerName**

Property of  
VcCurve 392

**LayersWithNonWorkInterval**

Property of  
VcGantt 527

**Left**

Property of  
VcRect 873

**LeftMargin**

Property of  
VcTableFormatField 981

**LeftTable**

Property of  
VcGantt 527

**LeftTableDiagramWidthRatio**

Property of  
VcGantt 528

**Legend**

- Arrangement 273, 274
- extended attributes 273
- Font 274
- Title 273

**LegendElementsArrangement**

- Property of
  - VcBoundingBox 304

**LegendElementsBottomMargin**

- Property of
  - VcBoundingBox 305

**LegendElementsMaximumColumnCount**

- Property of
  - VcBoundingBox 305

**LegendElementsMaximumRowCount**

- Property of
  - VcBoundingBox 305

**LegendElementsTopMargin**

- Property of
  - VcBoundingBox 306

**LegendFont**

- Property of
  - VcBoundingBox 306

**LegendText**

- Property of
  - VcLayer 717
  - VcMapEntry 811

**LegendTitle**

- Property of
  - VcBoundingBox 306

**LegendTitleFont**

- Property of
  - VcBoundingBox 307

**LegendTitleVisible**

- Property of
  - VcBoundingBox 307

**Level**

- Property of
  - VcGroupLevelLayout 658

**Licensing 14, 151**

- Request License Information 277

**Line formats**

- administration 190, 192

**Line grid 255****Line Grids**

- administrate 205

**LineColor**

- Property of
  - VcBox 313
  - VcCalendarGrid 367
  - VcCurve 393
  - VcDateLine 468
  - VcDateLineGrid 483
  - VcInterval 691
  - VcLayer 717
  - VcLinkAppearance 774

**LineColorDataFieldIndex**

- Property of
  - VcCalendarGrid 368
  - VcDateLineGrid 483
  - VcLayer 717

**LineColorMapName**

- Property of
  - VcCalendarGrid 368
  - VcDateLineGrid 483
  - VcLayer 718

**LineFormat**

- see also
  - VcLineFormat 750

**LineFormatCollection**

- Property of
  - VcGantt 528
- see also

VcLineFormatCollection 753

**LineFormatField**

see also

VcLineFormatField 759

**LineThickness**

Property of

VcBox 313

VcCurve 393

VcDateLine 469

VcDateLineGrid 484

VcInterval 691

VcLayer 718

VcLinkAppearance 774

**LineType**

Property of

VcBox 314

VcCalendarGrid 368

VcCurve 394

VcDateLine 470

VcDateLineGrid 485

VcLayer 719

VcLinkAppearance 776

**Link 103**

appearances 102

ID 769

predecessor node 164

see also

VcLink 767

show 222

successor node 164

type 165

**Link appearance**

order 781

**Link appearance collection**

add 783

add by specification 784

copy 784

delete 787

**Link appearance object**

by index 785

by name 786

enumerator object 785

iteration, initial value 784

iteration, subsequent value 786

layer of predecessor 777

layer of successor 779

number in collection 782

predecessor port symbol 778

routing type 778

successor port symbol 780

visible 780

**LinkAppearance**

see also

VcLinkAppearance 773

**LinkAppearanceByIndex**

Method of

VcLinkAppearanceCollection 785

**LinkAppearanceByName**

Method of

VcLinkAppearanceCollection 786

**LinkAppearanceCollection**

Property of

VcGantt 529

see also

VcLinkAppearanceCollection 782

**LinkCollection**

Property of

VcGantt 529

see also

VcLinkCollection 789

**LinkDataTableName**

Property of

VcResourceScheduler2 889

**LinkDurationDataFieldIndex**

Property of  
 VcScheduler 951

**LinkDurationFieldIndex**  
 Property of  
 VcResourceScheduler2 891

**LinkPredecessorDataFieldIndex**  
 Property of  
 VcGantt 529

**LinkPredecessorOperationIDFieldIndex**  
**x**  
 Property of  
 VcResourceScheduler2 891

**LinkPredecessorTaskIDFieldIndex**  
 Property of  
 VcResourceScheduler2 892

**Links**  
 Administrage Link Appearances 222  
 rounded slants 151

**LinksDataTableName**  
 Property of  
 VcGantt 531

**LinkSuccessorDataFieldIndex**  
 Property of  
 VcGantt 532

**LinkSuccessorOperationIDFieldIndex**  
 Property of  
 VcResourceScheduler2 892

**LinkSuccessorTaskIDFieldIndex**  
 Property of  
 VcResourceScheduler2 893

**LinkTypeDataFieldIndex**  
 Property of  
 VcGantt 533

**Load**  
 Method of  
 VcGantt 572

## M

### **MajorTicks**

Property of  
 VcNumericScale 841  
 VcRibbon 941

### **MajorTicksEx**

Property of  
 VcNumericScale 842

### **Map 109**

by index 802  
 see also  
 VcMap 793

### **MapByIndex**

Method of  
 VcMapCollection 802

### **MapByName**

Method of  
 VcMapCollection 802

### **MapCollection**

Property of  
 VcGantt 534  
 see also  
 VcMapCollection 799

### **MapEntry**

see also  
 VcMapEntry 806

### **Maps**

Specifying value ranges by using  
 filters 793

### **Marked**

Property of  
 VcBox 315  
 VcCurve 396  
 VcGroup 642  
 VcNode 819

### **MarkedNodesFilter**

## 1022 Index

- Property of
  - VcFilterCollection 496
- MaxHorizontalPagesCount**
  - Property of
    - VcPrinter 862
- MaximumTextLineCount**
  - Property of
    - VcBoxFormatField 343
    - VcTableFormatField 982
- MaxVerticalPagesCount**
  - Property of
    - VcPrinter 863
- Methods**
  - Add
    - VcBoxCollection 322
    - VcBoxFormatCollection 334
    - VcCalendarCollection 360
    - VcCurveCollection 412
    - VcDataRecordCollection 438
    - VcDataTableCollection 449
    - VcDataTableFieldCollection 462
    - VcFilterCollection 496
    - VcGroupLevelLayoutCollection 670
    - VcIntervalCollection 700
    - VcLayerCollection 735
    - VcLineFormatCollection 754
    - VcLinkAppearanceCollection 783
    - VcMapCollection 800
  - AddBySpecification
    - VcBoxCollection 322
    - VcBoxFormatCollection 334
    - VcCalendarCollection 360
    - VcCurveCollection 413
    - VcFilterCollection 497
    - VcGroupLevelLayoutCollection 670
  - VcIntervalCollection 700
  - VcLayerCollection 735
  - VcLineFormatCollection 754
  - VcLinkAppearanceCollection 784
  - VcMapCollection 800
  - AddDuration
    - VcCalendar 352
  - AddSubCondition
    - VcFilter 492
  - BorderBox
    - VcBorderArea 301
  - BoxByIndex
    - VcBoxCollection 323
  - BoxByName
    - VcBoxCollection 324
  - CalcDuration
    - VcCalendar 352
  - CalculateCurrentWidth
    - VcLayer 733
  - CalculateLineCount
    - VcLayerFormatField 749
  - CalendarByIndex
    - VcCalendarCollection 361
  - CalendarByName
    - VcCalendarCollection 361
  - Clear
    - VcCurve 402
  - ConvertDistance
    - VcGantt 554
  - Copy
    - VcBoxCollection 324
    - VcBoxFormatCollection 335
    - VcCalendarCollection 361
    - VcCurveCollection 413
    - VcDataTableCollection 449
    - VcDataTableFieldCollection 462
    - VcFilterCollection 497

- VcGroupLevelLayoutCollection
  - 671
- VcIntervalCollection 701
- VcLayerCollection 736
- VcLineFormatCollection 755
- VcLinkAppearanceCollection 784
- VcMapCollection 801
- CopyFormatField
  - VcBoxFormat 331
  - VcLayerFormat 741
  - VcLineFormat 752
- CopySubCondition
  - VcFilter 493
- CreateDataDefinitionField
  - VcDataDefinitionTable 426
- CreateEntry
  - VcMap 796
- CurveByIndex
  - VcCurveCollection 414
- CurveByName
  - VcCurveCollection 414
- DataDefinitionFieldByIndex
  - VcDataDefinitionTable 427
- DataDefinitionFieldByName
  - VcDataDefinitionTable 427
- DataRecord
  - VcGroup 647
  - VcLink 770
  - VcNode 821
- DataRecordById
  - VcDataRecordCollection 439
- DataTableByIndex
  - VcDataTableCollection 450
- DataTableByName
  - VcDataTableCollection 451
- DataTableFieldByIndex
  - VcDataTableFieldCollection 463
- DataTableFieldByName
  - VcDataTableFieldCollection 464
- DateLineByIndex
  - VcDateLineCollection 476
- DateLineByName
  - VcDateLineCollection 476
- Delete
  - VcDataRecord 434
  - VcGroup 647
  - VcHistogramCollection 682
  - VcLink 770
  - VcNode 821
- DeleteEntry
  - VcMap 797
- DeleteLinkRecord
  - VcGantt 554
- DeleteNodeRecord
  - VcGantt 555
- DeletePoint
  - VcCurve 402
- DetectDataTableFieldName
  - VcGantt 555
- DetectDataTableName
  - VcGantt 555
- DetectFieldIndex
  - VcGantt 556
- DetermineIDOfFirstOperationByTaskID
  - VcResourceScheduler2 934
- DetermineIDOfLastOperationByTaskID
  - VcResourceScheduler2 935
- EndLoading
  - VcGantt 557
- ExportGraphicsToFileEx
  - VcGantt 557
- FilterByIndex



- VcFilterCollection 498
- FilterByName
  - VcFilterCollection 498
- FirstBox
  - VcBoxCollection 325
- FirstCalendar
  - VcCalendarCollection 362
- FirstCurve
  - VcCurveCollection 415
- FirstDataDefinitionField
  - VcDataDefinitionTable 428
- FirstDataRecord
  - VcDataRecordCollection 440
- FirstDataTable
  - VcDataTableCollection 451
- FirstDataTableField
  - VcDataTableFieldCollection 464
- FirstDateLine
  - VcDateLineCollection 477
- FirstFilter
  - VcFilterCollection 498
- FirstFormat
  - VcBoxFormatCollection 335
  - VcLineFormatCollection 755
  - VcTableFormatCollection 973
- FirstGroup
  - VcGroupCollection 651
- FirstGroupLevelLayout
  - VcGroupLevelLayoutCollection 671
- FirstHistogram
  - VcHistogramCollection 683
- FirstInterval
  - VcIntervalCollection 701
- FirstLayer
  - VcLayerCollection 736
- FirstLink
  - VcLinkCollection 790
- FirstLinkAppearance
  - VcLinkAppearanceCollection 784
- FirstMap
  - VcMapCollection 801
- FirstMapEntry
  - VcMap 797
- FirstNode
  - VcNodeCollection 828
- FirstNumericScale
  - VcNumericScaleCollection 849
- FirstTimeScale
  - VcTimeScaleCollection 994
- FitChartIntoView
  - VcGantt 559
- FitHistogramsIntoView
  - VcGantt 560
- FitRangeIntoView
  - VcGantt 560
  - VcHistogram 677
- FormatByIndex
  - VcBoxFormatCollection 336
  - VcLineFormatCollection 756
  - VcTableFormatCollection 973
- FormatByName
  - VcBoxFormatCollection 336
  - VcLineFormatCollection 756
  - VcTableFormatCollection 974
- GetCurrentViewDates
  - VcGantt 561
- GetCurrentYValues
  - VcHistogram 677
- GetDate
  - VcGantt 561
- GetEndOfPreviousWorktime
  - VcCalendar 353
- GetEnumerator

- VcBoxCollection 325
- VcBoxFormat 331
- VcBoxFormatCollection 337
- VcCalendarCollection 362
- VcCurveCollection 415
- VcDataDefinitionTable 429
- VcDataRecordCollection 441
- VcDataTableCollection 452
- VcDataTableFieldCollection 465
- VcDateLineCollection 478
- VcFilter 493
- VcFilterCollection 499
- VcFilterSubCondition 505
- VcGroupCollection 651
- VcGroupLevelLayoutCollection 671
- VcHistogramCollection 683
- VcLayerCollection 737
- VcLayerFormat 741
- VcLinkAppearanceCollection 785
- VcLinkCollection 790
- VcMapCollection 802
- VcNodeCollection 828
- VcTableFormat 971
- VcTableFormatCollection 974
- VcTimeScaleCollection 994
- GetFirstOverload
  - VcCurve 403
- GetFirstOverloadEx
  - VcCurve 404
- GetLicenseInformation
  - VcGantt 562
- GetLinkByID
  - VcGantt 562
- GetLinkByNodeIDs
  - VcGantt 563
- GetNewUniqueID
  - VcDataRecordCollection 442
- GetNextIntervalBorder
  - VcCalendar 354
- GetNextOverload
  - VcCurve 405
- GetNextOverloadEx
  - VcCurve 406
- GetNodeByID
  - VcGantt 563
- GetPositionInView
  - VcNode 821
- GetPreviousIntervalBorder
  - VcCalendar 354
- GetStartOfInterval
  - VcCalendar 355
- GetStartOfNextWorktime
  - VcCalendar 355
- GetValues
  - VcCurve 407
- GetValuesEx
  - VcCurve 408
- GetViewComponentSize
  - VcGantt 564
- GetXOffset
  - VcBox 319
- GroupByName
  - VcGroupCollection 652
- GroupLevelLayoutByIndex
  - VcGroupLevelLayoutCollection 672
- GroupLevelLayoutByName
  - VcGroupLevelLayoutCollection 672
- GroupNodes
  - VcGantt 565
- HistogramByIndex
  - VcHistogramCollection 684

- HistogramByName
  - VcHistogramCollection 684
- IdentifyField
  - VcGantt 566
- IdentifyLayerAt
  - VcGantt 567
- IdentifyObjectAt
  - VcGantt 568
- ImportConfiguration
  - VcGantt 570
- InsertLinkRecord
  - VcGantt 571
- InsertNodeRecord
  - VcGantt 571
- IntervalByIndex
  - VcIntervalCollection 702
- IntervalByName
  - VcIntervalCollection 702
- IsValid
  - VcFilter 494
  - VcFilterSubCondition 505
- IsWorktime
  - VcCalendar 356
- LayerByIndex
  - VcLayerCollection 737
- LayerByName
  - VcLayerCollection 737
- LinkAppearanceByIndex
  - VcLinkAppearanceCollection 785
- LinkAppearanceByName
  - VcLinkAppearanceCollection 786
- Load
  - VcGantt 572
- MapByIndex
  - VcMapCollection 802
- MapByName
  - VcMapCollection 802
- NextBox
  - VcBoxCollection 326
- NextCalendar
  - VcCalendarCollection 363
- NextCurve
  - VcCurveCollection 416
- NextDataDefinitionField
  - VcDataDefinitionTable 429
- NextDataRecord
  - VcDataRecordCollection 442
- NextDataTable
  - VcDataTableCollection 452
- NextDataTableField
  - VcDataTableFieldCollection 465
- NextDateLine
  - VcDateLineCollection 478
- NextFilter
  - VcFilterCollection 499
- NextFormat
  - VcBoxFormatCollection 337
  - VcLineFormatCollection 757
  - VcTableFormatCollection 975
- NextGroup
  - VcGroupCollection 652
- NextGroupLevelLayout
  - VcGroupLevelLayoutCollection 672
- NextHistogram
  - VcHistogramCollection 684
- NextInterval
  - VcIntervalCollection 702
- NextLayer
  - VcLayerCollection 738
- NextLink
  - VcLinkCollection 790
- NextLinkAppearance
  - VcLinkAppearanceCollection 786

- NextMap
  - VcMapCollection 803
- NextMapEntry
  - VcMap 798
- NextNode
  - VcNodeCollection 828
- NextNumericScale
  - VcNumericScaleCollection 850
- NextTimeScale
  - VcTimeScaleCollection 995
- NodeRowInView
  - VcNode 822
- NumericScaleByIndex
  - VcNumericScaleCollection 851
- NumericScaleByName
  - VcNumericScaleCollection 851
- OptimizeColumnWidth
  - VcTable 963
- OptimizeTimeScaleStartEnd
  - VcGantt 573
- OutlineIndent
  - VcNode 823
- OutlineOutdent
  - VcNode 823
- PrintEx
  - VcGantt 573
- PrintToFile
  - VcGantt 574
- Process
  - VcResourceScheduler2 935
- PutInOrderAfter
  - VcCalendarProfile 377
  - VcDateLine 474
  - VcHistogram 678
  - VcLayer 733
  - VcLinkAppearance 781
- RecalculateAllStructureCodes
  - VcGantt 575
- RelatedDataRecord
  - VcDataRecord 435
  - VcGroup 648
  - VcLink 771
  - VcNode 824
- Remove
  - VcBoxCollection 326
  - VcBoxFormatCollection 338
  - VcCalendarCollection 363
  - VcCurveCollection 417
  - VcDataRecordCollection 443
  - VcFilterCollection 500
  - VcGroupLevelLayoutCollection 673
  - VcIntervalCollection 703
  - VcLayerCollection 738
  - VcLineFormatCollection 757
  - VcLinkAppearanceCollection 787
  - VcMapCollection 804
- RemoveFormatField
  - VcBoxFormat 332
  - VcLayerFormat 742
  - VcLineFormat 752
- RemoveSubCondition
  - VcFilter 494
- ReOptimizeNodes
  - VcGroup 648
- Reset
  - VcGantt 575
- SaveAsEx
  - VcGantt 576
- ScheduleProject
  - VcScheduler 953
- ScrollToDate
  - VcGantt 576
- ScrollToGroupLine

- VcGantt 577
- ScrollToNode
  - VcGantt 578
- ScrollToNodeLine
  - VcGantt 579
- ScrollToValue
  - VcHistogram 678
- SelectGroups
  - VcGroupCollection 653
- SelectLinks
  - VcLinkCollection 791
- SelectMaps
  - VcMapCollection 804
- SelectNodes
  - VcNodeCollection 829
- SetMaxYValue
  - VcHistogram 679
- SetMinYValue
  - VcHistogram 679
- SetPositionInView
  - VcNode 824
- SetValues
  - VcCurve 409
- SetXYOffset
  - VcBox 320
- SetXYOffsetByTopLeftPixel
  - VcBox 320
- SortGroups
  - VcGantt 580
- SortNodes
  - VcGantt 580
- SuspendUpdate
  - VcGantt 581
- TableByIndex
  - VcTableCollection 964
- TimeScaleByIndex
  - VcTimeScaleCollection 995
- TimeScaleByName
  - VcTimeScaleCollection 996
- Update
  - VcBoxCollection 327
  - VcCalendar 356
  - VcCalendarCollection 364
  - VcDataRecord 436
  - VcDataRecordCollection 444
  - VcDataTableCollection 453
  - VcGroup 648
  - VcGroupLevelLayoutCollection 673
  - VcIntervalCollection 703
  - VcLayerCollection 739
  - VcLink 771
  - VcLinkAppearanceCollection 787
  - VcMapCollection 805
  - VcNode 825
- UpdateLinkRecord
  - VcGantt 582
- UpdateNodeRecord
  - VcGantt 583
- UpdateRowNumberFields
  - VcGantt 583
- Zoom
  - VcGantt 584
- Millimeter**
  - Property of
    - VcMapEntry 811
- MinimumRowHeight**
  - Property of
    - VcGantt 534
- MinimumTextLineCount**
  - Property of
    - VcBoxFormatField 344
    - VcTableFormatField 982
- MinimumWidth**

- Property of
    - VcBoxFormatField 345
    - VcLayerFormatField 745
  - MinorTicks**
    - Property of
      - VcNumericScale 842
      - VcRibbon 942
  - MinorTicksEx**
    - Property of
      - VcNumericScale 843
  - ModificationsAllowed**
    - Property of
      - VcGroupLevelLayout 658
  - Movable**
    - Property of
      - VcLayer 720
  - MovingLayersAsNodeWithShiftKeyAllowed**
    - Property of
      - VcGantt 535
  - MultiplePrimaryKeysAllowed**
    - Property of
      - VcDataTable 447
  - MultiState**
    - Property of
      - VcTableFormatField 983
  - MultiState fields 114**
- N**
- Name**
    - Property of
      - VcBox 316
      - VcBoxFormat 330
      - VcCalendar 350
      - VcCalendarGrid 369
      - VcCalendarProfile 376
      - VcCurve 396
      - VcDataDefinitionField 423
      - VcDataTable 447
      - VcDataTableField 457
      - VcDateLine 471
      - VcFilter 490
      - VcGroup 642
      - VcGroupLevelLayout 659
      - VcHistogram 675
      - VcLayer 720
      - VcLineFormat 751
      - VcLinkAppearance 777
      - VcMap 794
      - VcNumericScale 843
      - VcTableFormat 969
      - VcTimeScale 990
  - NextBox**
    - Method of
      - VcBoxCollection 326
  - NextCalendar**
    - Method of
      - VcCalendarCollection 363
  - NextCurve**
    - Method of
      - VcCurveCollection 416
  - NextDataDefinitionField**
    - Method of
      - VcDataDefinitionTable 429
  - NextDataRecord**
    - Method of
      - VcDataRecordCollection 442
  - NextDataTable**
    - Method of
      - VcDataTableCollection 452
  - NextDataTableField**
    - Method of
      - VcDataTableFieldCollection 465
  - NextDateLine**

- Method of
  - VcDateLineCollection 478
- NextFilter**
  - Method of
    - VcFilterCollection 499
- NextFormat**
  - Method of
    - VcBoxFormatCollection 337
    - VcLineFormatCollection 757
    - VcTableFormatCollection 975
- NextGroup**
  - Method of
    - VcGroupCollection 652
- NextGroupLevelLayout**
  - Method of
    - VcGroupLevelLayoutCollection 672
- NextHistogram**
  - Method of
    - VcHistogramCollection 684
- NextInterval**
  - Method of
    - VcIntervalCollection 702
- NextLayer**
  - Method of
    - VcLayerCollection 738
- NextLink**
  - Method of
    - VcLinkCollection 790
- NextLinkAppearance**
  - Method of
    - VcLinkAppearanceCollection 786
- NextMap**
  - Method of
    - VcMapCollection 803
- NextMapEntry**
  - Method of
    - VcMap 798
- NextNode**
  - Method of
    - VcNodeCollection 828
- NextNumericScale**
  - Method of
    - VcNumericScaleCollection 850
- NextTimeScale**
  - Method of
    - VcTimeScaleCollection 995
- Node 116**
  - assign calendars to nodes 156
  - ID 818
  - marking type in diagram 156
  - marking type in table 155
  - see also
    - VcNode 816
- Node rows**
  - initial number 158
  - minimal height 158
- NodeCalendarNameDataFieldIndex**
  - Property of
    - VcGantt 535
- NodeCollection**
  - Property of
    - VcGantt 536
    - VcGroup 643
  - see also
    - VcNodeCollection 827
- NodeDurationDataFieldIndex**
  - Property of
    - VcGantt 536
- NodeEndDateDataFieldIndex**
  - Property of
    - VcGantt 536
- NodeLevelLayout**
  - see also

- VcNodeLevelLayout 831
- NodeRowInView**
  - Method of
    - VcNode 822
- NodeRowNumberDataFieldIndex**
  - Property of
    - VcGantt 537
- Nodes**
  - all nodes of one group of this level in one line 655
  - data record 647, 770, 821
  - groups in one line 199
  - move all layers with shift key 535
  - move vertically 552
  - node layout optimized 659
  - optimized arrangement 523
  - related data record 648, 771, 824
- NodesAndGroupsBelowCollapsed**
  - Property of
    - VcGroup 643
- NodesArrangedInOneRow**
  - Property of
    - VcGroup 644
- NodesArrangedOptimized**
  - Property of
    - VcGroupLevelLayout 659
- NodesDataTableName**
  - Property of
    - VcGantt 538
- NodesOptimized**
  - Property of
    - VcGroup 644
- NodeSortingDataFieldIndex**
  - Property of
    - VcGantt 538
- NodeSortingOrder**
  - Property of
    - VcGantt 539
- NodeStartDateDataFieldIndex**
  - Property of
    - VcGantt 540
- NodesUseCalendars**
  - Property of
    - VcGantt 540
- Non Working Intervals**
  - mark 33
- NonWorkIntervalsCollapsed**
  - Property of
    - VcSection 956
- Number**
  - Property of
    - VcMapEntry 812
- Numeric scale**
  - 3D effect 845
  - background color 839
  - by index 851
  - font body 840
  - font color 840
  - histogram associated 841
  - major ticks 841, 842
  - maximum value 679
  - minimum value 679
  - minor ticks 842, 843
  - name 843
  - pattern 844
  - title 845
  - unit 846
  - unit label 846
  - unit width 846
- NumericScale**
  - see also
    - VcNumericScale 839
- NumericScaleByIndex**
  - Method of
    - VcGantt 539



VcNumericScaleCollection 851

**NumericScaleByName**

Method of

VcNumericScaleCollection 851

**NumericScaleCollection**

Property of

VcGantt 540

VcHistogram 676

see also

VcNumericScaleCollection 848

VcDataTableCollection 448

VcDataTableField 454

VcDataTableFieldCollection 461

VcDateLine 467

VcDateLineCollection 475

VcDateLineGrid 480

VcFilter 489

VcFilterCollection 495

VcFilterSubCondition 501

VcGantt 506

VcGroup 639

VcGroupCollection 650

VcGroupLevelLayout 654

VcGroupLevelLayoutCollection 669

VcHistogram 674

VcHistogramCollection 681

VcInterval 686

VcIntervalCollection 699

VcLayer 704

VcLayerCollection 734

VcLayerFormat 740

VcLayerFormatField 743

VcLineFormat 750

VcLineFormatCollection 753

VcLineFormatField 759

VcLink 767

VcLinkAppearance 773

VcLinkAppearanceCollection 782

VcLinkCollection 789

VcMap 793

VcMapCollection 799

VcMapEntry 806

VcNode 816

VcNodeCollection 827

VcNodeLevelLayout 831

VcNumericScale 839

VcNumericScaleCollection 848



**ObjectDraw events**

enabled 721

**ObjectDrawEventsEnabled**

Property of

VcLayer 721

**Objects**

VcBorderArea 301

VcBorderBox 303

VcBox 311

VcBoxCollection 321

VcBoxFormat 328

VcBoxFormatCollection 333

VcBoxFormatField 340

VcCalendar 349

VcCalendarCollection 358

VcCalendarGrid 365

VcCalendarProfile 376

VcCurve 379

VcCurveCollection 411

VcDataDefinition 418

VcDataDefinitionField 420

VcDataDefinitionTable 425

VcDataRecord 431

VcDataRecordCollection 437

VcDataTable 445

- VcPrinter 853
- VcRect 872
- VcResourceScheduler2 876
- VcRibbon 937
- VcScheduler 947
- VcSection 955
- VcTable 960
- VcTableCollection 964
- VcTableFormat 965
- VcTableFormatCollection 972
- VcTableFormatField 976
- VcTimeScale 988
- VcTimeScaleCollection 993
- OperationDataTableName**
  - Property of
    - VcResourceScheduler2 894
- OperationLoadPerItemFieldIndex**
  - Property of
    - VcResourceScheduler2 895
- OperationMaximumInterruptionTimeFieldIndex**
  - Property of
    - VcResourceScheduler2 895
- OperationMinimumSupplementTimeFieldIndex**
  - Property of
    - VcResourceScheduler2 896
- OperationOverlapQuantityFieldIndex**
  - Property of
    - VcResourceScheduler2 897
- OperationPostLoadFieldIndex**
  - Property of
    - VcResourceScheduler2 899
- OperationPreparationLoadFieldIndex**
  - Property of
    - VcResourceScheduler2 899
- OperationResultEndDateFieldIndex**
  - Property of
    - VcResourceScheduler2 900
- OperationResultPostEndDateFieldIndex**
  - Property of
    - VcResourceScheduler2 901
- OperationResultPreparationStartDateFieldIndex**
  - Property of
    - VcResourceScheduler2 901
- OperationResultProcessingTimeFieldIndex**
  - Property of
    - VcResourceScheduler2 902
- OperationResultStartDateFieldIndex**
  - Property of
    - VcResourceScheduler2 902
- OperationResultStatusFieldIndex**
  - Property of
    - VcResourceScheduler2 903
- OperationRouteFieldIndex**
  - Property of
    - VcResourceScheduler2 904
- OperationSequenceNumberFieldIndex**
  - Property of
    - VcResourceScheduler2 904
- OperationStartLockDateFieldIndex**
  - Property of
    - VcResourceScheduler2 905
- OperationTaskIDFieldIndex**
  - Property of
    - VcResourceScheduler2 906
- OperationWorkInProgressFieldIndex**
  - Property of
    - VcResourceScheduler2 906
- Operator**
  - Property of
    - VcFilterSubCondition 504

**OptimizeColumnWidth**

Method of  
VcTable 963

**OptimizedNodesSortDataFieldIndex**

Property of  
VcGroupLevelLayout 659

**OptimizedNodesSortOrder**

Property of  
VcGroupLevelLayout 660

**OptimizeTimeScaleStartEnd**

Method of  
VcGantt 573

**Orientation**

Property of  
VcPrinter 863

**Origin**

Property of  
VcBox 316

**OutgoingLinks**

Property of  
VcNode 819, 820

**OutlineIndent**

Method of  
VcNode 823

**OutlineOutdent**

Method of  
VcNode 823

**OverlaidNodesSortDataFieldIndex**

Property of  
VcGroupLevelLayout 660

**OverlaidNodesSortOrder**

Property of  
VcGroupLevelLayout 660

**OverlapLayerEnabled**

Property of  
VcGantt 541

**OverlapLayerName**

Property of  
VcGantt 541

**P****Page numbers 865****PageDescription**

Property of  
VcPrinter 864

**PageDescriptionString**

Property of  
VcPrinter 864

**PageFrame**

Property of  
VcPrinter 865

**PageNumberMode**

Property of  
VcPrinter 865

**PageNumbers**

Property of  
VcPrinter 866

**PagePaddingEnabled**

Property of  
VcPrinter 866

**PaperSize**

Property of  
VcPrinter 867

**Pattern**

Property of  
VcBoxFormatField 345  
VcCalendarGrid 369  
VcInterval 692  
VcLayer 721  
VcLineFormatField 763  
VcMapEntry 812  
VcNumericScale 844  
VcRibbon 942  
VcTableFormatField 983

**PatternColor**

- Property of
  - VcCalendarGrid 372
  - VcInterval 695
  - VcLayer 724

**PatternColorDataFieldIndex**

- Property of
  - VcCalendarGrid 373
  - VcLayer 724

**PatternColorMapName**

- Property of
  - VcCalendarGrid 373
  - VcLayer 725

**PatternDataFieldIndex**

- Property of
  - VcCalendarGrid 373
  - VcLayer 725

**PatternMapName**

- Property of
  - VcCalendarGrid 374
  - VcLayer 726

**PDF Files**

- Export 142

**performance 288****Period**

- Property of
  - VcDateLineGrid 486

**PlanningEndDate**

- Property of
  - VcResourceScheduler2 907

**PlanningStartDate**

- Property of
  - VcResourceScheduler2 908

**PlanningStrategy**

- Property of
  - VcResourceScheduler2 909

**PointsEquidistant**

- Property of
  - VcCurve 397

**Position**

- Property of
  - VcRibbon 943
  - VcTable 961

**PredecessorLayerName**

- Property of
  - VcLinkAppearance 777

**PredecessorNode**

- Property of
  - VcLink 769

**PredecessorPortSymbol**

- Property of
  - VcLinkAppearance 778

**Primary key**

- composite 447

**PrimaryKey**

- Property of
  - VcDataTableField 458

**PrintDate**

- Property of
  - VcPrinter 867

**Printer**

- Property of
  - VcGantt 541
- see also
  - VcPrinter 853

**PrinterName**

- Property of
  - VcPrinter 868

**PrintEx**

- Method of
  - VcGantt 573

**Printing**

- actual zoom factor 858

- automatic re-optimization of groups 869
- current printer 858
- diagram 859
- folding marks 861
- into file 574
- mode of page numbering 865
- number of table columns 870
- scaling mode 869
- PrintPreviewWithFirstPage**
  - Property of
    - VcPrinter 868
- PrintToFile**
  - Method of
    - VcGantt 574
- Priority**
  - boxes 214
  - Property of
    - VcBox 317
    - VcCalendarGrid 374
    - VcDateLine 472
    - VcDateLineGrid 486
    - VcLayerFormatField 745
- Process**
  - Method of
    - VcResourceScheduler2 935
- Project end 146**
- Project start 146**
- Properties**
  - AbsoluteBottomMarginInCM
    - VcPrinter 854
  - AbsoluteLeftMarginInCM
    - VcPrinter 854
  - AbsoluteRightMarginInCM
    - VcPrinter 855
  - AbsoluteTopMarginInCM
    - VcPrinter 855
- Active
  - VcCalendarCollection 358
  - VcHistogramCollection 681
  - VcNumericScaleCollection 848
- ActiveNodeFilter
  - VcGantt 510
- ActualEndDateDataFieldIndex
  - VcScheduler 948
- ActualStartDateDataFieldIndex
  - VcScheduler 948
- Addend
  - VcCurve 380
- AdjustToReferenceDate
  - VcDateLineGrid 481
- AjaxExtensionsEnabled
  - VcGantt 511
- Alignment
  - VcBoxFormatField 340
  - VcLayerFormatField 743
  - VcLineFormatField 759
  - VcPrinter 856, 857
  - VcTableFormatField 977
- AllData
  - VcDataRecord 431
  - VcLink 767
  - VcNode 817
- AllNodesInOneRow
  - VcGroupLevelLayout 655
- AlwaysCurrentDate
  - VcDateLine 467
- AnnotationAtBottom
  - VcDateLineGrid 481
- AnnotationAtCenter
  - VcDateLineGrid 481
- AnnotationAtTop
  - VcDateLineGrid 482
- Arrangement

- VcGantt 511
- AssignmentDataTableName
  - VcResourceScheduler2 879
- AssignmentIsResultFieldIndex
  - VcResourceScheduler2 881
- AssignmentIsVisibleFieldIndex
  - VcResourceScheduler2 881
- AssignmentLoadOrConsumptionPerItemFieldIndex
  - VcResourceScheduler2 882
- AssignmentMaximumLoadFieldIndex
  - VcResourceScheduler2 883
- AssignmentMinimumLoadFieldIndex
  - VcResourceScheduler2 883
- AssignmentOperationIDFieldIndex
  - VcResourceScheduler2 884
- AssignmentResourceIDFieldIndex
  - VcResourceScheduler2 885
- AssignmentResourceSelectionStrategyFieldIndex
  - VcResourceScheduler2 885
- AutomaticSchedulingEnabled
  - VcScheduler 948
- BackgroundColor
  - VcBoxFormatField 341
  - VcCalendarGrid 366
  - VcInterval 687
  - VcLayer 705
  - VcLineFormatField 760
  - VcNumericScale 839
  - VcRibbon 938
  - VcTableFormatField 977
  - VcTimeScale 988
- BackgroundColorDataFieldIndex
  - VcCalendarGrid 366
  - VcLayer 706
  - VcLineFormatField 760
- VcTableFormatField 978
- BackgroundColorMapName
  - VcCalendarGrid 367
  - VcLayer 707
  - VcLineFormatField 761
  - VcTableFormatField 978
- BaseTimeUnit
  - VcResourceScheduler2 886
- BaseTimeUnitsPerStep
  - VcResourceScheduler2 887
- BorderArea
  - VcGantt 512
- Bottom
  - VcRect 872
- BottomMargin
  - VcTableFormatField 978
- BoxCollection
  - VcGantt 512
- BoxFormatCollection
  - VcGantt 513
- CalendarCollection
  - VcGantt 513
- CalendarGrid
  - VcSection 955
- CalendarGridCollection
  - VcGantt 513
- CalendarGridName
  - VcGroupLevelLayout 655
  - VcNodeLevelLayout 831
- CalendarGridsVisible
  - VcGroupLevelLayout 656
  - VcHistogram 674
  - VcNodeLevelLayout 832
  - VcTimeScale 989
- CalendarGridsWithChildGroups
  - VcGroupLevelLayout 656
- CalendarName

- VcCalendarGrid 367
- VcHistogram 675
- VcRibbon 938
- CalendarNameDataFieldIndex
  - VcGroupLevelLayout 656
- CalendarProfileCollection
  - VcCalendar 350
  - VcGantt 514
- CalendarProfileName
  - VcInterval 688
- CollapseColumn
  - VcTableFormat 966
- Collapsed
  - VcGroupLevelLayout 656
- Color
  - VcMapEntry 806
- ColumnTitle
  - VcTable 960
- ColumnWidth
  - VcTable 961
- ComparisonValueAsString
  - VcFilterSubCondition 501
- CompletionDataFieldIndex
  - VcLayer 709
- ConnectionOperator
  - VcFilterSubCondition 502
- ConsiderFilterEntries
  - VcMap 793
- ConstantText
  - VcLayerFormatField 744
  - VcLineFormatField 761
  - VcTableFormatField 979
- Count
  - VcBoxCollection 321
  - VcBoxFormatCollection 333
  - VcCalendarCollection 359
  - VcCurveCollection 411
  - VcDataDefinitionTable 425
  - VcDataRecordCollection 437
  - VcDataTableCollection 448
  - VcDataTableFieldCollection 461
  - VcDateLineCollection 475
  - VcFilterCollection 495
  - VcGroupCollection 650
  - VcGroupLevelLayoutCollection 669
  - VcHistogramCollection 682
  - VcIntervalCollection 700
  - VcLayerCollection 734
  - VcLineFormatCollection 753
  - VcLinkAppearanceCollection 782
  - VcLinkCollection 789
  - VcMap 794
  - VcMapCollection 799
  - VcNodeCollection 827
  - VcNumericScaleCollection 849
  - VcTableFormatCollection 972
  - VcTimeScaleCollection 993
- CurrentHorizontalPagesCount
  - VcPrinter 857
- CurrentZoomFactor
  - VcPrinter 858
- CurveCollection
  - VcHistogram 675
- CuttingMarks
  - VcPrinter 858
- DataDefinition
  - VcGantt 514
- DataDefinitionTable
  - VcDataDefinition 418
  - VcFilter 490
- DataField
  - VcDataRecord 432
  - VcGroup 640

- VcLink 768
- VcNode 817
- DataFieldIndex
  - VcFilterSubCondition 502
- DataFieldValue
  - VcMapEntry 807
- DataRecordCollection
  - VcDataTable 445
- DataRecordEventsEnabled
  - VcResourceScheduler2 887
- DataTableCollection
  - VcGantt 515
- DataTableFieldCollection
  - VcDataTable 446
- DataTableName
  - VcDataRecord 433
  - VcDataTableField 454
- Date
  - VcDateLine 468
- DateFormat
  - VcDataDefinitionField 420
  - VcDataTableField 455
- DateGridsVisible
  - VcTimeScale 989
- DateLineCollection
  - VcGantt 515
- DateLineGrid
  - VcSection 956
- DateLineGridName
  - VcGroupLevelLayout 657
- DateLineGridsVisible
  - VcGroupLevelLayout 657
- DateLineGridsWithChildGroups
  - VcGroupLevelLayout 657
- DateOutputFormat
  - VcGantt 515
  - VcLineFormatField 761
- VcRibbon 938
- DatesWithHourAndMinute
  - VcFilter 490
- DayInEndMonth
  - VcInterval 688
- DayInStartMonth
  - VcInterval 688
- DefaultOperationMaximumInterruptionTime
  - VcResourceScheduler2 888
- DefaultPrinterName
  - VcPrinter 858
- DefaultResourceCalendarName
  - VcResourceScheduler2 888
- Description
  - VcDataTable 446
- DiagramAlternatingRowBackgroundColor
  - VcGantt 517
- DiagramBackgroundColor
  - VcGantt 517
- DiagramEnabled
  - VcPrinter 859
- DiagramHistogramHeightRatio
  - VcGantt 518
- DocumentName
  - VcPrinter 859
- Duration
  - VcInterval 689
- DurationDataFieldIndex
  - VcLayer 709
  - VcScheduler 949
- EarlyEndDateDataFieldIndex
  - VcScheduler 949
- EarlyStartDateDataFieldIndex
  - VcScheduler 949
- Editable



- VcDataDefinitionField 421
- VcDataTableField 456
- Enabled
  - VcGantt 518
- EndDataFieldIndex
  - VcLayer 709
- EndDateForAutomaticScheduling
  - VcGantt 519
  - VcScheduler 949
- EndDateNotLaterThanDataFieldIndex
  - VcScheduler 950
- EndTime
  - VcInterval 689
- EndMonth
  - VcInterval 689
- EndTime
  - VcInterval 690
- EndWeekday
  - VcInterval 690
- ExtendedDataTablesEnabled
  - VcGantt 519
- FieldsSeparatedByLines
  - VcBoxFormat 328
  - VcTableFormat 966
- FieldText
  - VcBox 311
- FilePath
  - VcGantt 520
- FillReference1BackgroundColor
  - VcCurve 381
- FillReference1Name
  - VcCurve 381
- FillReference1Pattern
  - VcCurve 382
- FillReference1PatternColor
  - VcCurve 385
- FillReference2Color
  - VcCurve 386
- FillReference2Name
  - VcCurve 387
- FillReference2Pattern
  - VcCurve 387
- FillReference2PatternColor
  - VcCurve 390
- FilterCollection
  - VcGantt 520
- FilterName
  - VcCurve 391
  - VcFilterSubCondition 503
  - VcLayer 710
  - VcLinkAppearance 773
  - VcTableFormat 967
- FitToPage
  - VcPrinter 860
- FoldingMarksType
  - VcPrinter 860
- Font
  - VcNumericScale 840
  - VcRibbon 940
  - VcTimeScale 990
- FontAntiAliasingEnabled
  - VcGantt 521
- FontBody
  - VcMapEntry 808
- FontColor
  - VcNumericScale 840
  - VcRibbon 941
  - VcTimeScale 990
- FontName
  - VcMapEntry 808
- FontSize
  - VcMapEntry 809
- Format
  - VcLayer 710

- FormatField
  - VcBoxFormat 329
  - VcLayerFormat 740
  - VcLineFormat 750
  - VcTableFormat 967
- FormatFieldCount
  - VcBoxFormat 329
  - VcLayerFormat 741
  - VcLineFormat 751
  - VcTableFormat 968
- FormatName
  - VcBox 312
  - VcBoxFormatField 342
  - VcDateLineGrid 482
  - VcLayerFormatField 744
  - VcLineFormatField 763
  - VcTableFormatField 979
- FreeFloatDataFieldIndex
  - VcScheduler 950
- FullUsageOfPlanningUnitsEnabled
  - VcResourceScheduler2 889
- GetEnumerator
  - VcMap 794
- GraphicsFileName
  - VcBoundingBox 303
  - VcLayer 710
  - VcMapEntry 809
  - VcTableFormatField 979
- GraphicsFileNameDataFieldIndex
  - VcLayer 712
  - VcTableFormatField 980
- GraphicsFileNameMapName
  - VcLayer 714
  - VcTableFormatField 981
- GraphicsHeight
  - VcBoxFormatField 342
  - VcTableFormatField 981
- GroupCollection
  - VcGantt 521
- GroupDataFieldIndex
  - VcGroupLevelLayout 658
- GroupingDataFieldIndex
  - VcGantt 521
- GroupingLevel
  - VcGroup 641
- GroupingModificationsAllowed
  - VcGantt 522
- GroupNodesVisible
  - VcGroupLevelLayout 658
- GroupOptimizationOnInteractionsEnabled
  - VcGantt 523
- GroupSortingDataFieldIndex
  - VcGantt 523
- GroupSortingOrder
  - VcGantt 524
- Height
  - VcLayer 715
  - VcRect 872
- HeightDataFieldIndex
  - VcLayer 715
- Hidden
  - VcDataDefinitionField 422
  - VcDataTableField 456
- HierarchyDataFieldIndex
  - VcGantt 524
- Histogram
  - VcCurve 392
  - VcNumericScale 841
- HistogramCollection
  - VcGantt 525
- HistogramSeparationLineColor
  - VcGantt 525
- HorAlignment

- VcDateLineGrid 482
- HorizontalOffset
  - VcLayer 716
- ID
  - VcDataRecord 434
  - VcGroup 641
  - VcLink 769
  - VcNode 818
- IncomingLinks
  - VcNode 818
- IndentColumn
  - VcTableFormat 968
- IndentWidth
  - VcTableFormat 969
- Index
  - VcBoxFormatField 343
  - VcDataDefinitionField 422
  - VcDataTableField 457
  - VcFilterSubCondition 503
  - VcLayerFormatField 744
  - VcLineFormatField 763
  - VcTableFormatField 981
- InitialRowCount
  - VcGantt 526
- IntervalCollection
  - VcCalendar 350
  - VcCalendarProfile 376
- KeepingNodesTogetherDataFieldIndex
  - VcGantt 526
- LabelSizeDependence
  - VcLayer 716
- LateEndDateDataFieldIndex
  - VcScheduler 950
- LateStartDateDataFieldIndex
  - VcScheduler 951
- LayerCollection
  - VcGantt 526
- LayerName
  - VcCurve 392
- LayersWithNonWorkInterval
  - VcGantt 527
- Left
  - VcRect 873
- LeftMargin
  - VcTableFormatField 981
- LeftTable
  - VcGantt 527
- LeftTableDiagramWidthRatio
  - VcGantt 528
- LegendElementsArrangement
  - VcBoundingBox 304
- LegendElementsBottomMargin
  - VcBoundingBox 305
- LegendElementsMaximumColumnCount
  - VcBoundingBox 305
- LegendElementsMaximumRowCount
  - VcBoundingBox 305
- LegendElementsTopMargin
  - VcBoundingBox 306
- LegendFont
  - VcBoundingBox 306
- LegendText
  - VcLayer 717
  - VcMapEntry 811
- LegendTitle
  - VcBoundingBox 306
- LegendTitleFont
  - VcBoundingBox 307
- LegendTitleVisible
  - VcBoundingBox 307
- Level
  - VcGroupLevelLayout 658

- LineColor
  - VcBox 313
  - VcCalendarGrid 367
  - VcCurve 393
  - VcDateLine 468
  - VcDateLineGrid 483
  - VcInterval 691
  - VcLayer 717
  - VcLinkAppearance 774
- LineColorDataFieldIndex
  - VcCalendarGrid 368
  - VcDateLineGrid 483
  - VcLayer 717
- LineColorMapName
  - VcCalendarGrid 368
  - VcDateLineGrid 483
  - VcLayer 718
- LineFormatCollection
  - VcGantt 528
- LineThickness
  - VcBox 313
  - VcCurve 393
  - VcDateLine 469
  - VcDateLineGrid 484
  - VcInterval 691
  - VcLayer 718
  - VcLinkAppearance 774
- LineType
  - VcBox 314
  - VcCalendarGrid 368
  - VcCurve 394
  - VcDateLine 470
  - VcDateLineGrid 485
  - VcLayer 719
  - VcLinkAppearance 776
- LinkAppearanceCollection
  - VcGantt 529
- LinkCollection
  - VcGantt 529
- LinkDataTableName
  - VcResourceScheduler2 889
- LinkDurationDataFieldIndex
  - VcScheduler 951
- LinkDurationFieldIndex
  - VcResourceScheduler2 891
- LinkPredecessorDataFieldIndex
  - VcGantt 529
- LinkPredecessorOperationIDFieldIndex
  - VcResourceScheduler2 891
- LinkPredecessorTaskIDFieldIndex
  - VcResourceScheduler2 892
- LinksDataTableName
  - VcGantt 531
- LinkSuccessorDataFieldIndex
  - VcGantt 532
- LinkSuccessorOperationIDFieldIndex
  - VcResourceScheduler2 892
- LinkSuccessorTaskIDFieldIndex
  - VcResourceScheduler2 893
- LinkTypeDataFieldIndex
  - VcGantt 533
- MajorTicks
  - VcNumericScale 841
  - VcRibbon 941
- MajorTicksEx
  - VcNumericScale 842
- MapCollection
  - VcGantt 534
- Marked
  - VcBox 315
  - VcCurve 396
  - VcGroup 642
  - VcNode 819

- MarkedNodesFilter
  - VcFilterCollection 496
- MaxHorizontalPagesCount
  - VcPrinter 862
- MaximumTextLineCount
  - VcBoxFormatField 343
  - VcTableFormatField 982
- MaxVerticalPagesCount
  - VcPrinter 863
- Millimeter
  - VcMapEntry 811
- MinimumRowHeight
  - VcGantt 534
- MinimumTextLineCount
  - VcBoxFormatField 344
  - VcTableFormatField 982
- MinimumWidth
  - VcBoxFormatField 345
  - VcLayerFormatField 745
- MinorTicks
  - VcNumericScale 842
  - VcRibbon 942
- MinorTicksEx
  - VcNumericScale 843
- ModificationsAllowed
  - VcGroupLevelLayout 658
- Movable
  - VcLayer 720
- MovingLayersAsNodeWithShiftKeyAllowed
  - VcGantt 535
- MultiplePrimaryKeysAllowed
  - VcDataTable 447
- MultiState
  - VcTableFormatField 983
- Name
  - VcBox 316
  - VcBoxFormat 330
  - VcCalendar 350
  - VcCalendarGrid 369
  - VcCalendarProfile 376
  - VcCurve 396
  - VcDataDefinitionField 423
  - VcDataTable 447
  - VcDataTableField 457
  - VcDateLine 471
  - VcFilter 490
  - VcGroup 642
  - VcGroupLevelLayout 659
  - VcHistogram 675
  - VcLayer 720
  - VcLineFormat 751
  - VcLinkAppearance 777
  - VcMap 794
  - VcNumericScale 843
  - VcTableFormat 969
  - VcTimeScale 990
- NodeCalendarNameDataFieldIndex
  - VcGantt 535
- NodeCollection
  - VcGantt 536
  - VcGroup 643
- NodeDurationDataFieldIndex
  - VcGantt 536
- NodeEndDateDataFieldIndex
  - VcGantt 536
- NodeRowNumberDataFieldIndex
  - VcGantt 537
- NodesAndGroupsBelowCollapsed
  - VcGroup 643
- NodesArrangedInOneRow
  - VcGroup 644
- NodesArrangedOptimized
  - VcGroupLevelLayout 659

- NodesDataTableName
  - VcGantt 538
- NodesOptimized
  - VcGroup 644
- NodeSortingDataFieldIndex
  - VcGantt 538
- NodeSortingOrder
  - VcGantt 539
- NodeStartDateDataFieldIndex
  - VcGantt 540
- NodesUseCalendars
  - VcGantt 540
- NonWorkIntervalsCollapsed
  - VcSection 956
- Number
  - VcMapEntry 812
- NumericScaleCollection
  - VcGantt 540
  - VcHistogram 676
- ObjectDrawEventsEnabled
  - VcLayer 721
- OperationDataTableName
  - VcResourceScheduler2 894
- OperationLoadPerItemFieldIndex
  - VcResourceScheduler2 895
- OperationMaximumInterruptionTimeFieldIndex
  - VcResourceScheduler2 895
- OperationMinimumSupplementTimeFieldIndex
  - VcResourceScheduler2 896
- OperationOverlapQuantityFieldIndex
  - VcResourceScheduler2 897
- OperationPostLoadFieldIndex
  - VcResourceScheduler2 899
- OperationPreparationLoadFieldIndex
  - VcResourceScheduler2 899
- OperationResultEndDateFieldIndex
  - VcResourceScheduler2 900
- OperationResultPostEndDateFieldIndex
  - VcResourceScheduler2 901
- OperationResultPreparationStartDateFieldIndex
  - VcResourceScheduler2 901
- OperationResultProcessingTimeFieldIndex
  - VcResourceScheduler2 902
- OperationResultStartDateFieldIndex
  - VcResourceScheduler2 902
- OperationResultStatusFieldIndex
  - VcResourceScheduler2 903
- OperationRouteFieldIndex
  - VcResourceScheduler2 904
- OperationSequenceNumberFieldIndex
  - VcResourceScheduler2 904
- OperationStartLockDateFieldIndex
  - VcResourceScheduler2 905
- OperationTaskIDFieldIndex
  - VcResourceScheduler2 906
- OperationWorkInProgressFieldIndex
  - VcResourceScheduler2 906
- Operator
  - VcFilterSubCondition 504
- OptimizedNodesSortDataFieldIndex
  - VcGroupLevelLayout 659
- OptimizedNodesSortOrder
  - VcGroupLevelLayout 660
- Orientation
  - VcPrinter 863
- Origin
  - VcBox 316
- OutgoingLinks
  - VcNode 819, 820

## 1046 Index

- OverlaidNodesSortDataFieldIndex
  - VcGroupLevelLayout 660
- OverlaidNodesSortOrder
  - VcGroupLevelLayout 660
- OverlapLayerEnabled
  - VcGantt 541
- OverlapLayerName
  - VcGantt 541
- PageDescription
  - VcPrinter 864
- PageDescriptionString
  - VcPrinter 864
- PageFrame
  - VcPrinter 865
- PageNumberMode
  - VcPrinter 865
- PageNumbers
  - VcPrinter 866
- PagePaddingEnabled
  - VcPrinter 866
- PaperSize
  - VcPrinter 867
- Pattern
  - VcBoxFormatField 345
  - VcCalendarGrid 369
  - VcInterval 692
  - VcLayer 721
  - VcLineFormatField 763
  - VcMapEntry 812
  - VcNumericScale 844
  - VcRibbon 942
  - VcTableFormatField 983
- PatternColor
  - VcCalendarGrid 372
  - VcInterval 695
  - VcLayer 724
- PatternColorDataFieldIndex
  - VcCalendarGrid 373
  - VcLayer 724
- PatternColorMapName
  - VcCalendarGrid 373
  - VcLayer 725
- PatternDataFieldIndex
  - VcCalendarGrid 373
  - VcLayer 725
- PatternMapName
  - VcCalendarGrid 374
  - VcLayer 726
- Period
  - VcDateLineGrid 486
- PlanningEndDate
  - VcResourceScheduler2 907
- PlanningStartDate
  - VcResourceScheduler2 908
- PlanningStrategy
  - VcResourceScheduler2 909
- PointsEquidistant
  - VcCurve 397
- Position
  - VcRibbon 943
  - VcTable 961
- PredecessorLayerName
  - VcLinkAppearance 777
- PredecessorNode
  - VcLink 769
- PredecessorPortSymbol
  - VcLinkAppearance 778
- PrimaryKey
  - VcDataTableField 458
- PrintDate
  - VcPrinter 867
- Printer
  - VcGantt 541
- PrinterName

- VcPrinter 868
- PrintPreviewWithFirstPage
  - VcPrinter 868
- Priority
  - VcBox 317
  - VcCalendarGrid 374
  - VcDateLine 472
  - VcDateLineGrid 486
  - VcLayerFormatField 745
- ReferenceDate
  - VcRibbon 943
- ReferencePoint
  - VcBox 317
- RelationshipFieldIndex
  - VcDataTableField 458
- ReOptimizeNodesInGroupsEnabled
  - VcPrinter 869
- RequestImage
  - VcGantt 542
- ResourceCalendarNameFieldIndex
  - VcResourceScheduler2 910
- ResourceCapacityType
  - VcResourceScheduler2 910
- ResourceCapacityTypeFieldIndex
  - VcResourceScheduler2 911
- ResourceConstraintTypeFieldIndex
  - VcResourceScheduler2 912
- ResourceDataTableName
  - VcResourceScheduler2 913
- ResourceEfficiencyFieldIndex
  - VcResourceScheduler2 915
- ResourceGroupDataTableName
  - VcResourceScheduler2 916
- ResourceGroupIDFieldIndex
  - VcResourceScheduler2 917
- ResourceNameFieldIndex
  - VcResourceScheduler2 917
- ResourceResultLoadCurveNamePrefix
  - VcResourceScheduler2 918
- ResourceResultStockCurveNamePrefix
  - VcResourceScheduler2 919
- ResourceScheduler2
  - VcGantt 542
- ResourceSelectionStrategy
  - VcResourceScheduler2 920
- ResourceType
  - VcResourceScheduler2 921
- ResultProcessingStepCount
  - VcResourceScheduler2 922
- Ribbon
  - VcSection 957
  - VcTimeScale 991
- Right
  - VcRect 874
- RightMargin
  - VcTableFormatField 984
- RoundedLinkSlantsEnabled
  - VcGantt 543
- RoutingType
  - VcLinkAppearance 778
- RowBackColorAsARGB
  - VcGroupLevelLayout 661
- RowBackColorDataFieldIndex
  - VcGroupLevelLayout 661
- RowBackColorMapName
  - VcGroupLevelLayout 661
- RowBackgroundColorAsARGB
  - VcNodeLevelLayout 832
- RowBackgroundColorDataFieldIndex
  - VcNodeLevelLayout 832
- RowBackgroundColorMapName
  - VcNodeLevelLayout 833



- RowHeightReductionEnabled
  - VcGantt 543
- RowMargins
  - VcGantt 544
- RowPattern
  - VcGroupLevelLayout 662
  - VcNodeLevelLayout 833
- RowPatternColorAsARGB
  - VcGroupLevelLayout 662
  - VcNodeLevelLayout 833
- RowPatternColorDataFieldIndex
  - VcGroupLevelLayout 662
  - VcNodeLevelLayout 834
- RowPatternColorMapName
  - VcGroupLevelLayout 663
  - VcNodeLevelLayout 834
- RowPatternDataFieldIndex
  - VcGroupLevelLayout 663
  - VcNodeLevelLayout 834
- RowPatternMapName
  - VcGroupLevelLayout 663
  - VcNodeLevelLayout 835
- ScalingMode
  - VcPrinter 869
- ScheduledProjectEndDate
  - VcScheduler 951
- ScheduledProjectStartDate
  - VcScheduler 952
- Scheduler
  - VcGantt 544
- ScheduleSuccessorsOnlyEnabled
  - VcScheduler 952
- ScrollEventsEnabled
  - VcGantt 545
- SecondsPerWorkday
  - VcCalendar 351
- Section
  - VcTimeScale 991
- SeparationLineColor
  - VcGroupLevelLayout 664
  - VcNodeLevelLayout 835
  - VcTableFormat 970
- SeparationLineColorDataFieldIndex
  - VcGroupLevelLayout 664
- SeparationLineColorMapName
  - VcGroupLevelLayout 664
- SeparationLineInterval
  - VcNodeLevelLayout 835
- SeparationLinesVisible
  - VcGroupLevelLayout 665
  - VcNodeLevelLayout 836
- SeparationLinesVisibleAtTop
  - VcGroupLevelLayout 665
  - VcNodeLevelLayout 836
- SeparationLineThickness
  - VcGroupLevelLayout 665
  - VcNodeLevelLayout 836
- SeparationLineType
  - VcGroupLevelLayout 666
  - VcNodeLevelLayout 837
- Shape
  - VcLayer 728
- Sizeable
  - VcLayer 729
- SortDataFieldIndex
  - VcGroupLevelLayout 667
  - VcNodeLevelLayout 837
- SortOrder
  - VcGroupLevelLayout 667
  - VcNodeLevelLayout 838
- Source
  - VcCurve 397
- Specification
  - VcBox 318

- VcBoxFormat 330
- VcCalendar 351
- VcCalendarProfile 377
- VcCurve 398
- VcDateLine 472
- VcFilter 491
- VcGroupLevelLayout 667
- VcInterval 695
- VcLayer 729
- VcLineFormat 751
- VcMap 795
- StackReferenceName
  - VcCurve 398
- StartDataFieldIndex
  - VcLayer 729
- StartDate
  - VcSection 957
- StartDateForAutomaticScheduling
  - VcGantt 545
  - VcScheduler 952
- StartDateNotEarlierThanDataFieldIndex
  - VcScheduler 952
- StartDateTime
  - VcInterval 695
- StartMonth
  - VcInterval 696
- StartTime
  - VcInterval 696
- StartWeekday
  - VcInterval 696
- StringsCaseSensitive
  - VcFilter 491
- SubCondition
  - VcFilter 491
- SubConditionCount
  - VcFilter 492
- SubGroups
  - VcGroup 645
- SubRowMargins
  - VcGantt 545
- SuccessorLayerName
  - VcLinkAppearance 779
- SuccessorNode
  - VcLink 770
- SuccessorPortSymbol
  - VcLinkAppearance 780
- SummaryBarsVisible
  - VcGantt 546
  - VcGroupLevelLayout 668
- SuperGroup
  - VcGroup 645
  - VcNode 820
- TableColumnRanges
  - VcPrinter 870
- TableFormatCollection
  - VcTable 962
- TableTimeScaleOnAllPages
  - VcPrinter 870
- TaskDataTableName
  - VcResourceScheduler2 923
- TaskDueDateFieldIndex
  - VcResourceScheduler2 924
- TaskPlanningStrategyFieldIndex
  - VcResourceScheduler2 924
- TaskPriorityFieldIndex
  - VcResourceScheduler2 925
- TaskQuantityFieldIndex
  - VcResourceScheduler2 926
- TaskReleaseDateFieldIndex
  - VcResourceScheduler2 927
- TaskResultEndDateFieldIndex
  - VcResourceScheduler2 928
- TaskResultPostEndDateFieldIndex

- VcResourceScheduler2 928
- TaskResultPreparationStartDateFieldIndex
  - VcResourceScheduler2 929
- TaskResultProcessingStepFieldIndex
  - VcResourceScheduler2 929
- TaskResultProcessingTimeFieldIndex
  - VcResourceScheduler2 930
- TaskResultRouteFieldIndex
  - VcResourceScheduler2 931
- TaskResultStartDateFieldIndex
  - VcResourceScheduler2 931
- Text
  - VcBoundingBox 308
  - VcDateLine 473
  - VcInterval 697
- TextAlignment
  - VcRibbon 944
- TextAndGraphicsCombined
  - VcTableFormatField 984
- TextDataFieldIndex
  - VcLayerFormatField 745
  - VcLineFormatField 764
  - VcTableFormatField 984
- TextEntrySupplyingEventEnabled
  - VcGantt 546
- TextFont
  - VcBoundingBox 309
  - VcBoxFormatField 346
  - VcLayerFormatField 746
  - VcTableFormatField 984
- TextFontColor
  - VcBoxFormatField 347
  - VcLayerFormatField 746
  - VcLineFormatField 764
  - VcTableFormatField 985
- TextFontColorDataFieldIndex
  - VcLayerFormatField 746
  - VcLineFormatField 765
  - VcTableFormatField 985
- TextFontColorMapName
  - VcLayerFormatField 746
  - VcLineFormatField 765
  - VcTableFormatField 985
- TextFontDataFieldIndex
  - VcLayerFormatField 747
  - VcLineFormatField 765
  - VcTableFormatField 986
- TextFontMapName
  - VcLayerFormatField 747
  - VcLineFormatField 766
  - VcTableFormatField 986
- TextLineCount
  - VcLayerFormatField 747
  - VcLineFormatField 766
- TextLineCountDataFieldIndex
  - VcLayerFormatField 748
- TextLineCountMapName
  - VcLayerFormatField 748
- ThreeDEffect
  - VcLayer 730
  - VcNumericScale 845
  - VcTableFormat 970
  - VcTimeScale 992
- TickColor
  - VcRibbon 944
- TickPosition
  - VcRibbon 945
- TimeScaleAdjustment
  - VcPrinter 870
- TimeScaleCollection
  - VcGantt 547
- TimeScaleEnd
  - VcGantt 547

- TimeScaleStart
  - VcGantt 548
- TimeUnit
  - VcCurve 399
  - VcGantt 549
  - VcInterval 697
  - VcSection 958
- TimeUnitsPerStep
  - VcGantt 549
- Title
  - VcNumericScale 845
- ToleranceTimeOnASAPDueDates
  - VcResourceScheduler2 932
- ToleranceTimeOnJITReleaseDates
  - VcResourceScheduler2 932
- ToleranceTimeOnStartLockDates
  - VcResourceScheduler2 933
- ToolTipChangeDuration
  - VcGantt 550
- ToolTipDuration
  - VcGantt 550
- ToolTipPointerDuration
  - VcGantt 551
- ToolTipShowAfterClick
  - VcGantt 551
- ToolTipTextSupplyingEventEnabled
  - VcGantt 551
- Top
  - VcRect 874
- TopMargin
  - VcTableFormatField 986
- TotalFloatDataFieldIndex
  - VcScheduler 953
- TruncatedTextSuppressed
  - VcLayerFormatField 748
- TurningAnnotationEnabled
  - VcDateLineGrid 487
- Type
  - VcBoundingBox 310
  - VcBoxFormatField 347
  - VcCalendar 351
  - VcCalendarProfile 377
  - VcCurve 399
  - VcDataDefinitionField 423
  - VcDataTableField 460
  - VcInterval 697
  - VcMap 795
  - VcRibbon 945
  - VcTableFormatField 986
- Unit
  - VcDateLineGrid 487
  - VcNumericScale 846
- UnitLabel
  - VcNumericScale 846
- UnitSeparation
  - VcRibbon 946
- UnitsPerStep
  - VcCurve 400
- UnitWidth
  - VcNumericScale 846
  - VcSection 959
- UsedAsOverlapLayer
  - VcLayer 730
- UseGraphicalAttributes
  - VcInterval 698
- UseGraphicalAttributesOfIntervals
  - VcCalendarGrid 375
- UseReferenceDate
  - VcDateLineGrid 488
  - VcRibbon 946
- ValencyDataFieldIndex
  - VcCurve 401
- VerticalNodeMovementAllowed
  - VcGantt 552

- VerticalNodeMovementViaTableAllowed
    - VcGantt 552
  - VerticalOffset
    - VcLayer 731
  - VerticalOffsetDataFieldIndex
    - VcLayer 731
  - VerticalOffsetMapName
    - VcLayer 731
  - Visible
    - VcBox 319
    - VcCalendarGrid 375
    - VcCurve 401
    - VcDateLine 473
    - VcDateLineGrid 488
    - VcGroup 646
    - VcGroupLevelLayout 668
    - VcHistogram 676
    - VcLayer 732
    - VcLinkAppearance 780
    - VcTable 962
  - VisibleInLegend
    - VcLayer 732
  - Width
    - VcRect 875
  - WorkInProcessType
    - VcResourceScheduler2 933
  - WritingDebugFilesEnabled
    - VcResourceScheduler2 934
  - ZoomFactor
    - VcGantt 553
  - ZoomFactorAsDouble
    - VcPrinter 871
  - ZoomingPerMouseWheelAllowed
    - VcGantt 553
  - Property page**
    - Border Area 152
    - General 146
    - Layout 158
    - Link 164
    - Node 154
    - Objects 162
    - Schedule 166
  - PutInOrderAfter**
    - Method of
      - VcCalendarProfile 377
      - VcDateLine 474
      - VcHistogram 678
      - VcLayer 733
      - VcLinkAppearance 781
- R**
- RecalculateAllStructureCodes**
    - Method of
      - VcGantt 575
  - Rect**
    - see also
      - VcRect 872
  - Reference curve 96**
  - ReferenceDate**
    - Property of
      - VcRibbon 943
  - ReferencePoint**
    - Property of
      - VcBox 317
  - RelatedDataRecord**
    - Method of
      - VcDataRecord 435
      - VcGroup 648
      - VcLink 771
      - VcNode 824
  - Relation**
    - type 165
  - RelationshipFieldIndex**

- Property of
  - VcDataTableField 458
- Remove**
  - Method of
    - VcBoxCollection 326
    - VcBoxFormatCollection 338
    - VcCalendarCollection 363
    - VcCurveCollection 417
    - VcDataRecordCollection 443
    - VcFilterCollection 500
    - VcGroupLevelLayoutCollection 673
    - VcIntervalCollection 703
    - VcLayerCollection 738
    - VcLineFormatCollection 757
    - VcLinkAppearanceCollection 787
    - VcMapCollection 804
- RemoveFormatField**
  - Method of
    - VcBoxFormat 332
    - VcLayerFormat 742
    - VcLineFormat 752
- RemoveSubCondition**
  - Method of
    - VcFilter 494
- ReOptimizeNodes**
  - Method of
    - VcGroup 648
- ReOptimizeNodesInGroupsEnabled**
  - Property of
    - VcPrinter 869
- RequestImage**
  - Property of
    - VcGantt 542
- Reset**
  - Method of
    - VcGantt 575
- Resource scheduler**
  - lock date 905
  - resource group 917
  - resource name 918
  - scheduling progress 619
  - task end date 928
  - task planning strategy 925
  - task priority 926
  - task processing time 930
  - task route 931
  - task start date 931
  - warnings 620
- Resource Scheduler 117**
- resource scheduling**
  - calendar name 910
- Resource scheduling**
  - allowed due date variation 932
  - allowed release date variation 932
  - assignment, visible 882
  - assignment, associated operation 884
  - assignment, associated resource 885
  - assignment, data set generation 881
  - assignment, quantity multiplier 882
  - assignment, table name 880
  - assignmentData table 890
  - calendar, default name 888
  - capacity of the production system 910
  - capacity, full usage of 889
  - completion 907
  - debug files 934
  - end date of scheduling period 907
  - events 887
  - job completion 933
  - link predecessor operation 891, 893
  - link, predecessor task 892

- link, successor task 893
  - link, temporal distance 891
  - locked start date allowance 933
  - operation finish date 900
  - operation maximum interruption time 896
  - operation minimum supplement time 896
  - operation post time 899
  - operation preparation time 899
  - operation processing time 902
  - operation start date 902
  - operation status 903
  - operation table, quantity multiplier 895
  - operation, default value for maximum interruption time 888
  - operation, end date of post time 901, 928
  - operation, overlap quantity 898
  - operation, start date of preparation time 901, 929
  - operation, table name 894
  - operations table, associated task 906
  - operations table, routes 904
  - operations table, sequence of operations 904
  - planning strategy 909
  - process 935
  - resource constraint 912
  - resource selection 920
  - resource selection strategy 886
  - resource type 922
  - resource type single 912
  - resource, maximum work load limit 884
  - resource, minimum work load limit 883
  - resource, name of group table 916
  - resource, table name 914
  - setting to the Gantt object 542
  - start date of scheduling period 908
  - stock curve 919
  - task due date 924
  - task quantity 927
  - task release date 927
  - task sequence 930
  - task, table name 923
  - tasks, number of 922
  - time unit, basic 886
  - time units per step 887
  - workload curve 918
- resource scheduling**
- efficiency 915
- ResourceCalendarNameFieldIndex**
- Property of
    - VcResourceScheduler2 910
- ResourceCapacityType**
- Property of
    - VcResourceScheduler2 910
- ResourceCapacityTypeFieldIndex**
- Property of
    - VcResourceScheduler2 911
- ResourceConstraintTypeFieldIndex**
- Property of
    - VcResourceScheduler2 912
- ResourceDataTableName**
- Property of
    - VcResourceScheduler2 913
- ResourceEfficiencyFieldIndex**
- Property of
    - VcResourceScheduler2 915
- ResourceGroupDataTableName**
- Property of
    - VcResourceScheduler2 916
- ResourceGroupIDFieldIndex**

- Property of
  - VcResourceScheduler2 917
- ResourceNameFieldIndex**
  - Property of
    - VcResourceScheduler2 917
- ResourceResultLoadCurveNamePrefix**
  - Property of
    - VcResourceScheduler2 918
- ResourceResultStockCurveNamePrefix**
  - Property of
    - VcResourceScheduler2 919
- ResourceScheduler2**
  - Property of
    - VcGantt 542
  - see also
    - VcResourceScheduler2 876
- ResourceSelectionStrategy**
  - Property of
    - VcResourceScheduler2 920
- ResourceType**
  - Property of
    - VcResourceScheduler2 921
- ResultProcessingStepCount**
  - Property of
    - VcResourceScheduler2 922
- Return status 79**
- Ribbon 135, 251**
  - background color 938
  - calendar 938
  - date format 940
  - fill pattern 942
  - font 940
  - font color 941
  - major ticks 941
  - minor ticks 942
  - position 943
  - Property of
    - VcSection 957
    - VcTimeScale 991
  - reference date 943, 946
  - see also
    - VcRibbon 937
  - text alignment 944
  - tick color 944
  - tick position 945
  - type 945
  - unit separation 946
- Right**
  - Property of
    - VcRect 874
- RightMargin**
  - Property of
    - VcTableFormatField 984
- RoundedLinkSlantsEnabled**
  - Property of
    - VcGantt 543
- RoutingType**
  - Property of
    - VcLinkAppearance 778
- Row background color**
  - alternating 160
- Row height**
  - reduction 150, 151
- RowBackColorAsARGB**
  - Property of
    - VcGroupLevelLayout 661
- RowBackColorDataFieldIndex**
  - Property of
    - VcGroupLevelLayout 661
- RowBackColorMapName**
  - Property of
    - VcGroupLevelLayout 661



**RowBackgroundColorAsARGB**

Property of  
VcNodeLevelLayout 832

**RowBackgroundColorDataFieldIndex**

Property of  
VcNodeLevelLayout 832

**RowBackgroundColorMapName**

Property of  
VcNodeLevelLayout 833

**RowHeightReductionEnabled**

Property of  
VcGantt 543

**RowMargins**

Property of  
VcGantt 544

**RowPattern**

Property of  
VcGroupLevelLayout 662  
VcNodeLevelLayout 833

**RowPatternColorAsARGB**

Property of  
VcGroupLevelLayout 662  
VcNodeLevelLayout 833

**RowPatternColorDataFieldIndex**

Property of  
VcGroupLevelLayout 662  
VcNodeLevelLayout 834

**RowPatternColorMapName**

Property of  
VcGroupLevelLayout 663  
VcNodeLevelLayout 834

**RowPatternDataFieldIndex**

Property of  
VcGroupLevelLayout 663  
VcNodeLevelLayout 834

**RowPatternMapName**

Property of

VcGroupLevelLayout 663

VcNodeLevelLayout 835

**Run Time Licenses 15**

**S**

**SaveAsEx**

Method of  
VcGantt 576

**ScalingMode**

Property of  
VcPrinter 869

**ScheduledProjectEndDate**

Property of  
VcScheduler 951

**ScheduledProjectStartDate**

Property of  
VcScheduler 952

**ScheduleProject**

Method of  
VcScheduler 953

**Scheduler**

Property of  
VcGantt 544

see also

VcScheduler 947

**ScheduleSuccessorsOnlyEnabled**

Property of  
VcScheduler 952

**Scheduling 121, 544**

actual end date 948

actual start date 948

Autoschedule 167

duration 949

earliest possible end date 949

earliest possible start date 949

free float 950

latest possible end date 950

- latest possible start date 951
- link duration 951
- schedule input 166
- schedule result 166
- scheduled end date 950
- scheduled start date 952
- scheduling only of nodes with predecessors 952
- total float 953
- Scheduling: 951, 952**
- Scroll events**
  - enable/disable 150
- ScrollEventsEnabled**
  - Property of VcGantt 545
- Scrolling**
  - to a value in histogram 678
- ScrollToDate**
  - Method of VcGantt 576
- ScrollToGroupLine**
  - Method of VcGantt 577
- ScrollToNode**
  - Method of VcGantt 578
- ScrollToNodeLine**
  - Method of VcGantt 579
- ScrollToValue**
  - Method of VcHistogram 678
- SecondsPerWorkday**
  - Property of VcCalendar 351
- Section 134, 249**
  - collapsing workfree periods 251
- Property of VcTimeScale 991
- see also VcSection 955
- unit width 251
- SelectGroups**
  - Method of VcGroupCollection 653
- SelectLinks**
  - Method of VcLinkCollection 791
- SelectMaps**
  - Method of VcMapCollection 804
- SelectNodes**
  - Method of VcNodeCollection 829
- Separation lines**
  - interval 835
- SeparationLineColor**
  - Property of VcGroupLevelLayout 664
  - VcNodeLevelLayout 835
  - VcTableFormat 970
- SeparationLineColorDataFieldIndex**
  - Property of VcGroupLevelLayout 664
- SeparationLineColorMapName**
  - Property of VcGroupLevelLayout 664
- SeparationLineInterval**
  - Property of VcNodeLevelLayout 835
- SeparationLinesVisible**
  - Property of VcGroupLevelLayout 665
  - VcNodeLevelLayout 836

**SeparationLinesVisibleAtTop**

- Property of
  - VcGroupLevelLayout 665
  - VcNodeLevelLayout 836

**SeparationLineThickness**

- Property of
  - VcGroupLevelLayout 665
  - VcNodeLevelLayout 836

**SeparationLineType**

- Property of
  - VcGroupLevelLayout 666
  - VcNodeLevelLayout 837

**Session state 280, 281, 282**

**Session Timeout**

- Reloading 297

**SetMaxYValue**

- Method of
  - VcHistogram 679

**SetMinYValue**

- Method of
  - VcHistogram 679

**SetPositionInView**

- Method of
  - VcNode 824

**SetValues**

- Method of
  - VcCurve 409

**SetXYOffset**

- Method of
  - VcBox 320

**SetXYOffsetByTopLeftPixel**

- Method of
  - VcBox 320

**Shape**

- Property of
  - VcLayer 728

**Shift calendar**

- annotation of the time ribbon 697

**Shipping 16**

**Sizeable**

- Property of
  - VcLayer 729

**Smallest time interval 147**

**SortDataFieldIndex**

- Property of
  - VcGroupLevelLayout 667
  - VcNodeLevelLayout 837

**SortGroups**

- Method of
  - VcGantt 580

**Sorting 124, 201**

**SortNodes**

- Method of
  - VcGantt 580

**SortOrder**

- Property of
  - VcGroupLevelLayout 667
  - VcNodeLevelLayout 838

**Source**

- Property of
  - VcCurve 397

**Specification**

- Property of
  - VcBox 318
  - VcBoxFormat 330
  - VcCalendar 351
  - VcCalendarProfile 377
  - VcCurve 398
  - VcDateLine 472
  - VcFilter 491
  - VcGroupLevelLayout 667
  - VcInterval 695
  - VcLayer 729
  - VcLineFormat 751

- VcMap 795
- StackReferenceName**
  - Property of
    - VcCurve 398
- StartDataFieldIndex**
  - Property of
    - VcLayer 729
- StartDate**
  - Property of
    - VcSection 957
- StartDateForAutomaticScheduling**
  - Property of
    - VcGantt 545
    - VcScheduler 952
- StartDateNotEarlierThanDataFieldIndex**
  - Property of
    - VcScheduler 952
- StartDateTime**
  - Property of
    - VcInterval 695
- StartMonth**
  - Property of
    - VcInterval 696
- StartTime**
  - Property of
    - VcInterval 696
- StartWeekday**
  - Property of
    - VcInterval 696
- StringsCaseSensitive**
  - Property of
    - VcFilter 491
- SubCondition**
  - Property of
    - VcFilter 491
- SubConditionCount**
  - Property of
    - VcFilter 492
- SubGroups**
  - Property of
    - VcGroup 645
- SubRowMargins**
  - Property of
    - VcGantt 545
- SuccessorLayerName**
  - Property of
    - VcLinkAppearance 779
- SuccessorNode**
  - Property of
    - VcLink 770
- SuccessorPortSymbol**
  - Property of
    - VcLinkAppearance 780
- Summary bar 88**
- Summary bars**
  - displaying 200
- SummaryBarsVisible**
  - Property of
    - VcGantt 546
    - VcGroupLevelLayout 668
- SuperGroup**
  - Property of
    - VcGroup 645
    - VcNode 820
- Support 20**
- SuspendUpdate**
  - Method of
    - VcGantt 581

## T

- Table format**
  - by index 973
- Tabellenformat**

- enumeration object 971
- Table 131**
  - by index 964
  - columns 228
  - editing 228
  - editing formats 230
  - see also
    - VcTable 960
  - specifying 226
  - table formats 228
- Table format**
  - 3D effect 970
  - display of +/- 966
  - field by index 967
  - filter 967
  - indentation column 968
  - indentation width of text 969
  - name 969
  - number of table columns 968
  - separation line color 970
  - separation lines visible 966
- TableByIndex**
  - Method of
    - VcTableCollection 964
- TableCollection**
  - see also
    - VcTableCollection 964
- TableColumnRanges**
  - Property of
    - VcPrinter 870
- TableFormat**
  - see also
    - VcTableFormat 965
- TableFormatCollection**
  - Property of
    - VcTable 962
  - see also
    - VcTableFormatCollection 972
- TableFormatField**
  - see also
    - VcTableFormatField 976
- TableTimeScaleOnAllPages**
  - Property of
    - VcPrinter 870
- TaskDataTableName**
  - Property of
    - VcResourceScheduler2 923
- TaskDueDateFieldIndex**
  - Property of
    - VcResourceScheduler2 924
- TaskPlanningStrategyFieldIndex**
  - Property of
    - VcResourceScheduler2 924
- TaskPriorityFieldIndex**
  - Property of
    - VcResourceScheduler2 925
- TaskQuantityFieldIndex**
  - Property of
    - VcResourceScheduler2 926
- TaskReleaseDateFieldIndex**
  - Property of
    - VcResourceScheduler2 927
- TaskResultEndDateFieldIndex**
  - Property of
    - VcResourceScheduler2 928
- TaskResultPostEndDateFieldIndex**
  - Property of
    - VcResourceScheduler2 928
- TaskResultPreparationStartDateFieldIndex**
  - Property of
    - VcResourceScheduler2 929
- TaskResultProcessingStepFieldIndex**
  - Property of

- VcResourceScheduler2 929
- TaskResultProcessingTimeFieldIndex**
  - Property of
    - VcResourceScheduler2 930
- TaskResultRouteFieldIndex**
  - Property of
    - VcResourceScheduler2 931
- TaskResultStartDateFieldIndex**
  - Property of
    - VcResourceScheduler2 931
- Text**
  - Property of
    - VcBoundingBox 308
    - VcDateLine 473
    - VcInterval 697
- TextAlignment**
  - Property of
    - VcRibbon 944
- TextAndGraphicsCombined**
  - Property of
    - VcTableFormatField 984
- TextDataFieldIndex**
  - Property of
    - VcLayerFormatField 745
    - VcLineFormatField 764
    - VcTableFormatField 984
- TextEntrySupplyingEventEnabled**
  - Property of
    - VcGantt 546
- TextFont**
  - Property of
    - VcBoundingBox 309
    - VcBoxFormatField 346
    - VcLayerFormatField 746
    - VcTableFormatField 984
- TextFontColor**
  - Property of
    - VcBoxFormatField 347
    - VcLayerFormatField 746
    - VcLineFormatField 764
    - VcTableFormatField 985
- TextFontColorDataFieldIndex**
  - Property of
    - VcLayerFormatField 746
    - VcLineFormatField 765
    - VcTableFormatField 985
- TextFontColorMapName**
  - Property of
    - VcLayerFormatField 746
    - VcLineFormatField 765
    - VcTableFormatField 985
- TextFontDataFieldIndex**
  - Property of
    - VcLayerFormatField 747
    - VcLineFormatField 765
    - VcTableFormatField 986
- TextFontMapName**
  - Property of
    - VcLayerFormatField 747
    - VcLineFormatField 766
    - VcTableFormatField 986
- TextLineCount**
  - Property of
    - VcLayerFormatField 747
    - VcLineFormatField 766
- TextLineCountDataFieldIndex**
  - Property of
    - VcLayerFormatField 748
- TextLineCountMapName**
  - Property of
    - VcLayerFormatField 748
- ThreeDEffect**
  - Property of
    - VcLayer 730

- VcNumericScale 845
- VcTableFormat 970
- VcTimeScale 992
- TickColor**
  - Property of
    - VcRibbon 944
- TickPosition**
  - Property of
    - VcRibbon 945
- Time interval**
  - smallest 147
- Time scale 133**
  - adjust 871
  - by index 995
  - ribbons 135, 251
  - sections 134
  - start and end 134
- Time scheduling**
  - automatic 948
- Time unit 147**
- Time unit width 135**
- TimeScale**
  - see also
    - VcTimeScale 988
- TimeScaleAdjustment**
  - Property of
    - VcPrinter 870
- TimeScaleByIndex**
  - Method of
    - VcTimeScaleCollection 995
- TimeScaleByName**
  - Method of
    - VcTimeScaleCollection 996
- TimeScaleCollection**
  - Property of
    - VcGantt 547
  - see also
    - VcTimeScaleCollection 993
- TimeScaleEnd**
  - Property of
    - VcGantt 547
- TimeScaleStart**
  - Property of
    - VcGantt 548
- TimeUnit**
  - Property of
    - VcCurve 399
    - VcGantt 549
    - VcInterval 697
    - VcSection 958
- TimeUnitsPerStep**
  - Property of
    - VcGantt 549
- Title**
  - Property of
    - VcNumericScale 845
- ToleranceTimeOnASAPDueDates**
  - Property of
    - VcResourceScheduler2 932
- ToleranceTimeOnJITReleaseDates**
  - Property of
    - VcResourceScheduler2 932
- ToleranceTimeOnStartLockDates**
  - Property of
    - VcResourceScheduler2 933
- Tool tip**
  - disappearance on click 551
  - duration of appearance 550
  - duration of change 550
  - time elapsed till appearance 551
- ToolTipChangeDuration**
  - Property of
    - VcGantt 550
- ToolTipDuration**

- Property of
  - VcGantt 550
- ToolTipPointerDuration**
  - Property of
    - VcGantt 551
- Tooltips 150**
  - data field for text 155
  - during runtime 139
- ToolTipShowAfterClick**
  - Property of
    - VcGantt 551
- ToolTipTextSupplyingEventEnabled**
  - Property of
    - VcGantt 551
- Top**
  - Property of
    - VcRect 874
- TopMargin**
  - Property of
    - VcTableFormatField 986
- TotalFloatDataFieldIndex**
  - Property of
    - VcScheduler 953
- Tree view style 226**
- TruncatedTextSuppressed**
  - Property of
    - VcLayerFormatField 748
- TurningAnnotationEnabled**
  - Property of
    - VcDateLineGrid 487
- Type**
  - Property of
    - VcBoundingBox 310
    - VcBoxFormatField 347
    - VcCalendar 351
    - VcCalendarProfile 377
    - VcCurve 399
    - VcDataDefinitionField 423
    - VcDataTableField 460
    - VcInterval 697
    - VcMap 795
    - VcRibbon 945
    - VcTableFormatField 986

## U

### Unicode 140

#### Unit

- Property of
  - VcDateLineGrid 487
  - VcNumericScale 846

#### UnitLabel

- Property of
  - VcNumericScale 846

#### UnitSeparation

- Property of
  - VcRibbon 946

#### UnitsPerStep

- Property of
  - VcCurve 400

#### UnitWidth

- Property of
  - VcNumericScale 846
  - VcSection 959

#### Update

- Method of
  - VcBoxCollection 327
  - VcCalendar 356
  - VcCalendarCollection 364
  - VcDataRecord 436
  - VcDataRecordCollection 444
  - VcDataTableCollection 453
  - VcGroup 648
  - VcGroupLevelLayoutCollection 673



VcIntervalCollection 703

VcLayerCollection 739

VcLink 771

VcLinkAppearanceCollection 787

VcMapCollection 805

VcNode 825

**UpdateLinkRecord**

Method of

VcGantt 582

**UpdateNodeRecord**

Method of

VcGantt 583

**UpdateRowNumberFields**

Method of

VcGantt 583

**URL on nodes 296**

**UsedAsOverlapLayer**

Property of

VcLayer 730

**UseGraphicalAttributes**

Property of

VcInterval 698

**UseGraphicalAttributesOfIntervals**

Property of

VcCalendarGrid 375

**User account**

control not working 290

**UseReferenceDate**

Property of

VcDateLineGrid 488

VcRibbon 946

placing in a form 22

**VcBorderArea 301**

BorderBox 301

**VcBorderBox 303**

GraphicsFileName 303

LegendElementsArrangement 304

LegendElementsBottomMargin 305

LegendElementsMaximumColumnCount 305

LegendElementsMaximumRowCount 305

LegendElementsTopMargin 306

LegendFont 306

LegendTitle 306

LegendTitleFont 307

LegendTitleVisible 307

Text 308

TextFont 309

Type 310

**VcBox 311**

FieldText 311

FormatName 312

GetXYOffset 319

LineColor 313

LineThickness 313

LineType 314

Marked 315

Name 316

Origin 316

Priority 317

ReferencePoint 317

SetXYOffset 320

SetXYOffsetByTopLeftPixel 320

Specification 318

Visible 319

**VcBoxClickingEventArgs**

Event argument object of

**V**

**ValencyDataFieldIndex**

Property of

VcCurve 401

**VARCHART XGantt**

- VcBoxLeftClicking 584
- VcBoxCollection 321**
  - Add 322
  - AddBySpecification 322
  - BoxByIndex 323
  - BoxByName 324
  - Copy 324
  - Count 321
  - FirstBox 325
  - GetEnumerator 325
  - NextBox 326
  - Remove 326
  - Update 327
- VcBoxFormat 328**
  - CopyFormatField 331
  - FieldsSeparatedByLines 328
  - FormatField 329
  - FormatFieldCount 329
  - GetEnumerator 331
  - Name 330
  - RemoveFormatField 332
  - Specification 330
- VcBoxFormatCollection 333**
  - Add 334
  - AddBySpecification 334
  - Copy 335
  - Count 333
  - FirstFormat 335
  - FormatByIndex 336
  - FormatByName 336
  - GetEnumerator 337
  - NextFormat 337
  - Remove 338
- VcBoxFormatField 340**
  - Alignment 340
  - BackgroundColor 341
  - FormatName 342
  - GraphicsHeight 342
  - Index 343
  - MaximumTextLineCount 343
  - MinimumTextLineCount 344
  - MinimumWidth 345
  - Pattern 345
  - TextFont 346
  - TextFontColor 347
  - Type 347
- VcBoxLeftClicking**
  - Event of
    - VcGantt 584
- VcCalendar 349**
  - AddDuration 352
  - CalcDuration 352
  - CalendarProfileCollection 350
  - GetEndOfPreviousWorktime 353
  - GetNextIntervalBorder 354
  - GetPreviousIntervalBorder 354
  - GetStartOfInterval 355
  - GetStartOfNextWorktime 355
  - IntervalCollection 350
  - IsWorktime 356
  - Name 350
  - SecondsPerWorkday 351
  - Specification 351
  - Type 351
  - Update 356
- VcCalendarCollection 358**
  - Active 358
  - Add 360
  - AddBySpecification 360
  - CalendarByIndex 361
  - CalendarByName 361
  - Copy 361
  - Count 359
  - FirstCalendar 362

- GetEnumerator 362
- NextCalendar 363
- Remove 363
- Update 364
- VcCalendarGrid 365**
  - BackgroundColor 366
  - BackgroundColorDataFieldIndex 366
  - BackgroundColorMapName 367
  - CalendarName 367
  - LineColor 367
  - LineColorDataFieldIndex 368
  - LineColorMapName 368
  - LineType 368
  - Name 369
  - Pattern 369
  - PatternColor 372
  - PatternColorDataFieldIndex 373
  - PatternColorMapName 373
  - PatternDataFieldIndex 373
  - PatternMapName 374
  - Priority 374
  - UseGraphicalAttributesOfIntervals 375
  - Visible 375
- VcCalendarProfile 376**
  - IntervalCollection 376
  - Name 376
  - PutInOrderAfter 377
  - Specification 377
  - Type 377
- VcComponentScrolled**
  - Event of
    - VcGantt 585
- VcComponentScrolledEventArgs**
  - Event argument objekt of
    - VcComponentScrolled 586
- VcComponentScrolling**
  - Event of
    - VcGantt 588
- VcComponentScrollingEventArgs**
  - Event argument objekt of
    - VcComponentScrolling 589
- VcCurve 379**
  - Addend 380
  - Clear 402
  - DeletePoint 402
  - FillReference1BackgroundColor 381
  - FillReference1Name 381
  - FillReference1Pattern 382
  - FillReference1PatternColor 385
  - FillReference2Color 386
  - FillReference2Name 387
  - FillReference2Pattern 387
  - FillReference2PatternColor 390
  - FilterName 391
  - GetFirstOverload 403
  - GetFirstOverloadEx 404
  - GetNextOverload 405
  - GetNextOverloadEx 406
  - GetValues 407
  - GetValuesEx 408
  - Histogram 392
  - LayerName 392
  - LineColor 393
  - LineThickness 393
  - LineType 394
  - Marked 396
  - Name 396
  - PointsEquidistant 397
  - SetValues 409
  - Source 397
  - Specification 398
  - StackReferenceName 398
  - TimeUnit 399

- Type 399
- UnitsPerStep 400
- ValencyDataFieldIndex 401
- Visible 401
- VcCurveClickingEventArgs**
  - Event argument objekt of
    - VcCurveLeftClicking 592
- VcCurveCollection 411**
  - Add 412
  - AddBySpecification 413
  - Copy 413
  - Count 411
  - CurveByIndex 414
  - CurveByName 414
  - FirstCurve 415
  - GetEnumerator 415
  - NextCurve 416
  - Remove 417
- VcCurveLeftClicking**
  - Event of
    - VcGantt 591
- VcDataDefinition 418**
  - DataDefinitionTable 418
- VcDataDefinitionField 420**
  - DateFormat 420
  - Editable 421
  - Hidden 422
  - Index 422
  - Name 423
  - Type 423
- VcDataDefinitionTable 425**
  - Count 425
  - CreateDataDefinitionField 426
  - DataDefinitionFieldByIndex 427
  - DataDefinitionFieldByName 427
  - FirstDataDefinitionField 428
  - GetEnumerator 429
  - NextDataDefinitionField 429
- VcDataRecord 431**
  - AllData 431
  - DataField 432
  - DataTableName 433
  - Delete 434
  - ID 434
  - RelatedDataRecord 435
  - Update 436
- VcDataRecordCollection 437**
  - Add 438
  - Count 437
  - DataRecordByID 439
  - FirstDataRecord 440
  - GetEnumerator 441
  - GetNewUniqueID 442
  - NextDataRecord 442
  - Remove 443
  - Update 444
- VcDataRecordModified**
  - Event of
    - VcGantt 592
- VcDataRecordModifiedEventArgs**
  - Event argument objekt of
    - VcDataRecordModified 592
- VcDataRecordModifying**
  - Event of
    - VcGantt 593
- VcDataRecordModifyingEventArgs**
  - Event argument objekt of
    - VcDataRecordModifying 593
- VcDataRecordNotFound**
  - Event of
    - VcGantt 594
- VcDataRecordNotFoundEventArgs**
  - Event argument objekt of
    - VcDataRecordNotFound 594

**VcDataTable 445**

- DataRecordCollection 445
- DataTableFieldCollection 446
- Description 446
- MultiplePrimaryKeysAllowed 447
- Name 447

**VcDataTableCollection 448**

- Add 449
- Copy 449
- Count 448
- DataTableByIndex 450
- DataTableByName 451
- FirstDataTable 451
- GetEnumerator 452
- NextDataTable 452
- Update 453

**VcDataTableField 454**

- DataTableName 454
- DateFormat 455
- Editable 456
- Hidden 456
- Index 457
- Name 457
- PrimaryKey 458
- RelationshipFieldIndex 458
- Type 460

**VcDataTableFieldCollection 461**

- Add 462
- Copy 462
- Count 461
- DataTableFieldByIndex 463
- DataTableFieldByName 464
- FirstDataTableField 464
- GetEnumerator 465
- NextDataTableField 465

**VcDateLine 467**

- AlwaysCurrentDate 467

- Date 468
- LineColor 468
- LineThickness 469
- LineType 470
- Name 471
- Priority 472
- PutInOrderAfter 474
- Specification 472
- Text 473
- Visible 473

**VcDateLineCollection 475**

- Count 475
- DateLineByIndex 476
- DateLineByName 476
- FirstDateLine 477
- GetEnumerator 478
- NextDateLine 478

**VcDateLineGrid 480**

- AdjustToReferenceDate 481
- AnnotationAtBottom 481
- AnnotationAtCenter 481
- AnnotationAtTop 482
- FormatName 482
- HorAlignment 482
- LineColor 483
- LineColorDataFieldIndex 483
- LineColorMapName 483
- LineThickness 484
- LineType 485
- Period 486
- Priority 486
- TurningAnnotationEnabled 487
- Unit 487
- UseReferenceDate 488
- Visible 488

**VcDiagramClickingEventArgs**

- Event argument object of

- VcDiagramLeftClicking 598
- VcDiagramHorizontalScrolled**
  - Event of
    - VcGantt 594
- VcDiagramHorizontalScrolledEventArgs**
  - Event argument objekt of
    - VcDiagramHorizontalScrolled 595
- VcDiagramHorizontalScrolling**
  - Event of
    - VcGantt 596
- VcDiagramHorizontalScrollingEventArgs**
  - Event argument objekt of
    - VcDiagramHorizontalScrolling 596
- VcDiagramLeftClicking**
  - Event of
    - VcGantt 598
- VcErrorOccurring**
  - Event of
    - VcGantt 599
- VcErrorOccurringEventArgs**
  - Event argument objekt of
    - VcErrorOccurring 599
- VcFieldSelecting**
  - Event of
    - VcGantt 600
- VcFieldSelectingEventArgs**
  - Event argument objekt of
    - VcFieldSelecting 600
- VcFilter 489**
  - AddSubCondition 492
  - CopySubCondition 493
  - DataDefinitionTable 490
  - DatesWithHourAndMinute 490
  - GetEnumerator 493
  - IsValid 494
  - Name 490
  - RemoveSubCondition 494
  - Specification 491
  - StringsCaseSensitive 491
  - SubCondition 491
  - SubConditionCount 492
- VcFilterCollection 495**
  - Add 496
  - AddBySpecification 497
  - Copy 497
  - Count 495
  - FilterByIndex 498
  - FilterByName 498
  - FirstFilter 498
  - GetEnumerator 499
  - MarkedNodesFilter 496
  - NextFilter 499
  - Remove 500
- VcFilterSubCondition 501**
  - ComparisonValueAsString 501
  - ConnectionOperator 502
  - DataFieldIndex 502
  - FilterName 503
  - GetEnumerator 505
  - Index 503
  - IsValid 505
  - Operator 504
- VcGantt 506**
  - ActiveNodeFilter 510
  - AjaxExtensionsEnabled 511
  - Arrangement 511
  - BorderArea 512
  - BoxCollection 512
  - BoxFormatCollection 513
  - CalendarCollection 513
  - CalendarGridCollection 513
  - CalendarProfileCollection 514

- ConvertDistance 554
- DataDefinition 514
- DataTableCollection 515
- DateLineCollection 515
- DateOutputFormat 515
- DeleteLinkRecord 554
- DeleteNodeRecord 555
- DetectDataTableFieldName 555
- DetectDataTableName 555
- DetectFieldIndex 556
- DiagramAlternatingRowBackgroundColor 517
- DiagramBackgroundColor 517
- DiagramHistogramHeightRatio 518
- Enabled 518
- EndDateForAutomaticScheduling 519
- EndLoading 557
- ExportGraphicsToFileEx 557
- ExtendedDataTablesEnabled 519
- FilePath 520
- FilterCollection 520
- FitChartIntoView 559
- FitHistogramsIntoView 560
- FitRangeIntoView 560
- FontAntiAliasingEnabled 521
- GetCurrentViewDates 561
- GetDate 561
- GetLicenseInformation 562
- GetLinkByID 562
- GetLinkByNodeIDs 563
- GetNodeByID 563
- GetViewComponentSize 564
- GroupCollection 521
- GroupingDataFieldIndex 521
- GroupingModificationsAllowed 522
- GroupNodes 565
- GroupOptimizationOnInteractionsEnabled 523
- GroupSortingDataFieldIndex 523
- GroupSortingOrder 524
- HierarchyDataFieldIndex 524
- HistogramCollection 525
- HistogramSeparationLineColor 525
- IdentifyField 566
- IdentifyLayerAt 567
- IdentifyObjectAt 568
- ImportConfiguration 570
- InitialRowCount 526
- InsertLinkRecord 571
- InsertNodeRecord 571
- KeepingNodesTogetherDataFieldIndex 526
- LayerCollection 526
- LayersWithNonWorkInterval 527
- LeftTable 527
- LeftTableDiagramWidthRatio 528
- LineFormatCollection 528
- LinkAppearanceCollection 529
- LinkCollection 529
- LinkPredecessorDataFieldIndex 529
- LinksDataTableName 531
- LinkSuccessorDataFieldIndex 532
- LinkTypeDataFieldIndex 533
- Load 572
- MapCollection 534
- MinimumRowHeight 534
- MovingLayersAsNodeWithShiftKeyAllowed 535
- NodeCalendarNameDataFieldIndex 535
- NodeCollection 536
- NodeDurationDataFieldIndex 536
- NodeEndDateDataFieldIndex 536
- NodeRowNumberDataFieldIndex 537

NodesDataTableName 538  
 NodeSortingDataFieldIndex 538  
 NodeSortingOrder 539  
 NodeStartDateDataFieldIndex 540  
 NodesUseCalendars 540  
 NumericScaleCollection 540  
 OptimizeTimeScaleStartEnd 573  
 OverlapLayerEnabled 541  
 OverlapLayerName 541  
 Printer 541  
 PrintEx 573  
 PrintToFile 574  
 RecalculateAllStructureCodes 575  
 RequestImage 542  
 Reset 575  
 ResourceScheduler2 542  
 RoundedLinkSlantsEnabled 543  
 RowHeightReductionEnabled 543  
 RowMargins 544  
 SaveAsEx 576  
 Scheduler 544  
 ScrollEventsEnabled 545  
 ScrollToDate 576  
 ScrollToGroupLine 577  
 ScrollToNode 578  
 ScrollToNodeLine 579  
 SortGroups 580  
 SortNodes 580  
 StartDateForAutomaticScheduling 545  
 SubRowMargins 545  
 SummaryBarsVisible 546  
 SuspendUpdate 581  
 TextEntrySupplyingEventEnabled 546  
 TimeScaleCollection 547  
 TimeScaleEnd 547  
 TimeScaleStart 548  
 TimeUnit 549  
 TimeUnitsPerStep 549  
 ToolTipChangeDuration 550  
 ToolTipDuration 550  
 ToolTipPointerDuration 551  
 ToolTipShowAfterClick 551  
 ToolTipTextSupplyingEventEnabled 551  
 UpdateLinkRecord 582  
 UpdateNodeRecord 583  
 UpdateRowNumberFields 583  
 VcBoxLeftClicking 584  
 VcComponentScrolled 585  
 VcComponentScrolling 588  
 VcCurveLeftClicking 591  
 VcDataRecordModified 592  
 VcDataRecordModifying 593  
 VcDataRecordNotFound 594  
 VcDiagramHorizontalScrolled 594  
 VcDiagramHorizontalScrolling 596  
 VcDiagramLeftClicking 598  
 VcErrorOccurring 599  
 VcFieldSelecting 600  
 VcGroupLeftClicking 601  
 VcGroupModified 601  
 VcGroupModifying 602  
 VcGroupsMarked 604  
 VcGroupsMarking 605  
 VcHistogramLeftClicking 606  
 VcHistogramsHeightChanged 607  
 VcHistogramsHeightChanging 607  
 VcLinksLeftClicking 608  
 VcNodeLeftClicking 609  
 VcNodeModified 610  
 VcNodeModifiedEx 611  
 VcNodeModifying 612



- VcNodesMarked 614
- VcNodesMarking 614
- VcNumericScaleLeftClicking 615
- VcObjectDrawing 616
- VcObjectDrawn 618
- VcResourceSchedulingProgressing 619
- VcResourceSchedulingWarning 620
- VcTableCaptionLeftClicking 622
- VcTableColumnWidthChanging 623
- VcTableWidthChanging 624
- VcTextEntrySupplying 625
- VcTimeScaleLeftClicking 635
- VcTimeScaleSectionRescaled 636
- VcToolTipTextSupplying 636
- VerticalNodeMovementAllowed 552
- VerticalNodeMovementViaTableAllowed 552
- Zoom 584
- ZoomFactor 553
- ZoomingPerMouseWheelAllowed 553
- VcGroup 639**
  - DataField 640
  - DataRecord 647
  - Delete 647
  - GroupingLevel 641
  - ID 641
  - Marked 642
  - Name 642
  - NodeCollection 643
  - NodesAndGroupsBelowCollapsed 643
  - NodesArrangedInOneRow 644
  - NodesOptimized 644
  - RelatedDataRecord 648
  - ReOptimizeNodes 648
  - SubGroups 645
  - SuperGroup 645
  - Update 648
  - Visible 646
- VcGroupClickingEventArgs**
  - Event argument object of
    - VcGroupLeftClicking 601
- VcGroupCollection 650**
  - Count 650
  - FirstGroup 651
  - GetEnumerator 651
  - GroupName 652
  - NextGroup 652
  - SelectGroups 653
- VcGroupLeftClicking**
  - Event of
    - VcGantt 601
- VcGroupLevelLayout 654**
  - AllNodesInOneRow 655
  - CalendarGridName 655
  - CalendarGridsVisible 656
  - CalendarGridsWithChildGroups 656
  - CalendarNameDataFieldIndex 656
  - Collapsed 656
  - DateLineGridName 657
  - DateLineGridsVisible 657
  - DateLineGridsWithChildGroups 657
  - GroupDataFieldIndex 658
  - GroupNodesVisible 658
  - Level 658
  - ModificationsAllowed 658
  - Name 659
  - NodesArrangedOptimized 659
  - OptimizedNodesSortDataFieldIndex 659
  - OptimizedNodesSortOrder 660
  - OverlaidNodesSortDataFieldIndex 660

- OverlaidNodesSortOrder 660
- RowBackColorAsARGB 661
- RowBackColorDataFieldIndex 661
- RowBackColorMapName 661
- RowPattern 662
- RowPatternColorAsARGB 662
- RowPatternColorDataFieldIndex 662
- RowPatternColorMapName 663
- RowPatternDataFieldIndex 663
- RowPatternMapName 663
- SeparationLineColor 664
- SeparationLineColorDataFieldIndex 664
- SeparationLineColorMapName 664
- SeparationLinesVisible 665
- SeparationLinesVisibleAtTop 665
- SeparationLineThickness 665
- SeparationLineType 666
- SortDataFieldIndex 667
- SortOrder 667
- Specification 667
- SummaryBarsVisible 668
- Visible 668
- VcGroupLevelLayoutCollection 669**
  - Add 670
  - AddBySpecification 670
  - Copy 671
  - Count 669
  - FirstGroupLevelLayout 671
  - GetEnumerator 671
  - GroupLevelLayoutByIndex 672
  - GroupLevelLayoutByName 672
  - NextGroupLevelLayout 672
  - Remove 673
  - Update 673
- VcGroupModified**
  - Event of
    - VcGantt 601
- VcGroupModifiedEventArgs**
  - Event argument objekt of
    - VcGroupModified 602
- VcGroupModifying**
  - Event of
    - VcGantt 602
- VcGroupModifyingEventArgs**
  - Event argument objekt of
    - VcGroupModifying 603
- VcGroupsMarked**
  - Event of
    - VcGantt 604
- VcGroupsMarkedEventArgs**
  - Event argument objekt of
    - VcGroupsMarked 604
- VcGroupsMarking**
  - Event of
    - VcGantt 605
- VcHistogram 674**
  - CalendarGridsVisible 674
  - CalendarName 675
  - CurveCollection 675
  - FitRangeIntoView 677
  - GetCurrentYValues 677
  - Name 675
  - NumericScaleCollection 676
  - PutInOrderAfter 678
  - ScrollToValue 678
  - SetMaxYValue 679
  - SetMinYValue 679
  - Visible 676
- VcHistogramClickingEventArgs**
  - Event argument objekt of
    - VcHistogramLeftClicking 606
- VcHistogramCollection 681**
  - Active 681

- Count 682
- Delete 682
- FirstHistogram 683
- GetEnumerator 683
- HistogramByIndex 684
- HistogramByName 684
- NextHistogram 684
- VcHistogramLeftClicking**
  - Event of
    - VcGantt 606
- VcHistogramsHeightChanged**
  - Event of
    - VcGantt 607
- VcHistogramsHeightChangedEventArgs**
  - Event argument objekt of
    - VcHistogramsHeightChanged 607
- VcHistogramsHeightChanging**
  - Event of
    - VcGantt 607
- VcHistogramsHeightChangingEventArgs**
  - Event argument objekt of
    - VcHistogramsHeightChanging 608
- VcInterval 686**
  - BackgroundColor 687
  - CalendarProfileName 688
  - DayInEndMonth 688
  - DayInStartMonth 688
  - Duration 689
  - EndTime 689
  - EndMonth 689
  - EndTime 690
  - EndWeekday 690
  - LineColor 691
  - LineThickness 691
  - Pattern 692
  - PatternColor 695
  - Specification 695
  - StartDateTime 695
  - StartMonth 696
  - StartTime 696
  - StartWeekday 696
  - Text 697
  - TimeUnit 697
  - Type 697
  - UseGraphicalAttributes 698
- VcIntervalCollection 699**
  - Add 700
  - AddBySpecification 700
  - Copy 701
  - Count 700
  - FirstInterval 701
  - IntervalByIndex 702
  - IntervalByName 702
  - NextInterval 702
  - Remove 703
  - Update 703
- VcLayer 704**
  - BackgroundColor 705
  - BackgroundColorDataFieldIndex 706
  - BackgroundColorMapName 707
  - CalculateCurrentWidth 733
  - CompletionDataFieldIndex 709
  - DurationDataFieldIndex 709
  - EndDataFieldIndex 709
  - FilterName 710
  - Format 710
  - GraphicsFileName 710
  - GraphicsFileNameDataFieldIndex 712
  - GraphicsFileNameMapName 714
  - Height 715
  - HeightDataFieldIndex 715

- HorizontalOffset 716
- LabelSizeDependence 716
- LegendText 717
- LineColor 717
- LineColorDataFieldIndex 717
- LineColorMapName 718
- LineThickness 718
- LineType 719
- Movable 720
- Name 720
- ObjectDrawEventsEnabled 721
- Pattern 721
- PatternColor 724
- PatternColorDataFieldIndex 724
- PatternColorMapName 725
- PatternDataFieldIndex 725
- PatternMapName 726
- PutInOrderAfter 733
- Shape 728
- Sizeable 729
- Specification 729
- StartDataFieldIndex 729
- ThreeDEffect 730
- UsedAsOverlapLayer 730
- VerticalOffset 731
- VerticalOffsetDataFieldIndex 731
- VerticalOffsetMapName 731
- Visible 732
- VisibleInLegend 732
- VcLayerCollection 734**
  - Add 735
  - AddBySpecification 735
  - Copy 736
  - Count 734
  - FirstLayer 736
  - GetEnumerator 737
  - LayerByIndex 737
  - LayerByName 737
  - NextLayer 738
  - Remove 738
  - Update 739
- VcLayerFormat 740**
  - CopyFormatField 741
  - FormatField 740
  - FormatFieldCount 741
  - GetEnumerator 741
  - RemoveFormatField 742
- VcLayerFormatField 743**
  - Alignment 743
  - CalculateLineCount 749
  - ConstantText 744
  - FormatName 744
  - Index 744
  - MinimumWidth 745
  - Priority 745
  - TextDataFieldIndex 745
  - TextFont 746
  - TextFontColor 746
  - TextFontColorDataFieldIndex 746
  - TextFontColorMapName 746
  - TextFontDataFieldIndex 747
  - TextFontMapName 747
  - TextLineCount 747
  - TextLineCountDataFieldIndex 748
  - TextLineCountMapName 748
  - TruncatedTextSuppressed 748
- VcLineFormat 750**
  - CopyFormatField 752
  - FormatField 750
  - FormatFieldCount 751
  - Name 751
  - RemoveFormatField 752
  - Specification 751
- VcLineFormatCollection 753**

## **1076** Index

- Add 754
- AddBySpecification 754
- Copy 755
- Count 753
- FirstFormat 755
- FormatByIndex 756
- FormatByName 756
- NextFormat 757
- Remove 757
- VcLineFormatField 759**
  - Alignment 759
  - BackgroundColor 760
  - BackgroundColorDataFieldIndex 760
  - BackgroundColorMapName 761
  - ConstantText 761
  - DateOutputFormat 761
  - FormatName 763
  - Index 763
  - Pattern 763
  - TextDataFieldIndex 764
  - TextFontColor 764
  - TextFontColorDataFieldIndex 765
  - TextFontColorMapName 765
  - TextFontDataFieldIndex 765
  - TextFontMapName 766
  - TextLineCount 766
- VcLink 767**
  - AllData 767
  - DataField 768
  - DataRecord 770
  - Delete 770
  - ID 769
  - PredecessorNode 769
  - RelatedDataRecord 771
  - SuccessorNode 770
  - Update 771
- VcLinkAppearance 773**
  - FilterName 773
  - LineColor 774
  - LineThickness 774
  - LineType 776
  - Name 777
  - PredecessorLayerName 777
  - PredecessorPortSymbol 778
  - PutInOrderAfter 781
  - RoutingType 778
  - SuccessorLayerName 779
  - SuccessorPortSymbol 780
  - Visible 780
- VcLinkAppearanceCollection 782**
  - Add 783
  - AddBySpecification 784
  - Copy 784
  - Count 782
  - FirstLinkAppearance 784
  - GetEnumerator 785
  - LinkAppearanceByIndex 785
  - LinkAppearanceByName 786
  - NextLinkAppearance 786
  - Remove 787
  - Update 787
- VcLinkCollection 789**
  - Count 789
  - FirstLink 790
  - GetEnumerator 790
  - NextLink 790
  - SelectLinks 791
- VcLinksClickingEventArgs**
  - Event argument objekt of
    - VcLinksLeftClicking 608
- VcLinksLeftClicking**
  - Event of
    - VcGantt 608
- VcMap 793**

- ConsiderFilterEntries 793
- Count 794
- CreateEntry 796
- DeleteEntry 797
- FirstMapEntry 797
- GetEnumerator 794
- Name 794
- NextMapEntry 798
- Specification 795
- Type 795
- VcMapCollection 799**
  - Add 800
  - AddBySpecification 800
  - Copy 801
  - Count 799
  - FirstMap 801
  - GetEnumerator 802
  - MapByIndex 802
  - MapByName 802
  - NextMap 803
  - Remove 804
  - SelectMaps 804
  - Update 805
- VcMapEntry 806**
  - Color 806
  - DataFieldValue 807
  - FontBody 808
  - FontName 808
  - FontSize 809
  - GraphicsFileName 809
  - LegendText 811
  - Millimeter 811
  - Number 812
  - Pattern 812
- VcNode 816**
  - AllData 817
  - DataField 817
  - DataRecord 821
  - Delete 821
  - GetPositionInView 821
  - ID 818
  - IncomingLinks 818
  - Marked 819
  - NodeRowInView 822
  - OutgoingLinks 819, 820
  - OutlineIndent 823
  - OutlineOutdent 823
  - RelatedDataRecord 824
  - SetPositionInView 824
  - SuperGroup 820
  - Update 825
- VcNodeClickingEventArgs**
  - Event argument object of
    - VcNodeLeftClicking 610
- VcNodeCollection 827**
  - Count 827
  - FirstNode 828
  - GetEnumerator 828
  - NextNode 828
  - SelectNodes 829
- VcNodeLeftClicking**
  - Event of
    - VcGantt 609
- VcNodeLevelLayout 831**
  - CalendarGridName 831
  - CalendarGridsVisible 832
  - RowBackgroundColorAsARGB 832
  - RowBackgroundColorDataFieldIndex 832
  - RowBackgroundColorMapName 833
  - RowPattern 833
  - RowPatternColorAsARGB 833
  - RowPatternColorDataFieldIndex 834
  - RowPatternColorMapName 834

- RowPatternDataFieldIndex 834
- RowPatternMapName 835
- SeparationLineColor 835
- SeparationLineInterval 835
- SeparationLinesVisible 836
- SeparationLinesVisibleAtTop 836
- SeparationLineThickness 836
- SeparationLineType 837
- SortDataFieldIndex 837
- SortOrder 838
- VcNodeModified**
  - Event of
    - VcGantt 610
- VcNodeModifiedEventArgs**
  - Event argument objekt of
    - VcNodeModified 610
- VcNodeModifiedEx**
  - Event of
    - VcGantt 611
- VcNodeModifiedExEventArgs**
  - Event argument objekt of
    - VcNodeModifiedEx 611
- VcNodeModifying**
  - Event of
    - VcGantt 612
- VcNodeModifyingEventArgs**
  - Event argument objekt of
    - VcNodeModifying 613
- VcNodesMarked**
  - Event of
    - VcGantt 614
- VcNodesMarkedEventArgs**
  - Event argument objekt of
    - VcNodesMarked 614
- VcNodesMarking**
  - Event of
    - VcGantt 614
- VcNodesMarkingEventArgs**
  - Event argument objekt of
    - VcGroupsMarking 605
    - VcNodesMarking 615
- VcNumericScale 839**
  - BackgroundColor 839
  - Font 840
  - FontColor 840
  - Histogram 841
  - MajorTicks 841
  - MajorTicksEx 842
  - MinorTicks 842
  - MinorTicksEx 843
  - Name 843
  - Pattern 844
  - ThreeDEffect 845
  - Title 845
  - Unit 846
  - UnitLabel 846
  - UnitWidth 846
- VcNumericScaleClickingEventArgs**
  - Event argument objekt of
    - VcNumericScaleLeftClicking 616
- VcNumericScaleCollection 848**
  - Active 848
  - Count 849
  - FirstNumericScale 849
  - NextNumericScale 850
  - NumericScaleByIndex 851
  - NumericScaleByName 851
- VcNumericScaleLeftClicking**
  - Event of
    - VcGantt 615
- VcObjectDrawing**
  - Event of
    - VcGantt 616
- VcObjectDrawingEventArgs**

- Event argument objekt of
  - VcObjectDrawing 617
- VcObjectDrawn**
  - Event of
    - VcGantt 618
- VcObjectDrawnEventArgs**
  - Event argument objekt of
    - VcObjectDrawn 618
- VcPrinter 853**
  - AbsoluteBottomMarginInCM 854
  - AbsoluteLeftMarginInCM 854
  - AbsoluteRightMarginInCM 855
  - AbsoluteTopMarginInCM 855
  - Alignment 856, 857
  - CurrentHorizontalPagesCount 857
  - CurrentZoomFactor 858
  - CuttingMarks 858
  - DefaultPrinterName 858
  - DiagramEnabled 859
  - DocumentName 859
  - FitToPage 860
  - FoldingMarksType 860
  - MaxHorizontalPagesCount 862
  - MaxVerticalPagesCount 863
  - Orientation 863
  - PageDescription 864
  - PageDescriptionString 864
  - PageFrame 865
  - PageNumberMode 865
  - PageNumbers 866
  - PagePaddingEnabled 866
  - PaperSize 867
  - PrintDate 867
  - PrinterName 868
  - PrintPreviewWithFirstPage 868
  - ReOptimizeNodesInGroupsEnabled 869
- ScalingMode 869
- TableColumnRanges 870
- TableTimeScaleOnAllPages 870
- TimeScaleAdjustment 870
- ZoomFactorAsDouble 871
- VcRect 872**
  - Bottom 872
  - Height 872
  - Left 873
  - Right 874
  - Top 874
  - Width 875
- VcResourceScheduler2 876**
  - AssignmentDataTableName 879
  - AssignmentIsResultFieldIndex 881
  - AssignmentIsVisibleFieldIndex 881
  - AssignmentLoadOrConsumptionPerItemFieldIndex 882
  - AssignmentMaximumLoadFieldIndex 883
  - AssignmentMinimumLoadFieldIndex 883
  - AssignmentOperationIDFieldIndex 884
  - AssignmentResourceIDFieldIndex 885
  - AssignmentResourceSelectionStrategyFieldIndex 885
  - BaseTimeUnit 886
  - BaseTimeUnitsPerStep 887
  - DataRecordEventsEnabled 887
  - DefaultOperationMaximumInterruptionTime 888
  - DefaultResourceCalendarName 888
  - DetermineIDOfFirstOperationByTaskID 934
  - DetermineIDOfLastOperationByTaskID 935
  - FullUsageOfPlanningUnitsEnabled 889



- LinkDataTableName 889
- LinkDurationFieldIndex 891
- LinkPredecessorOperationIDFieldIndex 891
- LinkPredecessorTaskIDFieldIndex 892
- LinkSuccessorOperationIDFieldIndex 892
- LinkSuccessorTaskIDFieldIndex 893
- OperationDataTableName 894
- OperationLoadPerItemFieldIndex 895
- OperationMaximumInterruptionTimeFieldIndex 895
- OperationMinimumSupplementTimeFieldIndex 896
- OperationOverlapQuantityFieldIndex 897
- OperationPostLoadFieldIndex 899
- OperationPreparationLoadFieldIndex 899
- OperationResultEndDateFieldIndex 900
- OperationResultPostEndDateFieldIndex 901
- OperationResultPreparationStartDateFieldIndex 901
- OperationResultProcessingTimeFieldIndex 902
- OperationResultStartDateFieldIndex 902
- OperationResultStatusFieldIndex 903
- OperationRouteFieldIndex 904
- OperationSequenceNumberFieldIndex 904
- OperationStartLockDateFieldIndex 905
- OperationTaskIDFieldIndex 906
- OperationWorkInProgressFieldIndex 906
- PlanningEndDate 907
- PlanningStartDate 908
- PlanningStrategy 909
- Process 935
- ResourceCalendarNameFieldIndex 910
- ResourceCapacityType 910
- ResourceCapacityTypeFieldIndex 911
- ResourceConstraintTypeFieldIndex 912
- ResourceDataTableName 913
- ResourceEfficiencyFieldIndex 915
- ResourceGroupDataTableName 916
- ResourceGroupIDFieldIndex 917
- ResourceNameFieldIndex 917
- ResourceResultLoadCurveNamePrefix 918
- ResourceResultStockCurveNamePrefix 919
- ResourceSelectionStrategy 920
- ResourceType 921
- ResultProcessingStepCount 922
- TaskDataTableName 923
- TaskDueDateFieldIndex 924
- TaskPlanningStrategyFieldIndex 924
- TaskPriorityFieldIndex 925
- TaskQuantityFieldIndex 926
- TaskReleaseDateFieldIndex 927
- TaskResultEndDateFieldIndex 928
- TaskResultPostEndDateFieldIndex 928
- TaskResultPreparationStartDateFieldIndex 929
- TaskResultProcessingStepFieldIndex 929
- TaskResultProcessingTimeFieldIndex 930
- TaskResultRouteFieldIndex 931
- TaskResultStartDateFieldIndex 931
- ToleranceTimeOnASAPDueDates 932

- ToleranceTimeOnJITReleaseDates 932
- ToleranceTimeOnStartLockDates 933
- WorkInProcessType 933
- WritingDebugFilesEnabled 934
- VcResourceSchedulingProgressing**
  - Event of
    - VcGantt 619
- VcResourceSchedulingProgressingEventArgs**
  - Event argument objekt of
    - VcResourceSchedulingProgressing 619
- VcResourceSchedulingWarning**
  - Event of
    - VcGantt 620
- VcRibbon 937**
  - BackgroundColor 938
  - CalendarName 938
  - DateOutputFormat 938
  - Font 940
  - FontColor 941
  - MajorTicks 941
  - MinorTicks 942
  - Pattern 942
  - Position 943
  - ReferenceDate 943
  - TextAlignment 944
  - TickColor 944
  - TickPosition 945
  - Type 945
  - UnitSeparation 946
  - UseReferenceDate 946
- VcScheduler 947**
  - ActualEndDateDataFieldIndex 948
  - ActualStartDateDataFieldIndex 948
  - AutomaticSchedulingEnabled 948
  - DurationDataFieldIndex 949
  - EarlyEndDateDataFieldIndex 949
  - EarlyStartDateDataFieldIndex 949
  - EndDateForAutomaticScheduling 949
  - EndDateNotLaterThanDataFieldIndex 950
  - FreeFloatDataFieldIndex 950
  - LateEndDateDataFieldIndex 950
  - LateStartDateDataFieldIndex 951
  - LinkDurationDataFieldIndex 951
  - ScheduledProjectEndDate 951
  - ScheduledProjectStartDate 952
  - ScheduleProject 953
  - ScheduleSuccessorsOnlyEnabled 952
  - StartDateForAutomaticScheduling 952
  - StartDateNotEarlierThanDataFieldIndex 952
  - TotalFloatDataFieldIndex 953
- VcSection 955**
  - CalendarGrid 955
  - DateLineGrid 956
  - NonWorkIntervalsCollapsed 956
  - Ribbon 957
  - StartDate 957
  - TimeUnit 958
  - UnitWidth 959
- VcTable 960**
  - ColumnTitle 960
  - ColumnWidth 961
  - OptimizeColumnWidth 963
  - Position 961
  - TableFormatCollection 962
  - Visible 962
- VcTableCaptionClickingEventArgs**
  - Event argument objekt of

- VcTableCaptionLeftClicking 622
- VcTableCaptionLeftClicking**
  - Event of
    - VcGantt 622
- VcTableCollection 964**
  - TableByIndex 964
- VcTableColumnWidthChanging**
  - Event of
    - VcGantt 623
- VcTableColumnWidthChangingEventArgs**
  - Event argument objekt of
    - VcTableColumnWidthChanging 623
- VcTableFormat 965**
  - CollapseColumn 966
  - FieldsSeparatedByLines 966
  - FilterName 967
  - FormatField 967
  - FormatFieldCount 968
  - GetEnumerator 971
  - IndentColumn 968
  - IndentWidth 969
  - Name 969
  - SeparationLineColor 970
  - ThreeDEffect 970
- VcTableFormatCollection 972**
  - Count 972
  - FirstFormat 973
  - FormatByIndex 973
  - FormatByName 974
  - GetEnumerator 974
  - NextFormat 975
- VcTableFormatField 976**
  - Alignment 977
  - BackgroundColor 977
  - BackgroundColorDataFieldIndex 978
  - BackgroundColorMapName 978
  - BottomMargin 978
  - ConstantText 979
  - FormatName 979
  - GraphicsFileName 979
  - GraphicsFileNameDataFieldIndex 980
  - GraphicsFileNameMapName 981
  - GraphicsHeight 981
  - Index 981
  - LeftMargin 981
  - MaximumTextLineCount 982
  - MinimumTextLineCount 982
  - MultiState 983
  - Pattern 983
  - RightMargin 984
  - TextAndGraphicsCombined 984
  - TextDataFieldIndex 984
  - TextFont 984
  - TextFontColor 985
  - TextFontColorDataFieldIndex 985
  - TextFontColorMapName 985
  - TextFontDataFieldIndex 986
  - TextFontMapName 986
  - TopMargin 986
  - Type 986
- VcTableWidthChanging**
  - Event of
    - VcGantt 624
- VcTableWidthChangingEventArgs**
  - Event argument objekt of
    - VcTableWidthChanging 624
- VcTextEntrySupplying**
  - Event of
    - VcGantt 625
- VcTextEntrySupplying event 150**
- VcTextEntrySupplyingEventArgs**

- Event argument objekt of
  - VcTextEntrySupplying 625
- VcTimeScale 988**
  - BackgroundColor 988
  - CalendarGridsVisible 989
  - DateGridsVisible 989
  - Font 990
  - FontColor 990
  - Name 990
  - Ribbon 991
  - Section 991
  - ThreeDEffect 992
- VcTimeScaleClickingEventArgs**
  - Event argument objekt of
    - VcTimeScaleLeftClicking 635
- VcTimeScaleCollection 993**
  - Count 993
  - FirstTimeScale 994
  - GetEnumerator 994
  - NextTimeScale 995
  - TimeScaleByIndex 995
  - TimeScaleByName 996
- VcTimeScaleLeftClicking**
  - Event of
    - VcGantt 635
- VcTimeScaleSectionRescaled**
  - Event of
    - VcGantt 636
- VcToolTipTextSupplying**
  - Event of
    - VcGantt 636
- VcToolTipTextSupplying event 150**
- VcToolTipTextSupplyingEventArgs**
  - Event argument objekt of
    - VcToolTipTextSupplying 637
- VerticalNodeMovementAllowed**
  - Property of
    - VcGantt 552
- VerticalNodeMovementViaTableAllowed**
  - Property of
    - VcGantt 552
- VerticalOffset**
  - Property of
    - VcLayer 731
- VerticalOffsetDataFieldIndex**
  - Property of
    - VcLayer 731
- VerticalOffsetMapName**
  - Property of
    - VcLayer 731
- Viewer Metafile (\*.vmf) 141**
- Visible**
  - Property of
    - VcBox 319
    - VcCalendarGrid 375
    - VcCurve 401
    - VcDateLine 473
    - VcDateLineGrid 488
    - VcGroup 646
    - VcGroupLevelLayout 668
    - VcHistogram 676
    - VcLayer 732
    - VcLinkAppearance 780
    - VcTable 962
- VisibleInLegend**
  - Property of
    - VcLayer 732

## W

### Width

- Property of
  - VcRect 875

### Width ratio

table/complete diagram 159

**WorkInProcessType**

Property of

VcResourceScheduler2 933

**WritingDebugFilesEnabled**

Property of

VcResourceScheduler2 934

**Z**

**Zoom**

adjust the diagram to window size  
while keeping the height-to-width-  
ratio 559

Method of

VcGantt 584

**ZoomFactor**

Property of

VcGantt 553

**ZoomFactorAsDouble**

Property of

VcPrinter 871

**ZoomingPerMouseWheelAllowed**

Property of

VcGantt 553